How Secure is Secure? On Message and IV Lengths for Synchronous Stream Ciphers

E. Zenner and M. Boesgaard

CRYPTICO A/S
Fruebjergvej 3
2100 Copenhagen
Denmark
info@cryptico.com

Abstract. In this paper, we discuss security properties for synchronous stream ciphers. Let the key size and claimed security level be k. Following an initial proposal by Hawkes and Rose [6], we argue that a synchronous stream cipher should encrypt at most $2^{k/2}$ plaintext blocks before changing the key. We claim further that a synchronous stream cipher should ideally provide for IV sizes of both $k/2$ and k, with the second size being mandatory. In this context, recent results by Hong and Sarkar [3] are also briefly discussed.

1 Introduction

An historical perspective: The NESSIE project [4] was a project within the Information Societies Technology (IST) Programme of the European Commission, running from 2000 to 2003. Its purpose was the selection of a portfolio of strong cryptographic algorithms. These algorithms were grouped into several categories, one of them being stream ciphers.

During the project, a discussion erupted on how strong a stream cipher has to be in order to be considered secure. It was argued [6] that distinguishing attacks using unrealistic amounts of data (e.g. in the order of 2^{95} words of known plaintext) should not be considered a threat to the security of the cipher. No academic consensus was reached on this issue, and no candidate stream cipher submitted for NESSIE was accepted as as part of the suggested portfolio.

As a follow-up to NESSIE, the Ecrypt Network of Excellence [1] was called to life in 2004, also as part of an IST Programme. One of the objectives of this project is to improve the understanding of stream cipher security. To this end, a call for stream cipher primitives was issued, with the goal of collecting both design and cryptanalysis results for these ciphers. It is to be expected that during the course of the project, the open questions from the NESSIE project will have to be resolved.

This paper aims at providing some first thoughts on how such a solution could look like. We build on initial ideas proposed by Rose and Hawkes in [6] and extend them to obtain some more general results on the maximum message length available to the attacker and on the required length of the IV vectors. We do not claim academic novelty on our findings, but aim at collecting arguments and at initiating a discussion on what it takes for a stream cipher to be considered secure.

Organisation of the paper: In section 2, we introduce the general concept of a synchronous stream cipher and repeat the birthday paradox. Section 3 builds on [6] and argues for an upper bound on the data to be encrypted under one key. In section 4, we give arguments for IV sizes required for stream ciphers. Section 5 briefly discusses the results from [3]. Section 6 concludes the paper.

2 Concepts and Definitions

Synchronous stream ciphers: A synchronous stream cipher transforms a message into a ciphertext (or vice versa) under the control of a secret key and a public initialisation vector (IV). It maintains an internal state and operates in two phases:

1. In the *setup phase*, the key and IV are used to initialise the internal state of the stream cipher.
2. In the *encryption/decryption phase*, the message is transformed into the ciphertext (or vice versa). In order to do this, the message and ciphertext are broken into blocks of w bits, and for each such block, the following steps are executed:
   - Update the internal state of the cipher,
   - extract a w-bit keystream block of pseudorandom data, and
   - combine that block with the message/ciphertext block using a suitable group operation, e.g., bitwise xor.

For more formal models of stream ciphers see [7, 9].

Throughout the paper, we will be mostly interested in the relationship between the different parameters of a stream ciphers. The notation used for these sizes is given in Table 1.

Attack model: The classical attack model against ciphers uses an oracle that either contains a random function R or a cipher instance E_k. The attacker sends message blocks $p_i \in \{0,1\}^w$, positions $j_i \in \mathbb{N}$, and IVs $n_i \in \{0,1\}^v$ to the oracle and obtains outputs as follows:

- If the oracle contains a random function, then he obtains a random value.
- If the oracle contains a cipher instance, then he obtains the valid encryption of p_i if encrypted under key k and IV n_i at position j_i of a message stream.

The attacker is not allowed to request two encryptions under the same position/IV pair (j, n); the reason for this is explained in section 4. He is considered successful if by using less computational power than required for 2^k block encryptions, he is able to tell whether the oracle contains a random function or an instance of the cipher.

The above security model is simplified, and it is sometimes disputed in discussion. It is, however, the one that covers all other attack models; a cipher that is secure in this model is also secure, e.g., against prediction or key reconstruction. Thus, we will use it in the following discussion.

The birthday paradox: In cryptography, the probability of finding a collision after drawing a number of samples from a fixed set A of values plays an important role. While the mathematical details of these probabilities tend to be rather messy (see, e.g., [5]), some simple approximations are often sufficient to estimate the work involved in cryptanalysis:

- Let an urn contain all elements of A. Drawing S samples with replacement, there is a high probability of obtaining a collision if $S \geq \sqrt{A}$.
- Let two urns each contain all elements of A. Drawing S_1 samples from urn 1 with and S_2 samples from urn 2 without replacement, then there is a high probability of obtaining a collision between the sample sets if $S_1 \cdot S_2 \geq A$.

Parameter	Notation
Key size	k bit
IV size	v bit
Block size	w bit
Message size	$l \cdot w$ bit

Table 1. Notation for parameter sizes

3 Data Available to the Attacker

In this section, we elaborate on the ideas proposed by Rose and Hawkes in [6].

Block ciphers in stream cipher mode: In a strict sense, a block cipher $E_k(\cdot)$ is a family of pseudo-random permutations. As a consequence, if the same message is encrypted twice using a block cipher, the output will be the same in both cases. Thus, block ciphers should never be used directly in this way (called ECB mode), but should use a stream cipher mode of operation instead. The most well-known synchronous modes of operation are CTR and OFB mode[1], which are widely used and will be shortly introduced, following [2].

Let p_i and c_i denote that i-th plaintext and ciphertext block, respectively. Then *counter mode* (CTR) is defined by

$$z_i = E_k(\mathrm{IV}||i)$$
$$c_i = z_i \oplus p_i$$

for $i = 1, \ldots, l$.

On the other hand, *output feedback mode* (OFB) is defined by $z_0 = \mathrm{IV}$ and

$$z_i = E_k(z_{i-1})$$
$$c_i = z_i \oplus p_i$$

for $i = 1, \ldots, l$.

Some birthday attacks: For the following attacks, the attacker uses the oracle to extract slightly more than $2^{k/2}$ subsequent keystream blocks z_i (by computing $p_i \oplus c_i$) under the same IV.

In order to attack CTR mode, observe that a block cipher is a permutation. A sequence of subsequent keystream blocks $(z_1, \ldots, z_l)$ has the property that no keystream block ever repeats. This leads to a simple distinguishing attack: After roughly $l \approx 2^{k/2}$ keystream blocks of a random function, the sequence $z_1, \ldots, z_l$ would contain at least one keystream block twice (this is a consequence of the first birthday paradox). If, however, this does not happen after slightly more than $2^{k/2}$ subsequent keystream blocks, there is a high probability that we are dealing with a block cipher in counter mode.

A similar attack can be applied against OFB mode by observing that if any keystream block is ever repeated, then the subsequent keystream blocks are also identical (i.e. if $z_i = z_j$, then it follows that $z_{i+1} = z_{j+1}$ a.s.o.). For a random function, however, this is not the case - even if $z_i = z_j$, we will have $z_{i+1} \neq z_{j+1}$ with overwhelming probability. Since for a random function, a collision will have occured after slightly more than $2^{k/2}$ keystream blocks, it can then be distinguished from a cipher instance. This attack will even remain valid if the cipher is frequently re-initialized with a new nonce.

Consequences for stream ciphers: The general understanding is that dedicated stream ciphers are built either for high speed or low hardware requirements while being slightly inferior to block ciphers in security (see, e.g., [8]). It follows from this that the security goals for stream ciphers should not be harder to obtain than those for a block cipher in an appropriate stream cipher mode.

We have seen that for the two most common stream cipher modes, a distinguishing attack becomes possible after roughly $2^{k/2}$ keystream blocks have been produced. Nonetheless, these modes are not considered insecure - it is just recommended that the key is changed after $2^{k/2}$ blocks have been encrypted. We consider it a realistic restriction to ask the same for stream cipher encryption:

[1] CBC mode may be even more wide-spread in practice, but it is a self-synchronizing mode of operation and is thus not considered in this text.

When attacking a stream cipher with a k-bit key, the attacker can request at most $2^{k/2}$ keystream blocks from the encryption oracle.

4 Requirements on the IV Length

The purpose of the IV: Synchronous stream ciphers as described in section 2 must ensure that keystream blocks are never re-used. If plaintext blocks p are combined with keystream blocks z using some addition $+$, then multiple use of the same z implies that

$$\begin{aligned} c_i &= p_i + z \\ c_j &= p_j + z \\ \Rightarrow\ c_i - c_j &= p_i - p_j, \end{aligned} \tag{1}$$

i.e. that the attacker can obain information about p_i and p_j from the ciphertexts. In order to avoid this problem, it would be necessary to run the PRG underlying the stream cipher without ever re-setting it, which can cause synchronization problems and requires storing the full inner state of the stream cipher.

In order to solve this problem, it is common practice to make the keystream not only dependent on the key, but also on a (public) IV which is changed upon re-synchronization. This way, the generator will produce a new keystream every time it is invoked (with the exception of random collisions of the keystream blocks).

Models of IV useage: There are different ways of handling IVs, e.g.,

1. it is made sure that no IV is ever re-used, e.g., a counter is used as IV, or
2. the IV is drawn at random from a set of possible values.

There are also intermediate solutions, where an initial IV is drawn at random and some subsequent IVs are computed deterministically, before the next random IV is determined. However, the theoretical considerations can be limited to the two extreme cases of complete control or complete randomness.

Some more birthday attacks: Remember that according to section 3, the attacker is able to obtain at most $2^{k/2}$ subsequent keystream blocks. In order to protect against the attack described above, the attacker must not be able to obtain two encryptions under the same position/IV pair (j, n); otherwise, he will immediately have a successful distinguishing attack using equation 1.

Assuming that the IV is not under the control of the attacker, when will this condition be violated?

- In case of a counter IV, a position/IV pair will only be re-used after 2^v re-initializations. Since the attacker has only $2^{k/2}$ different keystream blocks at his disposal, it is possible to avoid this kind of attack if $v = k/2$.
- In case of a random IV, there is a chance that the same IV is drawn twice. The birthday paradox dictates that this happens after approx. $2^{v/2}$ attempts. If the attacker has $2^{k/2}$ different keystream blocks available, this means that we have to choose $v = k$ in order to avoid this kind of attack.

Note that in the intermediate case of choosing a random IV once in a while and increasing it as a counter, the chances for the attack are somewhere between those two extremes, though closer to the second case.

Consequences for IV lengths: Thus, the necessary IV size depends on the way the IV is actually determined. The following rules apply:

- If the IV is strictly a counter, then $v = k/2$ is a sufficient IV size.
- If the IV is determined randomly, then $v = k$ is required as IV size.

Since the security of a cipher should not depend on the mode it is deployed in, we propose that a stream cipher design should ideally provide for IV sizes of $v = k/2$ and $v = k$, with the second size being mandatory.

5 A Remark on Hong's and Sarkar's Paper

Recently, Hong and Sarkar stated [3] that applying Time-Memory-Tradeoff attacks in a consequent way, no stream cipher could provide security of k bits unless the IV is also chosen such that $v \geq k^2$. However, this result is based on the assumption that the attacker is precomputing a large key/IV lookup table and then applies it against encryptions under several keys.

It remains to be seen whether or not this constitutes a valid attack, since in essence, Hong and Sarkar do not propose a new attack, but a new security model for stream ciphers. So far, the attacker was entitled to see encryptions under *one* key and had to distinguish the output from random. In the Hong/Sarkar model, however, the attacker gets to see encryptions under *different* keys. We would like to point out that if this model is valid, then a similar property has always been known to hold for most keyed cryptographic algorithms in use (e.g. block and stream ciphers, message authentication codes, signatures, a.s.o.): If the key space has size 2^k, and if computational outputs for at least 2^d different keys are available to the attacker, then the attacker is able to reconstruct a key from a precomputed table of 2^{k-d} output-/key pairs with very high probability. The probability for the *individual* key to be reconstructed will still not be larger than 2^{-k}, as it should be. The only significant difference between synchronous stream ciphers and most other cryptographic primitives is that for the synchronous stream ciphers, this generic attack can be executed on a known-plaintext basis, while for most other primitives, it has to be chosen plaintext.

It can, of course, be argued that the use of different IVs under the same key also can be considered as an encryption under different keys of length $k + v$. However, in this case, all attacks presented in [3] require a precomputation time that is equivalent to at least 2^k trial encryptions, i.e. to brute force search. Thus, we believe that the attacks presented by Hong and Sarkar are only valid if attacking encryptions under several different keys is the attack model of choice. Whether or not this attack model will be accepted, remains to be discussed.

6 Conclusions

We have given brief arguments for the following security requirements for stream ciphers:

1. A synchronous stream cipher should encrypt at most $2^{k/2}$ plaintext blocks before changing the key.
2. A synchronous stream cipher should ideally provide for IV sizes of $v = k/2$ and $v = k$, with the second size being mandatory.

We would also like to point out that if a consensus should develop that a synchronous stream cipher has to offer protection for a system that uses several keys, then the IV size of $v = k$ proposed in [3] is the only choice left.

[2] To be more precise, the *entropy* of the IV has to be at least the key size.

References

1. Ecrypt Network of Excellence in Cryptography. http://www.ecrypt.eu.org/.
2. N. Ferguson and B. Schneier. *Practical Cryptography*. Wiley, 2003.
3. J. Hong and P. Sarkar. Rediscovery of time memory tradeoffs. IACR eprint 2005/090, http://eprint.iacr.org/2005/090.
4. New European Schemes for Signatures, Integrity, and Encryption (NESSIE). https://www.cosic.esat.kuleuven.ac.be/nessie/.
5. K. Nishimura and M. Sibuya. Probability to meet in the middle. *Journal of Cryptology*, 3(2):13–22, 1990.
6. G. Rose and P. Hawkes. On the applicability of distinguishing attacks against stream ciphers. 3rd NESSIE workshop, available at http://www.qualcomm.com.au/PublicationsDocs/StreamAttack.pdf, Nov. 2002.
7. R. Rueppel. *Analysis and Design of Stream Ciphers*. Springer, 1986.
8. A. Shamir. Stream ciphers: Dead or alive. Slides of Invited Talk, Asiacrypt 2004, http://www.iris.re.kr/ac04.
9. E. Zenner. *On Cryptographic Properties of LFSR-based Pseudorandom Generators*. PhD thesis, University of Mannheim (Germany), May 2004.