Stream Cipher Criteria

Erik Zenner

CRYPTICO A/S
info@cryptico.com

Abstract. The eStream project has given us a number of insights into design criteria for stream ciphers. Some of them were disputed, on others everyone seemed to agree. This text tries to collect these criteria into one comprehensive document and gives the author's views on their relevance for stream cipher design. It does not contain any groundbreaking new results; it is merely meant as a collection of ideas and an incentive for further discussion.

1 Preliminary remarks

1.1 Stream ciphers are bleeding-edge technology

On several occasions, the question was asked why the development of a (dedicated) stream cipher is necessary. After all, there exists a cryptographic standard AES whose security has been thoroughly scrutinized and that is sufficiently efficient both in hardware and software on most platforms. Using simple modes of operations, AES itself can be turned into a stream cipher. So why develop a dedicated stream cipher at all?

The answer is that dedicated stream ciphers can be designed such that they are even more efficient, since they are less universal cryptographic building blocks. In order to outweigh the advantages of AES in terms of analysis, standardization, and availability, stream ciphers have to provide a *significant* advantage on at least one platform. Being slightly better just is not enough to justify the extra effort on the end user's side.

Another consequence is that stream ciphers do not necessarily have to be perfect on all platforms and in all situations. There may, e.g., be room for a cipher that is secure only if less than 2^{20} output bits are encrypted. However, this restriction has to be clearly stated in the documentation. Whether such a special-purpose cipher should be recommended by eStream should become an issue of public discussion.

1.2 Stream ciphers are not pseudorandom generators

A widespread misconception is the confusion between a stream cipher and a pseudorandom generator (PRG). Many stream cipher designs (including the majority of those submitted to eStream) are based on PRGs, but that does not mean that they are identical. The task of a PRG is to expand a seed value

into an output stream that is indistinguishable from random. On the other hand, a stream cipher transforms a message, key, and initialization vector (IV) into an output stream in such a way that the output stream can not be distinguished from random under a variety of attack scenarios, including known or chosen message, known or chosen IV, multiple keys, or related keys. A stream cipher is thus a more complex cryptographic building block than a pseudorandom generator.

However, many stream ciphers can be used as PRGs by setting the message and IV to zero. When proposing a stream cipher for widespread use, one has to be aware that it will most likely be used as a PRG. This should not have any security implications, since breaking the resulting PRG would immediately indicate a break of the stream cipher in a chosen-message, chosen-IV scenario. There are, however, implications with regards to the performance, as will be discussed below.

2 Security

2.1 Long key streams

A recurring question concerns the amount of data required to attack a stream cipher. The validity of an distinguishing attack against the cipher SNOW 1.0 that required 2^{95} words of known plaintext and ciphertext was questioned by Rose and Hawkes in [5]. In the SASC 2006 discussion, it was pointed out that the biggest applications run today operate on Terabyte (2^{40} - 2^{50} bytes), and even then it seems questionable whether all this data would be encrypted under the same IV.

For most 128-bit block cipher modes of operation, distinguishing attacks become possible after roughly 2^{64} output blocks, and this is not considered to be a problem in practice. If such a gigantic amount of data would ever be reached, all that would be necessary would be the use of a new IV. Given that dedicated stream ciphers are considered bleeding-edge technology, it seems only reasonable to allow for similar restrictions on the use of stream ciphers.

However, the authors of a cipher should always document the maximum amount of data that may be encrypted before a new IV should be used. If this was done, an attack that requires more than the maximum amount of data should not be considered as a break, but as an indication of the security margin that is left until a break becomes possible.

2.2 Special kinds of attacks

Once in a while, the question is raised whether or not distinguishing attacks, side-channel attacks, multiple-key attacks, related-key attacks etc. are relevant. The answer to this question depends indeed completely on the application scenario. There may exist situations where a distinguishing attack will not reveal anything useful, where a side-channel attack is not possible etc.

However, if only a small portfolio of ciphers is to be recommended by eStream, then these ciphers should be as universal as possible. A list of usage restrictions

would not only reduce the usefulness of the recommendation, it would also induce an additional risk that ciphers are not used in the proper way. Besides, it may turn out to be difficult to state the usage restrictions in an unambiguous manner[1].

2.3 Key and IV sizes

For the eStream project, two profiles are considered:

1. Profile 1 ciphers are targeted at software platforms and provide a security level of 128 bit or 256 bit.
2. Profile 2 ciphers are targeted at hardware platforms and provide a security level of 80 bit.

The actual call for contributions requests *key lengths* of 80, 128, or 256 bit, but as was observed by Bernstein in [2], the security level does not necessarily have to correspond to the key length. Thus, we assume that the ciphers are indeed supposed to provide *security levels* of 80, 128, or 256 bit. We will denote this security level by L.

Required key lengths: According to the classical cryptographical paradigm for stream ciphers, the key length should be equal to L. However, if attacks against several keys at once are an issue, then Hong and Sarkar show in [4] that the added lengths of the key and IV should be $2L$.

If a stream cipher has a fixed key size of L, then the problem is propagated to the IV size. However, the IV size depends strongly on the application at hand. The IV length that can be used might be severely limited, e.g., by the amount of payload a system can handle. In the most extreme case, no IV is to be used at all. In this case, the problem is propagated to the key size.

Thus, a general-purpose stream cipher should be able accomodate key lengths from L up to $2L$. Where these requirements are not met, the usage restrictions should be clearly documented.

Required IV lengths: For the same reasons as those given in the paragraph on long key streams, it seems safe to assume that no more than 2^{64} different IVs are required before the key is exchanged.

However, as was pointed out by Zenner and Boesgaard in [6], a system that uses random IVs (instead of counters) requires a much larger IV length, namely 128 bit, in order to avoid random collisions of IVs. Analoguously, a cipher in profile 2 that uses no more than 2^{32} IVs should in fact be able to support an IV length of 64 bit.

[1] Just try to define the usage restrictions implied by the fact that distinguishing attacks were disregarded in the analysis. If perfect compression of the plaintext existed, this seems possible. But since this is not the case, the usage restriction depends both on (a) the plaintext entropy and on (b) the quality of the distinguishing attacks involved. If (a) is determined by the application and research in (b) was discouraged, no useful statement will be possible

Similar conclusions are obtained from the results by Hong and Sarkar [4] and by de Cannière et al. [3]. If IVs are used to increase resistance against attacks that target several keys, then IV lengths up to L should be accomodated by the cipher. Again, if any of these options are not present, the limitations on cipher use should be clearly documented.

3 Performance

3.1 Speed and Compactness

Independent of its actual security, it will be almost impossible for an eStream candidate to obtain the same level of public trust that the block cipher standard AES enjoys. Thus, if the cipher actually is to be used, it has to offer other advantages. Typically, this means that it has to be faster (in software) or more compact (in hardware).

From our experience, these advantages have to be significant. For most users, the cipher decision is based on the fact that for products bases on AES, a wide range of cheap software and hardware implementations are available, certification is easy, and advertising as "using the standard" is possible[2]. No user will give up these advantages lightly. Thus, we expect that for the profile under consideration, a stream cipher recommended by eStream has to provide a *clear* advantage over AES. Being just 20 percent faster or 20 percent more compact simply is not going to be enough.

It is possible that a cipher offers other advantages than speed and compactness. In this case, we strongly recommend that those are documented, clearly answering the question why or in which situations the cipher in consideration should be used instead of the AES.

3.2 Tradeoff between Throughput and State Size

When comparing software performance for the eStream candidates with their inner state size, then we observe a tradeoff: The ciphers providing the highest throughput rates tend to have rather large inner states[3]. The reason is roughly that with large inner states, relatively little information about this state is leaked with each output. This reduces the need for a strong inner mixing between consecutive outputs; a principle that already underlied the RC4 cipher.

It is known that the inner state size has to be at least $2L$ in order to prevent time-memory tradeoff attacks. In recent years, it turned out that ciphers not using at least $4L$ usually got broken, but since no generic attack against such

[2] Practice also shows that most users know a lot less about cryptography than any cryptographer would dare to imagine. Explaining cryptographic properties to them is typically futile. They will ask for the FIPS 140-2 certification, and they will not even change their opinion when they learn that encryption algorithms *can not* be certified under that standard.

[3] Or to get broken. Or both.

designs exists, this may just have been a coincidence. Whether increasing the throughput further at the expense of the inner states size is useful or not depends on a number of factors:

- Large inner states require intensive key and IV setup. Large key setup times can be tolerated by most applications, and they might even provide a few extra bits of security against brute-force attacks. Large IV setup times, however, become a hindrance when switching to packet-by-packet encryption, as is the case in most practical applications. This problem becomes worse the shorter the packets to be encrypted are.
- Large inner states decrease key agility. If an application has to handle several instances of the same cipher (e.g., a server that handles a lot of TLS connections at once), then the time for loading and storing the inner state of each instance becomes important.
- Large inner states are too costly for low-resource applications like embedded processors and hardware implementations. For those applications, look-up tables like e.g. S-boxes are also problematic.
- Large inner states can be problematic when a cipher is not used as stream cipher, but as pseudo-random generator (PRG). There are applications of PRGs that require a very frequent re-keying[4]. In such a case, the large inner state turns out to be a significant disadvantage.

Again, we have to remember that stream ciphers are specialized technology: Ciphers that use huge inner states to obtain extreme throughput rates may have their value in some applications. If the cipher is to be used in a wide variety of applications, however, smaller inner state sizes are recommended.

4 Last but not least: An appeal

The paradigm of scientific publishing dictates that only results that are better than prior ones are to be presented on conferences or included in journals. However, the impact of this paradigm on evaluating the security of a stream cipher (or a cryptographic building block in general) is a negative one. The default attack on each cipher is brute force, and every attack that was tried and did not improve over brute force is discarded.

As a result, everyone who analyses a cipher has to go through the same steps again, and as a community, we still don't know how many people in fact did analyse a cipher and did not find a weakness. Even worse: Ciphers that get broken and fixed repeatedly end up enjoying a higher level of trust than the ones that never ever got broken, because we simply don't have any results on the unbroken ones. In the worst case, an author will exaggerate his/her non-attack

[4] As an example, consider the robust PRG described by Barak and Halevi in [1]. If K is the stream cipher's key length, then it produces only K bits of output between two key setups. It is possible to change the construction to be more efficient, but then the whole proof of security has to be re-worked and re-verified again.

result such that it may sound like a break, damaging the reputation of a good design simply because otherwise, he/she won't get any academic credit.

We do propose to shift the paradigm from *"Tell me something I couldn't work out myself"* to *"Tell me something I didn't know"*. Many cryptographers involved in the eStream project have the skill to check all 35 stream cipher candidates against a large number of standard attacks. But most of us definitely do not have the time to do that. Thus, we should realize that a failed cryptographic attack is also of significant importance to the community. It may not be publishable at a scientific conference, but it should by all means be published at the eStream web pages. Especially a project like eStream should be understood not only as a science, but also as an engineering endeavour: We should start to talk not only about the ingenious new ideas, but also about the routine jobs that are of relevance to other participants.

5 Conclusions

In this paper, we have discussed a number of design criteria for stream ciphers. In way of conclusion, we propose the following consequences for cipher design and specifications that should be implemented during phase 2 of the eStream project:

- It should be publicly discussed whether eStream aims at finding a stream cipher, a PRG, or both.
- It should be publicly discussed whether special-purpose ciphers should also be part of the eStream recommendations, or whether the recommended candidates should be as general as possible.
- For each candidate, it should be clearly stated by the authors in which respect it offers an advantage over AES (counter mode), and how big this advantage is.
- For each candidate, all usage restrictions (max. output length under one IV, max. number of IVs under one key, max. number of keys, susceptibility to side-channel attacks etc.) should be clearly stated by the authors.
- For ciphers not making special restrictions on their usability, both key setup and IV setup should accomodate a key length of $2L$ and an IV length of L, where L is the claimed security level.
- Cryptographers involved in the project should be encouraged to publish their analysis results on eStream candidates on the project's web pages, even if the results only indicate resistance of the cipher against an attack.

References

1. B. Barak and S. Halevi. A model and architecture for pseudo-random generation with applications to /dev/random. http://eprint.iacr.org/2005/029.pdf, 2005.
2. D. Bernstein. Understanding brute force. http://www.ecrypt.eu.org/stream/papersdir/036.pdf, 2005.

3. C. de Cannière, J. Lano, and B. Preneel. Comments on the rediscovery of time memory data tradeoffs.
http://www.ecrypt.eu.org/stream/papersdir/040.pdf, 2005.
4. J. Hong and P. Sarkar. Rediscovery of time memory tradeoffs. IACR eprint 2005/090,
http://eprint.iacr.org/2005/090.
5. 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.
6. E. Zenner and M. Boesgaard. How secure is secure? On message and IV lengths for synchronous stream ciphers.
http://www.ecrypt.eu.org/stream/papersdir/039.pdf, 2005.