Parallel authenticated encryption with the duplex construction

Paweł Morawiecki[1] and Josef Pieprzyk[2]

[1] Section of Informatics, University of Commerce, Kielce, Poland

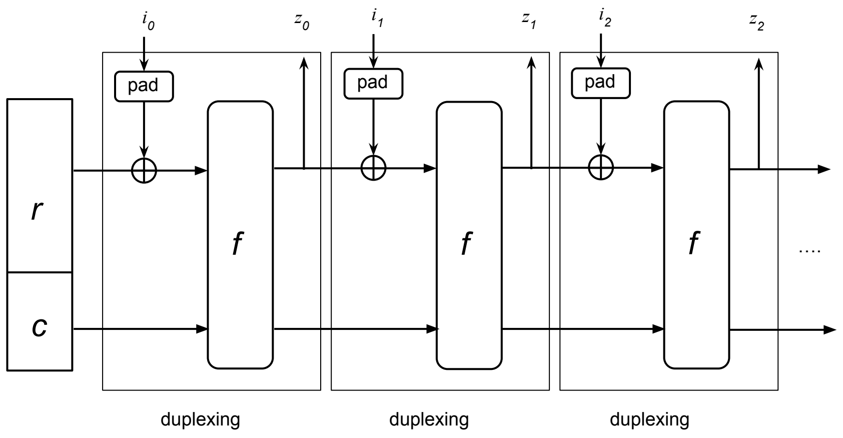
[2] Department of Computing, Macquarie University, Australia

Abstract. The authentication encryption (AE) scheme based on the duplex construction can no be paralellized at the algorithmic level. To be competitive with some block cipher based modes like OCB (Offset CodeBook) or GCM (Galois Counter Mode), a scheme should allow parallel processing. In this note we show how parallel AE can be realized within the framework provided by the duplex construction. The first variant, pointed by the duplex designers, is a tree-like structure. Then we simplify the scheme replacing the final node by the bitwise xor operation and show that such a scheme has the same security level.

1 Duplex construction

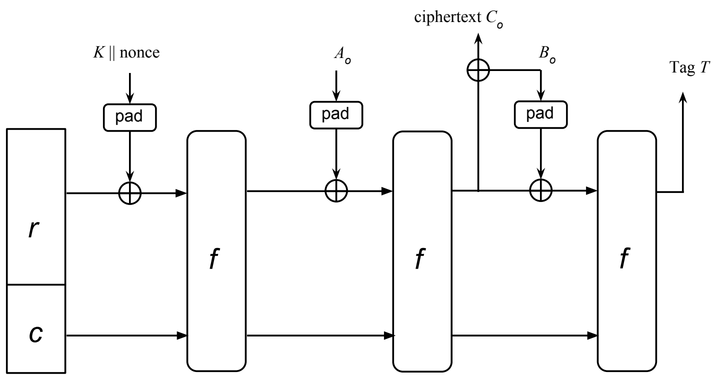
In 2010 Bertoni et al. introduced the duplex construction which provides the framework for an authenticated encryption scheme [3]. In this section we briefly discuss the construction with focus on the authenticated encryption. The duplex construction can be seen as a particular way to use the sponge construction [2], hence it inherits its security properties. The construction is based on the fixed permutation (or transformation) and allows the alternation of input and output blocks at the same rate as the sponge construction. Figure 1 shows the duplex construction.
Similarly as in the sponge construction, there are two parameters: r (bitrate) and c (capacity). The sum of those two parameters makes the state size. Different values for bitrate and capacity give trade-offs between speed and security. A higher bitrate gives a faster construction at the expense of a lower security. Upon initialization all the bits of the state are set to zero. The duplex construction accepts input calls (denoted by i_n in Figure 1) to the underlying permutation f. The padded input strings have the size of r bits. After a call to the permutation f, an output r-bit string is returned (denoted by z_n in Figure 1). Please note that the capacity part of the state is never directly manipulated by an input string i_n, nor is included in output strings z_n.
The authentication encryption scheme with associated data (AEAD) can be realized with the duplex construction. A secret key K, and message blocks B_i (optionally with associated data A_i) are processed as follows.

Fig. 1. Duplex construction



1. The key K (optionally concatenated with nonce) is absorbed.
2. r-bit block of associated data A_i is absorbed (and remains unencrypted). Then r-bit block of message B_i is absorbed and encrypted with the r-bit block squeezed from the current internal state.
3. A tag T is returned as the r-bit block squeezed after the last B block processed. (Empty blocks are processed if more than r bits are needed for tag T.)

Fig. 2. Authentication encryption scheme based on the duplex construction



We assume that all blocks are properly padded and frame bits are included. We refer to Algorithm 3 in [3] for a formal description.
Figure 2 illustrates a simple scenario where a pair of (A_0, B_0) is processed and the T is produced.

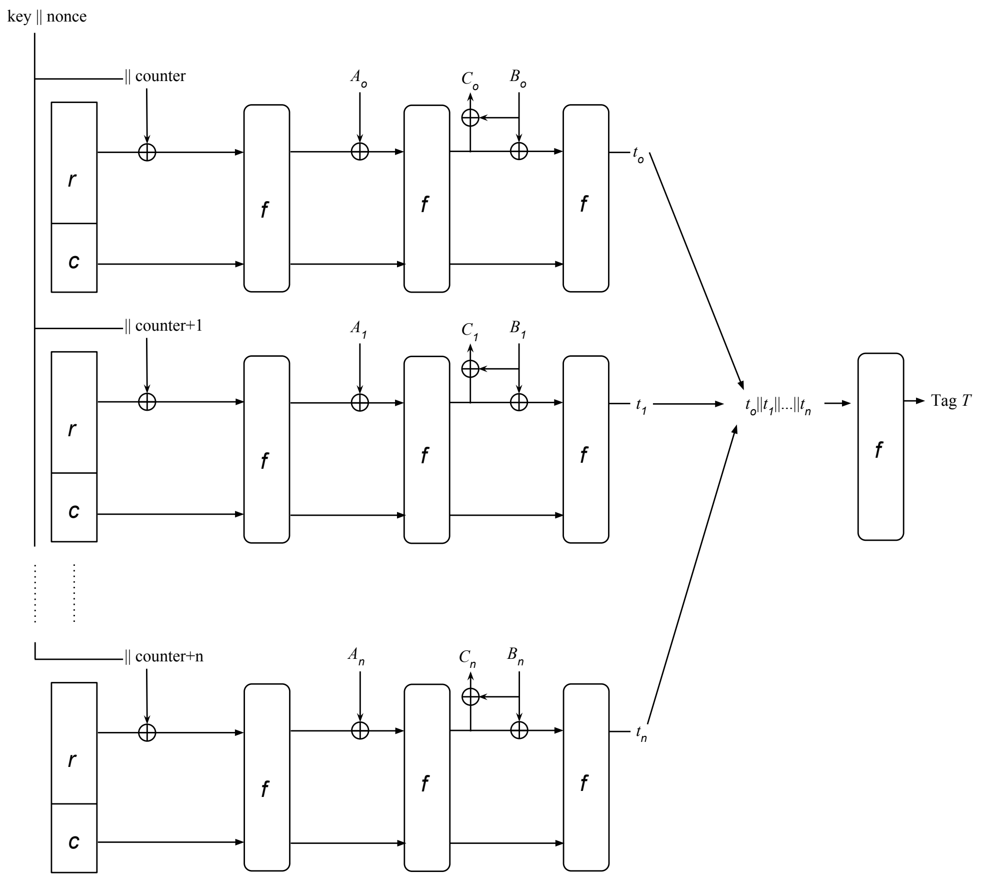
2 Parallel encryption with the duplex construction

To be competitive with some block cipher based modes like OCB (Offset CodeBook) or GCM (Galois Counter Mode), the AEAD scheme should allow parallel processing. However, the AEAD based on the duplex construction can no be paralellized at the algorithmic level. In [3] the authors just briefly state that tree-like processing (described in detail for hashing, see Section 3.3.2 in [2]) could be used also for the duplex-based AEAD providing parallel streams with some overhead. But no more details are given.

2.1 Parallel encryption with the final duplex node

First variant of parallel duplex-based AEAD is inspired by tree-hashing with the following parameters: tree height $H = 1$, node degree D equals to the number of 'streams'. It was defined in [2, Section 3.3] in terms of a sponge function as a compression function. Since the duplex construction has been proven equivalent to a cascade of sponge functions it inherits sponge security properties [3]. Consequently, tree-duplexing has the same security properties as a tree-hashing based on sponge functions. What is different though, is that a (parallel) authenticated encryption scheme has to provide confidentiality which is not required in (parallel) hashing.
Let us first describe a variant of parallel duplex-based AEAD, where beside a nonce we introduce a counter. Then we explain why a counter (or some equivalent mechanism) is needed to provide full confidentiality and prevent from leaking information about message blocks.
Message blocks $(A_0, B_0), (A_1, B_1), \ldots, (A_n, B_n)$ are processed in separate duplex streams and they meet in the final node to produce a tag for the whole message. An input to the first duplex (in each stream) is a secret key concatenated with a nonce and a counter. Then for each stream i the associated data A_i is processed followed by the message block B_i which is encrypted. In each stream, after the last call of permutation f, r bits are squeezed. They are concatenated and making an input for the final duplex node which generates the tag T. Figure 3 illustrates the scheme. For clarity we omit padding symbols in Figure 3 but we assume that every input block to the duplex is properly padded. We also assume that intermediate tags $t_0, t_1, \ldots, t_n$ has the same length as the final tag T.
To simplify security analysis we let an attacker know intermediate tags $t_0, t_1, \ldots, t_n$. This way we can analyze every stream separately. And it was already shown by Bertoni et al. [2, **?**] such a construction (a single stream) is secure provided that the underlying permutation f has no exploitable properties (structural distinguishers). Then the only question remains whether the final duplex call (producing the tag T) can be

Fig. 3. Parallel authentication encryption scheme based on the duplex construction



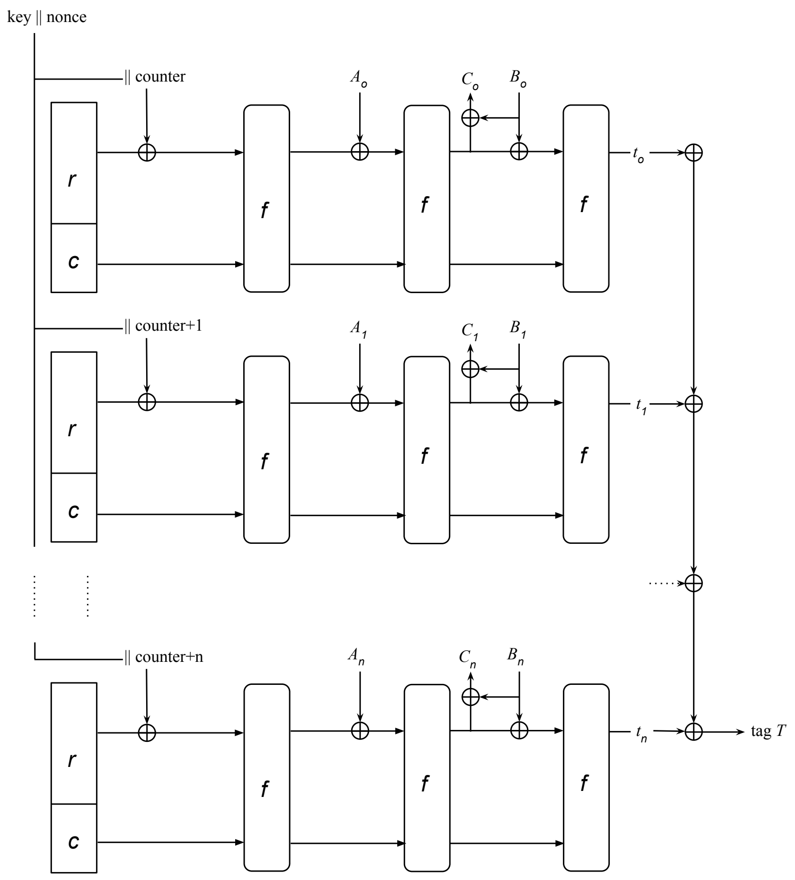
somehow exploited by the attacker. As $t_0, t_1, \ldots, t_n$ are already known, the key recovery attack (recovery of the state) starting from T does not make sense, it would only make the task harder. A forgery of T would mean that the attacker can somehow control $t_0, t_1, \ldots, t_n$, e.g., can find two messages giving the same t_0. But as a length of each of $t_0, t_1, \ldots, t_n$ is the same as T such a control would contradict the security of a given, single stream. Therefore we state that an addition of the final duplex call does not help the attacker and the whole scheme is secure.

We introduce the counter to maintain full confidentiality on the message, particularly on the relation between message blocks. If there were no counter (which differentiates each stream), the attacker could deduce some information on the message structure. For example, let B_0 equal to B_1 and let A_0 equal to A_1. Then the attacker having only ciphertexts C_0 and C_1 would deduce that message blocks B_0 and B_1 are the same.

2.2 Simpler scheme

The scheme can be simplified when the final duplex node is replaced by the XOR operation. Figure 4 shows such a scheme.

Fig. 4. Parallel authentication encryption scheme based on the duplex construction without the final duplex node



The security analysis is basically the same as for the previous scheme. The XOR of $t_0, t_1, \ldots, t_n$ does not help in the key recovery attack as

we give the attacker knowledge of $t_0, t_1, \ldots, t_n$ in advance. A tag forgery would mean that the attacker can control the xor differences between $t_0, t_1, \ldots, t_n$. This would contradict the security of a single stream. In particular we assume that the underlying f permutation has such differential properties that after the first call (with an input of key‖nonce‖counter), the attacker has no control of differences. Please note that exactly the same argument is used in [1] by Keccak designers where they analyze one of the duplex-based construction. They state:
"We rely on the initialization phase and its nonce to make differential attacks infeasible: an attacker has no control whatsoever over state differences between pairs of monkeyDuplex objects. This limits his attack path to state reconstruction."

3 Conclusion

We have shown how parallel AEAD can be realized within the framework provided by the duplex construction. We have simplified the tree-like scheme, suggested by the duplex designers, removing the final node and replacing it by the xor operation. In both discussed variants we highlight the importance of the counter (or some equivalent mechanism) for providing confidentiality.

References

1. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Permutation-based encryption, authentication and authenticated encryption (July 2012)
2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponges, http://sponge.noekeon.org/CSF-0.1.pdf
3. Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: Duplexing the sponge: single-pass authenticated encryption and other applications. Cryptology ePrint Archive, Report 2011/499 (2011), http://eprint.iacr.org/