Cryptanalysis of the SHA-3 candidates EnRUPT and SHAMATA
Extended Abstract

Sebastiaan Indesteege1,†,‡

†Dept. of Electrical Engineering ESAT/COSIC, K.U.Leuven, Belgium.
‡Interdisciplinary Institute for BroadBand Technology (IBBT), Belgium.
Email: sebastiaan.indesteege@esat.kuleuven.be
WWW: http://homes.esat.kuleuven.be/~sindeste/

Abstract

In this talk, we review the successful cryptanalysis of two cryptographic hash functions, EnRUPT and SHAMATA. Both were submitted as candidates to the NIST SHA-3 competition.

1. Introduction

Cryptographic hash functions play an important role in the security of many applications such as digital signatures and electronic commerce. Recent cryptanalytic advances have raised serious concerns regarding the long-term security of several popular hash functions such as MD5 and SHA-1. This has caused the National Institute of Standards and Technology (NIST) to organise a public competition, the SHA-3 competition [8, 9], aiming to develop and select the next generation hash function standard. Such a contest is similar to the process that lead to the selection of the Advanced Encryption Standard (AES) [4] in 2001.

Teams from all over the world have submitted new hash function designs to this competition. Of the 64 hash function proposals that were submitted to NIST, 51 have been accepted to the first round of evaluations. Recently, 14 second round candidates have been announced. Most of the current research on cryptographic hash functions is at least in some way related to this SHA-3 competition. Indeed, during the entire competition, a thorough evaluation of the security of all candidates is required in order to allow a good selection to be made.

In this talk, the successful cryptanalysis of two first-round SHA-3 candidates, EnRUPT [5] and SHAMATA [6], is summarised. It is shown that neither of these functions is collision resistant, a property which a secure cryptographic hash function is expected to offer. Informally,
collision resistance means that it should be hard to find two distinct messages \( m \neq m' \) that hash to the same value, i.e., \( h(m) = h(m') \). To find collisions for these two hash functions, coding theory turns out to be helpful to the cryptanalyst, as an important part of the analysis translates into the problem of finding low weight codewords in a linear code.

2. EnRUPT

The EnRUPT hash functions were proposed by O’Neil, Nohl and Henzen [10] The description of EnRUPT contains several parameters. Seven concrete EnRUPT variants are defined in the proposal. Each variant has a different hash output size, ranging from 128 bits to 512 bits. This section describes a collision attack on all seven EnRUPT variants, which was published in [5].

2.1. Basic Attack Strategy

We give an overview of the linearisation method for finding collision differential characteristics for a hash function, which we use to attack EnRUPT. This method was introduced by Chabaud and Joux [3], who applied it to SHA-0 and simplified variants thereof. Later, it was extended further and applied to SHA-1 by Rijmen and Oswald [12].

Consider a hypothetical hash function that consists only of linear operations over GF(2). When the input messages are restricted to a certain length, each output bit can be written as an affine function of the input bits. The difference in each output bit is given by a linear function of the differences in the input bits, as the constants (if any) cancel out. A message difference that leads to a collision can be found by equating the output differences to zero, and solving the resulting system of linear equations over GF(2), for instance using Gauss elimination. Any pair of messages with this difference will result in a collision.

Actual cryptographic hash functions contain (also) nonlinear components, so this method no longer applies. However, we may still be able to approximate the nonlinear components by linear ones and construct a linear approximation of the entire hash function. For our purpose, a good linear approximation \( \lambda(x) \) of a nonlinear function \( \gamma(x) \) is such that its differential behaviour is close to that of \( \gamma(x) \). More formally, the equation

\[
\gamma(x \oplus \Delta) \oplus \gamma(x) = \lambda(x \oplus \Delta) \oplus \lambda(x) = \lambda(\Delta)
\]

should hold for a relatively large fraction of values \( x \). For instance, an addition modulo \( 2^n \) could be approximated by a simple XOR operation, i.e., ignoring the carries.

A differential characteristic consists of a message difference and a list of the differences in all (relevant) intermediate values. For the linear approximation, it is easy to find a differential characteristic that leads to a collision with probability one. But for the actual hash function, this probability will be (much) lower. If the differential behaviour of all the nonlinear components corresponds to that of the linear approximations they were replaced with, i.e., if (1) holds simultaneously for each nonlinear component, we say that the differential characteristic is followed. In this case, the message pair under consideration will not only collide for the linearised hash
function, but also for the original, nonlinear hash function. Such a message pair is called a conforming message pair.

Hence, a procedure for finding a collision for the nonlinear hash function could be to find a differential characteristic leading to collisions for a linearised variant of the hash function. Then, a message pair conforming to the differential characteristic is searched. In order to lower the complexity of the attack, it is important to maximise the probability that the differential characteristic is followed, i.e., we need to find a good differential characteristic.

2.2. Linearising EnRUPT

We now apply this general strategy to EnRUPT. From the description of EnRUPT [10], it can readily be seen that the only operation in EnRUPT which is not linear over GF(2) is a modular addition in each round. Replacing these modular additions with XOR operations yields a linearised EnRUPT hash function, that consists solely of GF(2)-linear components.

During the collision search phase, many collisions for this linearised variant of EnRUPT are constructed, and a collision for the actual EnRUPT is searched among them. Since only the modular additions were approximated by XOR, these are the only places where the propagation of differences could differ between EnRUPT and its linearised variant. Instead of checking for a collision at the output, we can immediately check if the difference at the output of each modular addition still matches the differential characteristic.

2.3. Finding good characteristics

The key to lowering the attack complexity is to find a good differential characteristic, i.e., a characteristic which is likely to be followed for the nonlinear hash function. A generic approach to this problem, based on finding low weight codewords in a linear code, was proposed by Rijmen and Oswald [12] and extended by Pramstaller et al. in [11]. As observed by Rijmen and Oswald [12], all of the differential characteristics for the linearised hash function can be seen as the codewords of a linear code.

Each codeword contains a message difference, the input differences to all approximated modular additions, and finally the output difference. Each codeword is then a differential characteristic for the linearised variant of EnRUPT, and all such differential characteristics are codewords of this code. To restrict ourselves to collision differentials, i.e., differential characteristics ending in a zero output difference, we use Gauss elimination to force the columns of the generator matrix corresponding to the output difference to zero.

The reason for including the input differences to the modular additions in the codewords is that this information allows to compute the probability that the approximation is followed. It is well known that the differential behaviour of modular addition can be well approximated by that of XOR when the Hamming weight of the input difference, ignoring the most significant bit, is small [3, 7, 11, 12]. Thus, we will attempt to find a codeword with a low Hamming weight in this part of the codeword.
2.4. Low Weight Codewords

To find low weight codewords, we used a simple and straightforward algorithm that is based on the assumption that a codeword of very low weight exists in the code. For our purposes, this is a reasonable assumption, as only a very low weight codeword will lead to an attack faster than a generic attack. The algorithm is related to the algorithm of Canteaut and Chabaud [2] and the algorithm used to find low weight codewords for linearised SHA-1 by Pramstaller et al. [11].

Let $G$ be the generator matrix of the linear code. We randomly select a set $I$ of (appropriate) columns of the generator matrix $G$ and force them to zero using Gauss elimination, until only $d$ rows remain, where $d$ is a parameter of the algorithm. Then, the remaining space of $2^d$ codewords is searched exhaustively. This procedure is repeated until a codeword of sufficiently low weight is encountered. By replacing only the ‘oldest’ column(s) in $I$, instead of restarting from the beginning every time, the algorithm can be implemented efficiently in practice.

If a codeword of very low weight exists in the code, it is likely that all of the columns in the randomly constructed set $I$ will coincide with zeroes in the codeword, which implies that the codeword will be found in the exhaustive search phase. In the case of the codes originating from the seven linearised EnRUPT variants we consider, this algorithm finds a codeword of very low weight in a matter of minutes on a PC. Repeated runs of the algorithm always find the same codewords, so it is reasonable to assume that these are indeed the best codewords we can find.

Actually, the weight of a codeword is only a heuristic for the attack complexity resulting from the corresponding differential. Codewords with a lower weight are expected to result in a lower attack complexity, but we can easily enhance our algorithm to optimise the actual attack complexity, rather than just a crude heuristic.

2.5. Results

We have found good differential characteristics for each of the seven EnRUPT variants. The associated attack complexities range from $2^{36}$ to $2^{40}$ EnRUPT round computations, depending on the EnRUPT variant, which is significantly less than the $2^{n/2}$ required for a generic collision attack on an $n$-bit hash function. For details, we refer to [5], which also shows an example collision pair for EnRUPT-256.

3. SHAMATA

SHAMATA is a register-based hash function design that was proposed by Atalay et al. [1]. SHAMATA uses several components of the AES [4] block cipher internally. The collision attack described in this section is a joint work with Florian Mendel, Bart Preneel and Martin Schläffer [6].
3.1. Basic Attack Strategy

SHAMATA has an internal state consisting of a register of four 128-bit words and a register of twelve 128-bit words. For each 128-bit message block, the state is updated as follows. First, linearly transformed copies of the message block are XORed to six register words. Then, the internal state is clocked twice, i.e., both registers shift and there is a feedback from each register into the other.

This entire update procedure is linear over GF(2), except for a single AES round function in each clocking. The basic idea behind the attack is the observation that the all-ones difference \( \Delta \) is left unchanged by all operations in SHAMATA, except possibly the AES round. However, for specific values, also the AES round preserves the difference \( \Delta \).

If the AES round is active in only one of the two clockings, the desired differential transition can be forced easily by computing backwards to find an appropriate value for the message block. If both clockings are active, however, this is no longer possible due to a lack of message freedom. Hence, we aim to find a differential path that uses only \( \Delta \) differences, activates the AES round in at most one clocking of each round, and results in a full internal state collision.

3.2. Coding Theory

As we consider only \( \Delta \) differences, the difference in each 128-bit register can be represented by a single bit: either there is no difference, or there is a \( \Delta \) difference. If we assume that all AES rounds pass a \( \Delta \) difference unchanged, all of SHAMATA becomes linear. Thus, we can again construct a linear code where each codeword corresponds to a differential characteristic. To reach an internal state condition, the same approach is taken as for EnRUPT in Sect. 2.3.

Now we only have to find a codeword which satisfies the condition that at most one of the two clockings may be activated in each round. Intuitively, a codeword with a low weight in the relevant positions, is more likely to satisfy this property than a random codeword. Thus, we search for low-weight codewords of this code, considering only the weight of the relevant positions, until a satisfactory codeword is found.

3.3. Results

The shortest differential path that we found consists of 25 rounds. In all but the first round, at most one clocking is active. Due to the construction of SHAMATA, it is impossible to avoid activating both clockings in the first round. However, as is explained in [6], this problem can be overcome. Note that this path could also have been found using a simple brute force search, as it is very short. However, the coding theory approach scales much better, and it was not known a priori that such a short path would exist. This path results in a practical collision attack on SHAMATA-256, with a time complexity of \( 2^{40} \), and a theoretical collision attack on SHAMATA-512. For details, we refer to [6].
Acknowledgements

This work was supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

References


