Cryptanalysis of block EnRUPT

Elias Yarrkov*

2010-10-08 (revised 2010-10-12)

Abstract

EnRUPT is a cryptographic primitive with a variable block and key length. We show several attacks on it that stem from properties of its keying, including a very fast related-key attack.

1 Introduction

EnRUPT[5] is a simple symmetric algorithm with several modes, allowing it to perform as different types of cryptographic primitives. One mode of EnRUPT was previously attacked in [3] and [2]. This paper presents cryptanalysis of the 32-bit block cipher mode, though the same methods are expected to work for the 64-bit version.

2 EnRUPT description

EnRUPT is an unbalanced source-heavy Feistel network similar to XXTEA[7, 6]. The block consists of a variable number of 32- or 64-bit words, as does the key. The algorithm linearly loops through block and key words, XORing to the current block word a function of its neighbors, the current key word, and the round number. The block and the key are viewed as circular arrays; that is, for an n-word block and key, $x_r = x_{r+n}$ and $k_r = k_{r+n}$. The EnRUPT round function is $x_r \leftarrow x_r \oplus F(x_{r-1}, x_{r+1}, k_r, r)$, where $F(a, b, k, r) = ((((a \ll 1) \oplus b \oplus r \oplus k) \ggg 8) * 9) \oplus k$. The number of rounds is defined $s * (w_0 * 2 + w_1)$, where w_0 is block width in words and w_1 is key width in words. s is a security parameter, defaulted to 4.

*yarrkov@gmail.com, http://cipherdev.org/

Algorithm 1 EnRUPT block cipher implementation[1]

```
#define rotr(a,b) (((a)>>(b))|((a)<<(32-(b))))
void EnRUPT(uint32_t *x, int xw, uint32_t *k, int kw)
{
        int r;
        for (r=1; r<=4*(2*xw+kw); r++)
        {
                uint32_t t, rk;
                t = (x[(r-1+xw)%xw] << 1) ^ x[(r+1)%xw];
                rk = k[r%kw];
                x[r%xw] ^= rotr(t ^ rk ^ r, 8)*9 ^ rk;
        }
}
```

3 Attacks

The approaches used for XXTEA in [8] seem to be ineffective against EnRUPT. However, the keying is vulnerable to various things.

3.1 Key collisions

The round function of EnRUPT is not bijective for the given key word. Thus, we can attempt to find a difference Δ so that $F(a, b, k \oplus \Delta, r) = F(a, b, k, r)$ with some probability. Using a simple solving algorithm, $\Delta = 0x02128292$ comes out as probably the best choice, giving a collision at probability $2^{-9.15}$. If two keys differ in a single word, the difference will be encountered only once per cycle through the key words.

For example, for a 128-bit block and 512-bit key, the number of full cycles over the key is 6; we need to pass 5 to get a right pair. In this scenario, the probability of the full characteristic is thus about $2^{-45.75}$. We can request about 2^{46} encryptions under the original key, as well as a related key with Δ in k_0. This is expected to result in about one case where the direct ciphertext and its respective related-key ciphertext collide in x_1, x_2 and x_3, which is recognized as a right pair. Knowing $(x_3 \ll 1) \oplus x_1$ and the difference in x_0, we can proceed to recover candidates for k_0, used for x_0. Though this immediately gives an advantage over brute force, effectively extending to the full key seems to require a bit more queries with Δ in different key words.

These key collisions also lead to hash collisions, if the block cipher is used in a construction such as Davies-Meyer.

[1] derived from a public domain implementation at http://www.enrupt.com/

3.2 A p=1 related-key differential characteristic

Consider the round function, $((((x_{r-1} \ll 1) \oplus x_{r+1} \oplus r \oplus k) \ggg 8) * 9) \oplus k$. Observe that we can fully cancel the difference $0x80000080$ in $(x_{r-1} \ll 1) \oplus x_{r+1}$ with the difference $0x80000000$ in k. We can achieve the appropriate difference in $(x_{r-1} \ll 1) \oplus x_{r+1}$ with the difference $0x7FFFFF80$ in x_{r-1} and x_{r+1}. This immediately leads to a distinguisher. Specifically, we can use that difference in every block word, and the appropriate key difference in every key word; the ciphertext block difference will be equal to the plaintext block difference.

3.3 A chosen-plaintext attack

The previously mentioned property can be adapted into a simple chosen-plaintext attack similar to that for DES[4] due to its complementation property[1]. For this attack description, the block length will be considered equal to the key length.

We define and alternative form to view the block in: $o_r = (x_{r-1} \ll 1) \oplus x_{r+1}$ for r in $[0, w-1]$, where $w = w_0 = w_1$. Notice that o is a bijection of x. We can set a difference either 0 or $0x80000080$ in each word in o, giving 2^w possible differences in total. We can linearly solve those values back to the form used in x, and use the key difference $0x80000000$ where the matching o difference is $0x80000080$. Consider Δ_a all of the different block differences and ∇_a the respective key differences, with a in $[0, 2^w - 1]$.

To turn this into a chosen-plaintext attack, we first do 2^w chosen-plaintext queries to get the ciphertext $E_k(\Delta_a)$ for each of Δ_a, and we set the key $E_k(\Delta_a) \oplus \Delta_a$ to point to value a in a fast lookup table. Notice that

$$\begin{aligned} E_k(0) \oplus E_{k \oplus \nabla_a}(\Delta_a) &= \Delta_a \\ E_{k \oplus \nabla_a}(0) \oplus E_k(\Delta_a) &= \Delta_a \\ E_{k \oplus \nabla_a}(0) &= E_k(\Delta_a) \oplus \Delta_a \end{aligned}$$

We can use this to accelerate exhaustive search. We encrypt a zero-block with all possible keys, except we always leave the highest bit of each key word zero. After each block encryption with trial key k_t, we test for a match in the lookup table. If the ciphertext is found to point to a value a in the table, then probably $k = k_t \oplus \nabla_a$. As an optimization to reduce memory requirements, we don't need to store $E_k(\Delta_a) \oplus \Delta_a$ entirely, only enough to keep the rate of false positives low. In total, the work required for exhaustive search is reduced by at most w bits with 2^w chosen-plaintext queries and memory proportional to storing 2^w blocks. Generalizing to cases where the block length does not equal the key length, the reduction in work is $gcd(w_0, w_1)$ bits.

Simplistically, it could be said that the work*memory complexity doesn't actually change from exhaustive search. However, the cost of a single EnRUPT circuit (including the block) is larger than the cost of just storing a block or part of it, so this is expected to give a cheaper attack than plain exhaustive search.

3.4 Fast key recovery

The previously shown family of characteristics can't be directly used to recover the key, but we can weaken it to get key dependency, thus allowing efficient key recovery. The differences $\Delta o_r = 0x40000040$ and $\Delta k_r = 0x40000000$ cancel out with probability about $2^{-0.85}$. We can set the difference $0x40000040$ in one word in the o-form, and 0 or $0x80000080$ in the others. For example, for eight block and key words:

$\Delta x = \{0x4CCCCC80,\ 0,\ 0x19999980,\ 0,\ 0x73333340,\ 0,\ 0x66666600,\ 0\}$
$\Delta k = \{0,\ 0x80000000,\ 0,\ 0x40000000,\ 0,\ 0x80000000,\ 0,\ 0x80000000\}$

When the number of key words equals the number of block words, the number of cycles to perform over the block is 12, as in this example. Naively, we'd then calculate the right pair probability as $2^{-0.85*11}$. However, the probability of the characteristic failing at the appropriate place is $2^{-1.17}$, which is not negligible. In addition, when the characteristic does break, the output difference from F will be $0x80000000$; this will not spread further during the same cycle, because the next round's input from the current word will be shifted up by one. Thus, the probability is $2^{-0.85*10-1.17}$. This results in a requirement of about $2^{10.67}$ queries, under two keys in total. When we have a right pair, all input to the final call of F, except the used key word, is visible. The difference in the output is also visible. This leads to straightforward recovery of candidate key words.

More than one right pair will help key recovery. A test implementation recovers the full key in minutes with about four right pairs per key word. In this scenario, that means about $2^{15.67}$ queries. This could be further improved.

4 Conclusions

Flaws were shown in the block cipher keying of EnRUPT that lead to several attacks. Problems stemming from one property are strong enough to allow easily experimentally performing full key recovery. EnRUPT is thus not a secure family of pseudorandom permutations, as an optimal block cipher would be.

References

[1] M. E. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, and P. Schweitzer. Results of an initial attempt to cryptanalyze the NBS data encryption standard. Technical Report SEL 76–042, Stanford University, 1976.

[2] Sebastiaan Indesteege and Bart Preneel. Practical collisions for enrupt. pages 246–259, 2009.

[3] Dmitry Khovratovich and Ivica Nikolic. Cryptanalysis of enrupt. Cryptology ePrint Archive, Report 2008/467, 2008. http://eprint.iacr.org/.

[4] Sara Kiesler and Jennifer Goetz. National bureau of standards: Data encryption standard. Technical report, 1977.

[5] Sean O'Neil. Enrupt: First all-in-one symmetric cryptographic primitive. The State of the Art of Stream Ciphers (SASC), 2008.

[6] D J Wheeler and R J Needham. Correction of xtea. unpublished manuscript. In *Computer Laboratory, Cambridge University*, 1998.

[7] David J Wheeler and Robert M Needham. Tea extensions. Technical report, 1997.

[8] Elias Yarrkov. Cryptanalysis of xxtea. Cryptology ePrint Archive, Report 2010/254, 2010. http://eprint.iacr.org/.