Specjal Block Cipher

Karl Zander – pvial00@gmail.com

Abstract.

Specjal is an encryption algorithm that obfuscates blocks of data so that no one is able to read it without the key. It is an ARX cipher using only addition mod 2^64, bitwise rotations, the XOR operator and also swap operations. It has a 256 bit block size for all key lengths.

1. **Introduction**

Specjal is a fast block cipher operating on 256 bits of data at a time.

Specjal consists of a key scheduler and a round function.

The cipher is designed to operate at key lengths from 128 bits to 1024 bits. An initialization vector is required for encryption. The length of the IV must be 256-bits to match the block size (could be be different depending on the mode in implementation).

2. **Design goals**

The algorithm must adhere to the strict avalanche effect and produce proper confusion and diffusion and also introduce word transposition.

The algorithm must employ an initialization vector that does not effect the key generation process.

The algorithm must produce a non-repeating stream (without period) of bytes with uniform characteristics.

The algorithm must pass known statistical testing for random number generators. The algorithm must be fast to implement in software.

3. **Round Key Scheduling**

There are 512 bits of key material per round applied to the block. There are four 64 bit primary round keys and four 64 bit diffusion keys.

The key scheduler is fairly simple. It loads the key into a 64 bit word array k[], creates a temporary copy of it's key state, then loads the key k[] into that state. The key array is then run through Specjal's round function to produce a new key each time it's run that is loaded into the primary key state.

This is done in the following order, Ka[] is generated, Kb[] is generated and finally the diffusion keys are generated.

5. **Round Encryption/Decryption**

Only ARX operations are used during the round. The cipher operates on 256 bits of data at a time split up into four 64 bit words. The words are subjected one of many functions such as: being bitwise rotated left or right so many positions, added to other words modulo 2^64, XOR'ed with a primary round key, XOR'ed with another word and being added to the diffusion key.

In code and in this example of one round of encryption we'll refer to those four words as A, B, C, and D in that order. Encryption is done in 3 phases.

Encryption:

1. ARX phase
    - First, B is added to A, then A is bitwise rotated left 9 positions and XOR'ed with a primary round key.
    - Second, A is added to B, B is rotated 14 positions left and XOR'ed with a primary round key
    - Third, D is added to C, C is rotated 17 positions left and XOR'ed with a primary round key
    - Fourth, C is added to D, D is rotated left 36 positions and XOR'ed with a primary round key

2. Diffusion key phase
    - The four diffusion keys are added to A, B, C and D.

3. Transposition phase

In this phase the inner two words are swapped and the outer two words are swapped creating a specific word transposition order. This ensures that all words are evenly mixed. The following shows the order of swaps until returning to the original order.

Transposition swap order:

0 1 2 3

3 0 1 2

2 3 0 1

1 2 3 0

0 1 2 3

Decryption:

1. Transposition phase
   - The words are subjected to the above word transposition swapping the inner two words and then the outer two words

2. Diffusion key phase
   - The four diffusion keys are subtracted from A, B, C and D.
3. ARX phase
   - First, D is XOR'ed with a primary key from array Kd[], then D is rotated right 36 positions and then C is subtracted from D.
   - Second, C is XOR'ed with a primary key from array Kc[] then C is rotated right 17 positions and D is subtracted from C.
   - Third, B is XOR'ed with a primary key from array Kb[], then B is rotated right 14 positions and A is subtracted from B.
   - Fourth, A is XOR'ed with a primary key from array Ka[]. Then A is rotated right 9 positions. Finally, B is subtracted from A.

The following code demonstrates both the encryption and decryption functions:

Encryption:

```
void SroundF(struct specjal_state *state, uint64_t *xla, uint64_t *xlb, uint64_t *xra, uint64_t *xrb, int rounds) {
   uint64_t a, b, c, d, temp;
   a = *xla;
   b = *xlb;
   c = *xra;
   d = *xrb;
   for (int r = 0; r < rounds; r++) {
      a += b;
      a = specjal_rotl(a, 9);
```

```
        a ^= state->Ka[r];

        b += a;
        b = specjal_rotl(b, 14);
        b ^= state->Kb[r];

        c += d;
        c = specjal_rotl(c, 17);
        c ^= state->Kc[r];

        d += c;
        d = specjal_rotl(d, 36);
        d ^= state->Kd[r];

        a += state->d[r][0];
        b += state->d[r][1];
        c += state->d[r][2];
        d += state->d[r][3];

        temp = b;
        b = c;
        c = temp;
        temp = a;
        a = d;
        d = temp;

    }
    *xla = a;
    *xlb = b;
    *xra = c;
    *xrb = d;
}
```

Decryption:

```
void SroundB(struct specjal_state *state, uint64_t *xla, uint64_t *xlb, uint64_t *xra, uint64_t *xrb, int rounds) {
    uint64_t a, b, c, d, temp;
    a = *xla;
    b = *xlb;
    c = *xra;
    d = *xrb;
    for (int r = rounds; r --> 0;) {

        temp = b;
        b = c;
        c = temp;
        temp = a;
        a = d;
        d = temp;

        d -= state->d[r][3];
        c -= state->d[r][2];
        b -= state->d[r][1];
        a -= state->d[r][0];
```

```
        d ^= state->Kd[r];
        d = specjal_rotr(d, 36);
        d -= c;

        c ^= state->Kc[r];
        c = specjal_rotr(c, 17);
        c -= d;

        b ^= state->Kb[r];
        b = specjal_rotr(b, 14);
        b -= a;

        a ^= state->Ka[r];
        a = specjal_rotr(a, 9);
        a -= b;

    }
    *xla = a;
    *xlb = b;
    *xra = c;
    *xrb = d;
}
```

6. **Cryptanalysis**

TBD

7. **Statistical Properties**

Specjal was subjected to the Diehard battery of tests (dieharder). Overall, Specjal passed the tests. Tests were conducted on streams of 1 gigabyte of data with 100 different key, nonce pairs. Specjal was also subjected to NIST Statistical Test Suite battery of tests and passed.