The Cryptographic Software Tool CipherCAD and Cryptanalysis

Vlastimil Klima(*)
Vaclav Platenka(**)

(*) vlastimil.klima@knzsro.cz
KNZ, Ltd.
Prague, Czech Republic

(**) vaclav.platenka@unob.cz
University of Defence
Brno, Czech Republic

Abstract

CipherCAD is a graphical programming tool that can be used to model and explore cryptographic functions, protocols, etc. In this paper we present an application of the CipherCAD to model and explore five finalists of NIST SHA-3 competition. In this limited space we show some of the myriad possibilities of using CipherCAD for cryptanalysis of cryptographic functions and their evaluation. We also present cryptographic designing tools and options on the example of Skein-512 hash function. Furthermore, a comparative analysis is shown on the example of an avalanche test for all five finalists SHA-3: BLAKE, Grøstl, JH, KECCAK and Skein.

Keywords: CipherCAD, SHA-3, BLAKE, Grøstl, JH, KECCAK, Skein.

1 Introduction

CipherCAD was created within the research and development for the Czech National Security Authority [1]. In CipherCAD, all the studied cryptographic functions and protocols are assembled from building blocks that can be nested, concatenated or connected in any way. CipherCAD makes it possible to look at the data flow, to save them, or to perform other tests with them.

CipherCAD is a software application that is designed for the analysis and synthesis of cryptographic algorithms and protocols. The functional models of all currently used symmetric and asymmetric cryptographic algorithms can be implemented in CipherCAD. CipherCAD also contains the analytical tools for providing statistical and algebraic tests of cryptographic algorithms and their building blocks. More than one hundred cryptographic algorithms, analytical procedures and standardized protocols were processed using this application. All of them are available in the "Book of Schemes" or as separate "Notebooks" that are accessible from the front page of the application. In addition, the application supports the creation of documentation of the cryptographic algorithms and protocols.

For the creation of mathematical models, CipherCAD provides an interactive graphical interface which corresponds to the current level of CAD (Computer Aided Design) applications. The user creates functional models of cryptographic elements almost exclusively by graphical means using chosen components. From any component or a scheme, the application can generate required data to the algebraic and statistical analysis.
2 The Model of The Skein Algorithm in CipherCAD

For the demonstration of how to create a model of a cryptographic function, we chose the Skein 512-512 hash algorithm. Its core is a tweakable Threefish-512 block cipher with a 512-bit key and a 128-bit tweak. Threefish uses XOR, ADD and ROT operations (by a constant number of bits) with 64-bit words. These all form the basic nonlinear part of Threefish called MIX, see Fig. 1.

MIX is assembled from the basic components using the context menu and connections.

![Figure 1: The logical structure MIX is created by connections and components.](image)

Once having the structure, we can “encapsulate” it into the so-called "iterator". Thus the new single functional block is created. We can also “pack” the iterator to create a new small (graphical) block that will be labelled "MIX". Therefore, MIX is a new function, that has two 64-bit inputs and two 64-bit outputs - see Fig. 2 on the right.

![Figure 2: Iterator, encapsulation and creation of a new MIX function.](image)
The Threefish-512 block cipher works in 72 rounds; each of these rounds consists of four MIX functions and one permutation of eight 64-bit words, see Fig. 3.

![Figure 3: Four of the 72 rounds of the Threelfish-512 block cipher.](image)

As it is shown in the figures 3 and 4, after every four rounds, the appropriate subkey is added to the data.

![Figure 4: Inserting the key in Threelfish-512.](image)

Fig. 5, depicts the whole Threelfish-512 algorithm with additional final xor of plaintext. The main part of Threelfish-512 is an iterator consisting of 8 rounds. It is called 9 times, which gives (a total of) 72 Threelfish rounds.

An output from one iteration is led to the input of the next iteration using a memory component called „D feedback“. 
Treefish-512 with the final xor of the plaintext is the basic core of the UBI block, as it is depicted in the Fig. 6.

UBI processes many input message blocks simply by calling the UBI core many times, as it is seen in the Fig. 6. The result of the UBI core is used as a key in the following UBI core and the next message block is used as an input of the UBI core. By such chaining, a message of arbitrary length is processed by UBI.

Figure 5: One UBI block (Treefish-512 with final xor of plaintext).
The example of Skein-512 computation for 166-byte input is shown in the Fig. 6.

![Figure 6: Hashing a three-block message using UBI mode.](image)

The UBI block can also be packed into a single building element. In the Fig. 7, we can see it together with the input and output values of UBI.

![Figure 7: Display of UBI intermediate values when hashing the 4-bit "1000" message](image)
3 Cryptanalysis in CipherCAD

Once the algorithm model is created, we can use CipherCAD to perform various investigations of the algorithm. One of the basic tests of block ciphers is the avalanche test, which tests if any change of one bit of the input changes each output bit with the 0.5 probability. In order to test the influence of every single input bit of the block cipher, we go with the changing bit continuously from the least to the most significant position of the input block (walking bit method). For each input bit position, we compute how many bits of the output block have been changed. Then we display the result in the graph where the x-axis represents the input position and the y-axis represents the number of bit changes in the output. In the Fig. 8, there is an avalanche test for the whole Threefis-512 block cipher with 72 rounds. The results show that there are no significant deviations from the expected parameters and the histogram shows that around half of the output bits always change.

Figure 8: The avalanche test with the walking bit method of Threefish-512 block cipher.

On the other hand, Fig. 9 displays a walking bit test of one round of the Threefish-512 algorithm; that means the first key injection, four MIX functions and one permutation of eight 64-bit words was implemented. For better understanding of the influence of a particular bit, the whole test is repeated 1,000 times with a different random value of the input block and the results are averaged. From the graph in Fig. 9, we can see the influence of nonlinear structure of MIX function to the diffusion of input differential. There is a dependence on whether the input difference is in the left or in the right input word of MIX function. In the Fig. 9, the left circuit finds minimum and maximum number of changes. It can be seen that the minimum average Hamming weight of the output difference is 2 and the maximum one is 6.259. These very low numbers result from the fact that the MIX function is composed of three elements XOR, ADD and ROT, while a change in the most significant bit of ADD does not bring the difference to the neighboring bits.
3.1 4.1 Near-Pseudo-Collisions of Skein-512

In [4], sec. 9.2, near-pseudo-collision for eight rounds of Skein-512 compression function is described. This attack uses the property of ADD described above. Let δ denote the single-bit difference of the most significant bit of a 64-bit word \( \delta = 1000\ldots0 \). If we have the key difference = \((0, 0, 0, 0, 0, 0, \delta, \delta)\) and the tweak difference = \((0, \delta)\) then for Skein-512 we get the difference of the first subkey before the first round equal to \((0, 0, 0, 0, 0, 0, 0, 0, \delta)\), the difference of second subkey after the fourth round equal to \((0, 0, 0, 0, 0, 0, 0, \delta)\) and the difference of third subkey after the eighth round equal to \((0, 0, 0, 0, 0, 0, 0, 0, \delta)\). When we enter the difference \((0, 0, 0, 0, 0, 0, 0, \delta)\) in the input message, then after 8 rounds we get Threefish-512 difference with Hamming weight = 1. Then, after the application of chaining mode, we obtain corresponding near-pseudo-collision with Hamming weight 2, see Fig. 10. This holds for any values of the key, the tweaks and the messages. The output difference after eight Threefish-512 rounds corresponds to the difference of third subkey \((0, 0, 0, \delta, 0, 0, 0)\) and the output difference of Skein-512 compression function after eight rounds is \((0, 0, 0, 0, 0, \delta)\).
We wondered how this regularity behaves when we increase the number of rounds from 8 to 20. Thus we compiled a simple CipherCAD scheme, as depicted in Fig. 11. It preserves the differences in the messages, keys and tweaks described above, but their values are generated randomly (50,000,000 times). We have shown that the number of dependent bits on the output is equal to 512 for 8 rounds; in other words, the output difference holds accurately. For higher number of rounds, this rule gets less precise, the number of bits with the precise reaction decreases and the probability of a change is closer and closer to 0.5.

Fig. 11 shows the number of dependent bits (with a significant deviation) at the output of 9th to 20th round of Threefish-512. The whole number of the observed samples is 50,000,000. When we have dual-core processor, we can run two independent schemes, each of them for 25,000,000 random input samples. Thus we save the computational time. After calculation of these two parts (iterators), we run the remaining part of the scheme on the bottom. It computes the statistics and displays them in the table. The table confirms the results obtained in [4].
Figure 11: Observations of the influence of special differences for random messages, keys and tweaks in Threefish-512.

3.2 4.2 Avalanche Test of SHA-3 Finalists

To have a comparison of all five SHA-3 candidates, we will now show the results of the same avalanche test for the other four SHA-3 finalists. They are in figures 12 - 15. In the figures, we used color distinguishing: the red color indicates the results for one round, the blue color for two rounds, the black color for three rounds and the green color for four rounds.

In Fig. 12 (13, 14, 15) the walking bit method shows a one-bit input difference, which has the smallest impact on the output of the compression function for small number of rounds for a given algorithm, similarly as for Skein-512.

The chosen difference was then used in the following test, results of which are quoted in the table 1 (2, 3, 4).
Table 1: A bias of the function "Rounds" (BLAKE-512) for 50,000,000 input pairs.

<table>
<thead>
<tr>
<th>r</th>
<th># bits with bias</th>
<th>average bias</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>full &gt; 0.1 &gt; 0.01 &gt; 0.001</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>920 1020 1020 1020</td>
<td>0.49511745195312</td>
</tr>
<tr>
<td>2</td>
<td>0 154 450 598</td>
<td>0.0420845248283125</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 0</td>
<td>5.772469957000012-05</td>
</tr>
</tbody>
</table>

Table 2: A bias of the compression function f (Grøstl-512) for 50,000,000 input pairs.

<table>
<thead>
<tr>
<th>r</th>
<th># bits with bias</th>
<th>average bias</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>full &gt; 0.1 &gt; 0.01 &gt; 0.001</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1024 1024 1024 1024</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>1024 1024 1024 1024</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>64 64 64 64</td>
<td>0.03130697734375</td>
</tr>
<tr>
<td>4</td>
<td>0 0 0 0</td>
<td>5.244511716749999E-05</td>
</tr>
</tbody>
</table>

Figure 12: Walking bit method for the first and the second round of the "Rounds" function of BLAKE.

Figure 13: Walking bit method for the first to the third round of the compression function f of Grøstl-512.

Figure 14: Walking bit method for the first to the fourth round of R8 function of JH algorithm.
Table 3: A bias of R8 function (JH-512) for 50,000,000 input pairs.

Figure 15: Walking bit method for the first to the third round of $f[1600](A)$ function of KECCAK algorithm.

Table 4: A bias of $f[1600](A)$ function (KECCAK-512) for 50,000,000 input pairs.

Now we have achieved the results for all five algorithms for the subsequent comparison. To do that, we will simply find the minimal number of rounds of each algorithm, when visible dependencies among output and input bits can no longer be discerned.

We are specifically looking for a minimal number of rounds to achieve zero number of bits with bias exceeding 0.1%. The results are summarized in the Table 5. Since the algorithms have different numbers of rounds, we also calculated a ratio of the "number of dangerous rounds" to the "total number of rounds". The bigger the ratio, the more conservative the hash function is.

Although we performed only one test examining an avalanche property, we got a certain idea about the safety of candidates SHA-3. On the basis of that, we can’t say that Skein is the safest of all the candidates, but on contrary, we see that the differences in our safety measures are not fundamental for individual candidates, see Tab. 5.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Tested function</th>
<th>Total rounds</th>
<th>Minimum number of rounds</th>
<th>Total/Min.</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLAKE-512</td>
<td>Rounds</td>
<td>16</td>
<td>3</td>
<td>5,3</td>
<td>from 14 to 16 rounds tweak of BLAKE for Round 3</td>
</tr>
<tr>
<td>Grøstl-512</td>
<td>f</td>
<td>14</td>
<td>4</td>
<td>3,5</td>
<td></td>
</tr>
<tr>
<td>JH-512</td>
<td>R8</td>
<td>42</td>
<td>10</td>
<td>4,2</td>
<td>from 35.5 to 42 rounds tweak of JH for Round 3</td>
</tr>
<tr>
<td>KECCAK-512</td>
<td>f<a href="A">1600</a></td>
<td>24</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Skein-512</td>
<td>Threefish-512</td>
<td>72</td>
<td>18</td>
<td>4</td>
<td>with pseudo-near-collision</td>
</tr>
</tbody>
</table>

Table 5: Comparison of the minimum number of rounds with bias exceeding 0, 1%.

4 Conclusion

In this paper we presented some possibilities of CipherCAD application for cryptanalysis. It appears that this is a very good and illustrative tool for cryptanalytic research and comparative analyses. For instance, we have presented an intuitive creation of model of Skein-512 algorithm, avalanche tests for all five finalists of SHA-3 BLAKE, Grøstl, JH, KECCAK and Skein and a comparison of their results. A specific conclusion which we have arrived at in this case is that there are no significant differences among the observed algorithms.

References


Security and Protection of Information 2011 65