Wrocław University of Science and Technology

W04N Cryptography and Computer Security

Z21-22 Cryptography and Security K06-93a

Grzegrzółka Mikołaj

Attack on Grøstl Hash function - homework

1. Task

1. Please study a specification of Grostl hash function algorithm. You may refer to https://www.groestl.info/Groestl.pdf its whitepaper

2. Given an ability to manipulate single operations (like converting an XOR gate to a fixed output via a hardware Trojan), look for an opportunity to corrupt a Grostl hardware so that the resulting function becomes weak.

3. Try to find a solution as simple as possible concerning the number of malicious manipulations.

2. Introduction and a bit Grøstl explanation

Grøstl is a hash function proposed for the NIST hash function competition for the new standard SHA-3, to complement the older variants SHA-1 and SHA-2. Like other hash functions in the MD5/SHA family, Grøstl divides the input into blocks and iteratively computes $h_i = f(h_{i-1}, m_i)$. However, Grøstl maintains a hash state at least twice the size of the final output (512 or 1024 bits), which is only truncated at the end of hash computation.

In the case of Grøstl the compression function f is based on a pair of 256/512-bit permutation functions F and Q, and is defined as:

$$f(h, m) = F(h \oplus m) \oplus Q(m) \oplus h$$

The permutation functions F and Q are heavily based on AES block cipher, but operate on 8x8 or 8x16 arrays of bytes (in AES it's 4x4). And similiar to AES each round consists four operations:

1. AddRoundKey (the Grøstl round keys are fixed, but differ between F and Q)
2. SubBytes (this uses the Rijndael S-box, allowing sharing with AES implementations)
3. ShiftBytes (expanded compared to AES, this also differs between F and Q, and 512- and 1024-bit versions)
4. MixColumns (using an 8×8 matrix rather than Rijndael's 4×4)

Unlike AES, all rounds are identical and there is no final AddROundKey operation. Usually 10 rounds are recomended for 512-bit permutation and 14 for 1024-bit permutation.

The final double-width hash receives a final output transformation of:

$$\Omega(h) = h \oplus F(h)$$

And is then truncated tot he desired width.

Comparison of some Grøstl hashes for word „Götterdämmerung“:

```
Grøstl-224
```

57113d57ce2380cf893e2e95a1a24fe290f246cb2b8ce4e321de4f13

```
Grøstl-256
```

27f864c55952bc89727a4359712a5fcf07b09d35d22830a029ffd7cedaabc211

```
Grøstl-384
```

e09def3a41309a318eb20c5d88e765a37a82756b4ec12e6892ec9e133ed577fb306024aeca57e4d0d6dfaf9a30642ec3

```
Grøstl-512
```

56e93f77f6ce447d6cf211e880d720c4410c9a25bd87090150d5c9b6aaedd2d7b97590de7121b1ad08d04fce0886c507784ffba312b70d8424585bd5ea60387a

In this case also it’s noticable that even the smallest change in the message will drasticly change the overall hash. For example we can try with word „Archolos“ and with same word with end of the sentence „Archolos.“.

```
Grøstl-256("Archolos")
```

62640f3b38aa1b84800dd8352854cb27a583dd8af2249d17a5c12dd7a0ca863e

```
Grøstl-256("Archolos.")
```

f30e7d31d0dbe0e4f64daddaf578a0730fcfef92ea030b15fd8d09141270a42d

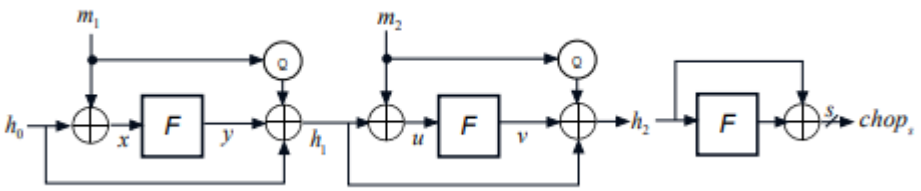
3. Colision attack on Grøstl

Let's look at the compresion function of Grøstl. Let F and Q be a 2n bit permutation:

$$f(h_{i-1}, m_i). = F(h_{i-1} \oplus m_i) \oplus Q(m_i) \oplus h_{i-1}$$

Where $h_{i-1}, m_i \in \{0,1\}^{2n}$.

The two block of Grøstl, will be represented according to the defined function:



Grøstl is using the chopMD mode of operation. The initial value is fixed as IV, the message M is first padded into l message blocks, the usual Merkle-Damgard iteration is applied then to F to compute the last chain value h_l. At the end the final transformation $\Omega(x) = x \oplus F(x)$ is applied to h_l and the final output ist he last s bits of $\Omega(h_l)$.

In this case I will try to explain the collsion attack on Grøstl without the lenght padding. We assume then the F and Q to be two ideal permutations and the adversary never makes repeat queries (Adversary will then never make queries that already knows the result). For attack we use two block of message (m_1, m_2) and we fix the initial value as $h_0 = IV$ where IV is $2n$-bit constant. Looking on two block of Grøstl we let (x, y) be the input and output queried pairs in the first block query. In the second block query (u, v) are the input and output queried pairs. We have then:

$$m_1 = x \oplus h_0$$

$$h_1 = Q(m_1) \oplus h_0 \oplus y$$

And for the second block of message:

$$m_2 = u \oplus h_0$$

$$h_2 = Q(m_2) \oplus h_1 \oplus v$$

The attack will have then seven steps, and proceds as follows:

1. Set h_0 to be the constant IV
2. Choose r random values of x and make queries to F, we get then r random values of (x, y). And since $h_1 = y \oplus Q(m_1) \oplus h_0$, we get r random values of h_1.
3. Choose r random values of u and make queries to F, we get then r random values of (u, v).
4. For each value of (u, v) we are able to compute $m_2 = u \oplus h_0$ and $h_2 = v \oplus Q(m_2) \oplus h_1$. And since there are r random values of h_1, we can get random values of m_2 and h_2.
5. There are total r values of (u, v), we can get then r^2 random values of h_2.
6. If the final hash value is truncated to s bits where $s \leq n$. Let r be $2^{s/4}$, we can get then $r^2 = 2^{s/2}$ random values of h_2 and $chop_s$. According tot he birthday paradox, there exists two pairs of (h_1, m_2) colliding at $chop_s(h_2)$ with probability 0.39.
7. The adversary needs then $2 \times 2^{s/4} + 2^{s/2} = 2^{s/4+1} + 2^{s/2}$ queries to the permutation F and $2^{s/4} + 2^{s/2}$ to the permutation Q to find a collision with probability of 0.39. The best possible collision attack requires $3 \times 2^{s/2}$ queries to F and $2^{s/2+1}$ queries to Q.

4. Hardware Trojan (theorycrafting)

The structure of Grøstl consist of two permutations constructed using trail design strategy, that's why Grøstl is considered very resistance against most crypto attacks. The Grøstl compression function is probably collision resistant and preimage resistant assuming that the permutations F and Q are ideal. Also overall structure is described as indifferientable form random oracle if the permutations F and Q are ideal and completly independent from each other. Descriptions of the Grøstl also mentions that collsion attacks are possible in only 3 rounds limit.

We should be able to manipulate input or output of the XOR gate:

$$h_i = F(0^{2n}) \oplus Q(m_i) \oplus h_{i-1}$$

If we change the output of XOR gate to zeros value, the overall permutation of the F will not be presented in the compression function:

$$Q(m_i) \oplus h_{i-1}$$

This is allready weakened and should be possible to attack. In this case only one permutation Q will be involved in the compression function and it will be only dependent on the message value (not on the key).

We can also make it even more simply by modifying the $Q(m_i) \oplus h_{i-1}$, we will have then:

$$h_i = F(h_{i-1} \oplus m_i) \oplus 0^{2n}$$

In this case F will only be involved in the compression function, and it will look like this:

$$F(h_{i-1} \oplus m_i)$$

References

Grøstl. *Grøstl – a SHA-3 candidate.* http://www.groestl.info/index.html.

Lars Ramklide Knudsen, Matusiewicz, K., Gauravaram, P. i Mendel, F. (2011). *Grøstl - a SHA-3 candidate.*

Luo, Y. i Xuejia, L. (2013). *Attacks on JH, Grøstl and SMASH Hash.*

Mendel, F., Rijmen, V. i Schläffer, M. (2014). *Collision Attack on 5 Rounds of Grøstl.*