Hacking hash functions or how to find a needle in a haystack

Krystian Matusiewicz and Josef Pieprzyk
Centre for Advanced Computing, Algorithms and Cryptography,
Department of Computing, Macquarie University

Summary

Most of the people don't realise how often they are using cryptographic hash functions. In spite of their ubiquity and importance, designing secure hash functions has been more of an art than of science.

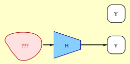
This project aims at developing new cryptanalytical techniques to assess the security of existing designs and acquire knowledge necessary to design next-generation hash functions, with solid security rationale.

What are hash functions?

Hash functions are algorithms that for input bit-streams of any length return the output of fixed length. This means that the hash of a few letters or a whole CD-ROM of data has the same length.

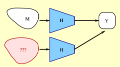
Cryptographic hash functions are very special hash functions that satisfy three additional security properties.

- Preimage resistant : Given an output Y of the hash function h it is difficult to find any *preimage* - an input X such that $h(X) = Y$.

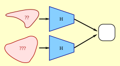


Preimage resistance assures us that the knowledge of the hash value does not reveal anything about the contest of the message.

- Second preimage resistant : Given a fixed input X to the hash function and the corresponding output $h(X)$ it is difficult to find a *second preimage* - another input X', $X' \neq X$ such that $h(X) = h(X')$.



- Collision resistant : It is hard to find any pair of distinct messages (X, X'), $X \neq X'$ such that $h(X) = h(X')$.



Second preimage resistance and collision resistance implies that the digest is "almost unique" for the message. If the message has been altered, the digest will be different (almost always). If two digests are equal, two corresponding messages are the same (almost for sure).

Those properties enable us to treat cryptographic hashes as digital "fingerprints" of messages.

Breaking a hash function: finding a collision or a preimage. The focus is usually on finding collisions. How hard is it? For a perfect hash function producing 160-bit digests, we need 2^{80} computations. Anything less than the theoretical bound of $2^{n/2}$ is a sign of a weakness of the design.

Applications: why do we care?

Digital signatures In digital signatures we don't sign a document but rather its hash. If we can find second preimages for the hash function, we can forge signatures. If we can find collisions, we can trick another person to sign a document and replace it with our version.

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Krystian says: LUCID is fun!
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.3 (GNU/Linux)

iD8DBQFFEyflH20dqWcePcoRApvtAJwIeo8oFuJqIN7y9qMYOCdw2vmhbQCffWOy
e8fWU58k9YnJr4hldiO+Xgc=
=pcyT
-----END PGP SIGNATURE-----
```

Password-based used identification If we can find second preimages for the hash function used by the system, we can circumvent security of password-based login authentication.

In many Linux distributions, passwords in `/etc/shadow` are stored as MD5 hashes

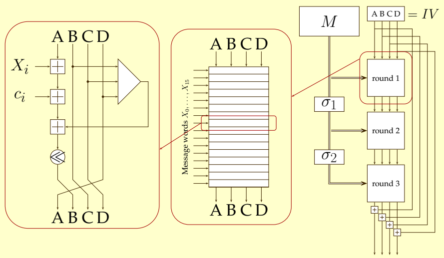
```
kmatus:$1$5IwJJ/.O$9Em5P./CiGE48TVO2QWbz/:13245:0:99999:7:::
bogo:$1$MPW3Z.Au$SJlzrNZUA1qBa8nkUF6Ki.:13245:0:99999:7:::
```

Data integrity verification Message digests are widely used as a means of verifying data integrity. Again, if we can break the hash function in use, we could trick users into installing malicious mutations of linux packages or windows programs.

Identifying data in P2P networks Most of P2P networks identify files using hashes. If we can produce collisions for the hash function used by P2P network, we can "poison" it introducing bogus files that cannot be distinguished from real ones (see the case of UUHash and Kazaa network).

The MD family

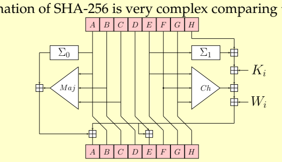
Most of the hash function published in the last 15 years closely follow the design principles of MD4. They are called the "MD" family of hashes and include such popular designs as MD5, SHA-1 and SHA-256. They all exhibit the general structure presented below. Each of a few rounds consists of a number of steps which are message dependent permutations of register bits.



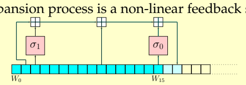
Big goal: analysis of SHA-256

After successful practical attacks on MD5 and theoretical attacks on the U.S. standard SHA-1 the question of security of the improved version called SHA-256 is very interesting. This function is much more complex and we need to learn a lot to be able to fully attack it. As the first step, we analysed different simplified variants to get the feeling how different components influence the behaviour of the function.

Step transformation of SHA-256 is very complex comparing to MD4 or MD5.



Message expansion process is a non-linear feedback shift register.



Cryptanalysis of a variant without Σ_0, Σ_1, σ_0 , σ_1 Without those components which are $\mathbb{F}_2$-linear, we approximated the function by a model linear over $\mathbb{Z}_{2^{32}}$. We had to deal with Boolean functions. We found collision differentials over $\mathbb{Z}_{2^{32}}$ working for a fully linearized model and we picked the one with small XOR weights and approximated Boolean functions.

For this variant, we were able to show that $Pr[collision] = 2^{-64}$.

Cryptanalysis of short variants of SHA-256-XOR Our other direction was to investigate SHA-256 with additions replaced by XORs. This time the main obstacle are Boolean functions.

The outline of our approach can be summarized as follows:

- choose linear approximations of MAJ and IF and construct a $\mathbb{F}_2$–linear model of SHA-256-XOR,
- find a suitable collision-producing difference for the linearized SHA-256-XOR,
- derive a set of conditions under which the real SHA-256-XOR behaves like the linear model with respect to difference propagation,
- find a message for which all the conditions (approximating equations) are satisfied.

Strengths: this framework works for all similarly designed hash functions, we can try to apply them to many designs of the MD family.

Problems to solve: Finding good differentials is hard. We need more efficient algoritms for finding messages satisfying conditions.

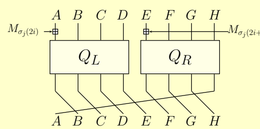
For SHA-256-XOR we can attack variants with with 20-22 steps.

Analysis of alternatives: FORK-256

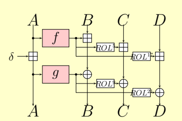
FORK-256 is a recently proposed alternative for SHA-256. Instead of one long run it consists of four short, parallel branches. Each branch is built of eight step transformations which use two message words in different order.

Step transformation consists of the addition of message words, two structures Q_L and Q_R, depicted on the right, and a rotation of registers.

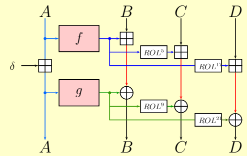
Step transformation of FORK-256



Q_L-structure



Microcollisions in Q_L and Q_R We discovered a way of finding differences in registers A and E that don't spread to other registers – inside of the step transformation differences coming from function f are cancelled out by output differences of g.



Collisions for two branches of FORK-256 Combining microcollision with easy differentials when the difference is present in registers B, C, D and F, G, H only, we were able to find differential paths that easily yield chosen-IV collisions for FORK-256 reduced to two branches.

We expect to improve our results and extend them to versions of FORK-256 with more branches.

About the authors

Krystian Matusiewicz is a PhD student in Department of Computing. His main interest are mathematical methods of cryptanalysis, he is working on the analysis and design of cryptographic hash functions.

Josef Pieprzyk is the director of the Centre for Advanced Computing, Algorithms and Cryptography. His research interests span across many areas of cryptography, one of them is the design and analysis of cryptographic hash functions.