# Hash Functions and Combinatorics on Words

Jan Legerský

jan.legersky@gmail.com

Theoretical Informatics GRoup
FNSPE CTU in Prague

April 25, 2014

# Hash Functions

Hash function is map $f : \{0,1\}^* \rightarrow \{0,1\}^n$ with following properties [4]:

# Hash Functions

Hash function is map $f : \{0,1\}^* \rightarrow \{0,1\}^n$ with following properties [4]:

- Easy computable
  - Time $\mathcal{O}(N)$, memory $\mathcal{O}(1)$.

# Hash Functions

Hash function is map $f : \{0,1\}^* \to \{0,1\}^n$ with following properties [4]:

- Easy computable
  - Time $\mathcal{O}(N)$, memory $\mathcal{O}(1)$.
- Collision resistance
  - It is computationally infeasible to find two different messages $M$ and $M'$ such that $f(M) = f(M')$ .

# Hash Functions

Hash function is map $f : \{0,1\}^* \to \{0,1\}^n$ with following properties [4]:

- Easy computable
  - Time $\mathcal{O}(N)$, memory $\mathcal{O}(1)$.
- Collision resistance
  - It is computationally infeasible to find two different messages $M$ and $M'$ such that $f(M) = f(M')$.
- Second preimage resistance
  - For given message $M_{\text{target}}$ is computationally infeasible to find message $M$ such that $f(M) = f(M_{\text{target}})$.

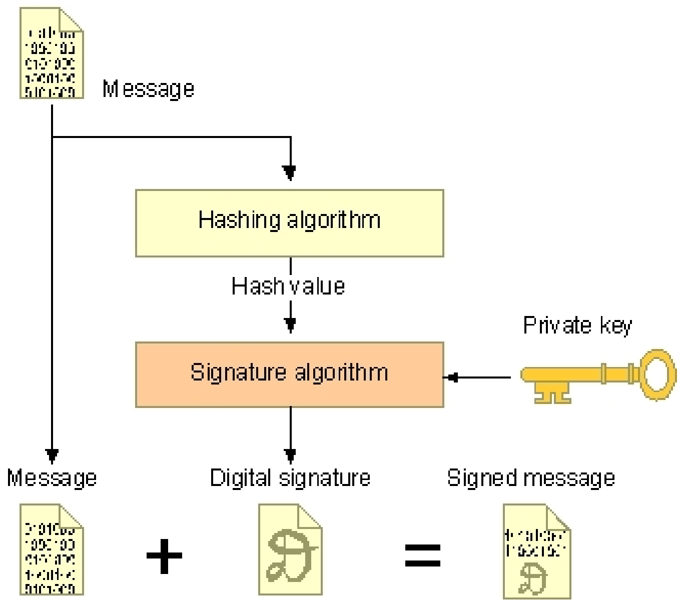
# Hash Functions

Hash function is map $f : \{0,1\}^* \to \{0,1\}^n$ with following properties [4]:

- Easy computable
  - Time $\mathcal{O}(N)$, memory $\mathcal{O}(1)$.
- Collision resistance
  - It is computationally infeasible to find two different messages $M$ and $M'$ such that $f(M) = f(M')$.
- Second preimage resistance
  - For given message $M_{\text{target}}$ is computationally infeasible to find message $M$ such that $f(M) = f(M_{\text{target}})$.
- Preimage resistance
  - For given hash $h_{\text{target}}$ is computationally infeasible to find message $M$ such that $f(M_{\text{target}}) = h_{\text{target}}$.

# Application

- Control of integrity
- Message authentication – HMAC
- Digital signatures
- Password verification
- File or data identifier
- Hash tables
- Pseudogenerators
- Key derivation



# Construction

Merkle-Damgård paradigm – using compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ iteratively.

# Construction

Merkle-Damgård paradigm – using compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ iteratively.

To compute hash of the message $M$ [5, 3]:

1. Pad the message $M$ to fill $m$-bits blocks:

$$M_1 M_2 \ldots M_{\ell-1} \underbrace{M'_\ell || 1 || 00 \ldots 0 || \text{length}(M)}_{m \text{ bits}} .$$

# Construction

Merkle-Damgård paradigm – using compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ iteratively.
To compute hash of the message $M$ [5, 3]:

1. Pad the message $M$ to fill $m$-bits blocks:

$$M_1 M_2 \dots M_{\ell-1} \underbrace{M'_\ell || 1 || 00 \dots 0 || \text{length}(M)}_{m \text{ bits}} .$$

2. Iterate $\ell$-times compression function $F$:

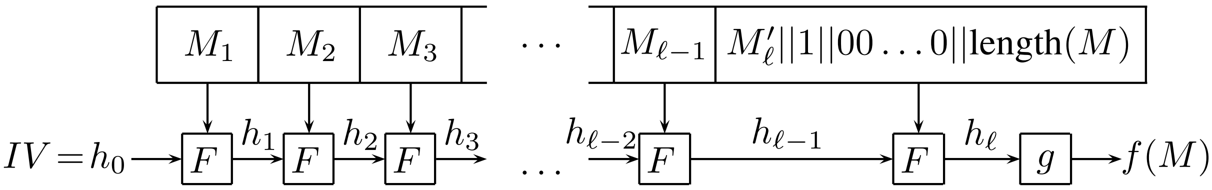
$$h_0 = IV ,$$
$$h_i = F(h_{i-1}, M_i) .$$

# Construction

Merkle-Damgård paradigm – using compression function $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ iteratively.

To compute hash of the message $M$ [5, 3]:

1. Pad the message $M$ to fill $m$-bits blocks:
$$M_1 M_2 \dots M_{\ell-1} \underbrace{M'_\ell || 1 || 00 \dots 0 || \text{length}(M)}_{m \text{ bits}} .$$
2. Iterate $\ell$-times compression function $F$:
$$h_0 = IV ,$$
$$h_i = F(h_{i-1}, M_i) .$$
3. Get digest:
$$f(M) := g(h_\ell) .$$


$M_1$
$M_2$
$M_3$
$\cdots$
$M_{\ell-1}$
$M'_\ell||1||00\ldots0||\text{length}(M)$
$IV = h_0$
$F$
$h_1$
$h_2$
$h_3$
$h_{\ell-2}$
$h_{\ell-1}$
$h_\ell$
$g$
$f(M)$

# Compression function

**Theorem**

If we know an attack against MD scheme, then we know an attack against compression function.

In other words, collision resistant compression function implies collision resistant hash function using MD scheme.

Davies-Meyer construction uses block cipher $E_k(x)$:

$$F(h, M) := E_M(h) \oplus h\,.$$

The weakness is easy findable fixed points:

$$\begin{aligned} h = F(h, M) &= E_M(h) \oplus h \\ 0 &= E_M(h) \\ h &= E_M^{-1}(0)\,. \end{aligned}$$

# Concrete examples

*"The quick brown fox jumps over the lazy dog"*

- MD4: `1bee69a46ba811185c194762abaeae90`
- MD5: `9e107d9d372bb6826bd81d3542a419d6`
- SHA-1: `2fd4e1c67a2d28fced849ee1bb76e7391b93eb12`
- SHA256: `d7a8fbb307d7809469ca9abcb0082e 4f8d5651e46d3cdb762d02d0bf37c9e592`
- Keccak256: `4d741b6f1eb29cb2a9b9911c82f56fa 8d73b04959d3d9d222895df6c0b28aa15`

# Attacks

- Against compression function
  - Dobbertin – MD4
  - Wang, Klíma – MD5
- Generic:
  - Collision attack – birthday attack
  - Multicollision attack – Joux
  - Second preimage attack – expandable message, collision tree,. . .

********************
CONTRACT
At the price of **$176,495** Alf Blowfish
sells his house to Ann Bonidea. .......

********************
CONTRACT
At the price of **$276,495** Alf Blowfish
sells his house to Ann Bonidea. .......

Figure: H. Dobbertin [2] presented algorithm for finding collisions of MD4, conference FSE 1996.

# Birthday paradox

How many people must be in a room to have probability 50% that there is a pair with same birthday?

# Birthday paradox

How many people must be in a room to have probability 50% that there is a pair with same birthday?
We have to pick $l$ elements which have randomly $N$ different values to get pair having same value with probability $p$.

# Birthday paradox

How many people must be in a room to have probability 50% that there is a pair with same birthday?
We have to pick $l$ elements which have randomly $N$ different values to get pair having same value with probability $p$.

$$l \doteq \sqrt{2\ln(\frac{1}{1-p})}\sqrt{N}$$

# Birthday paradox

How many people must be in a room to have probability 50% that there is a pair with same birthday?
We have to pick $l$ elements which have randomly $N$ different values to get pair having same value with probability $p$.

$$l \doteq \sqrt{2\ln(\frac{1}{1-p})}\sqrt{N}$$

So we can find collision of hash function with complexity $2^{\frac{n}{2}}$.

# Multicollision attack

We want to have $k$ messages with the same hash.

# Multicollision attack

We want to have $k$ messages with the same hash.

Ideal hash function: $(2^k!)^{\frac{1}{2^k}} \cdot 2^{\frac{n(2^k-1)}{2^k}}$ calls of compression function.

# Multicollision attack

We want to have $k$ messages with the same hash.

Ideal hash function: $(2^k!)^{\frac{1}{2^k}} \cdot 2^{\frac{n(2^k-1)}{2^k}}$ calls of compression function.

Joux attack against MD scheme: only $k \cdot 2^{\frac{n}{2}}$ calls of compression function.

# Second preimage attack with expandable message

Suppose message $M_{target}$ of size $2^k$ blocks.

1. Find $2^{\frac{n}{2}}$ fixed points $(h_i, M_{fix})$, i. e. $h_i = F(h_i, M_{fix})$.
2. Compute $2^{\frac{n}{2}}$ hashes $h'_j = f(IV, M_1)$.
3. Find collision between these two lists. Denote the colliding value $h_{exp}$.
4. Make an expandable message $M(\ell)$:

$$M(\ell) := M_1 || M_{fix}^{\ell-1}\,.$$

5. Find message $M_{link}$ such that $F(h_{exp}, M_{link}) = \widetilde{h_j}$ for some $j \in \widehat{2^k}$, where $\widetilde{h_j}$ are contexts of $M_{target}$.

Complexity of the attack is $2^{\frac{n}{2}+1} + 2^{n-k}$ calls of compression function instead of $2^n$ against ideal hash function.

# Infinite words

Let $\mathcal{A} = \{a_1, a_2, \ldots, a_k\}$ be the finite alphabet of letters.
Then the sequence $\mathbf{d} = d_1 d_2 d_3 \cdots$, where $d_i \in \mathcal{A}$, is called infinite word over the alphabet $\mathcal{A}$.

Finite nonempty word $w$ is factor of word $\mathbf{d}$, if there are words $x$ and $y$ such that $\mathbf{d} = xwy$.

We denote the set of all factors of the word $\mathbf{d}$ of lenght $m$ by $\mathcal{L}_m(\mathbf{d}) = \{w | w \text{ factor } \mathbf{d}, |w| = m\}$.

# Properties of infinite words

The word **d** is square-free, if it contains no factor $u = ww$, where $w$ is nonempty factor of **d**.

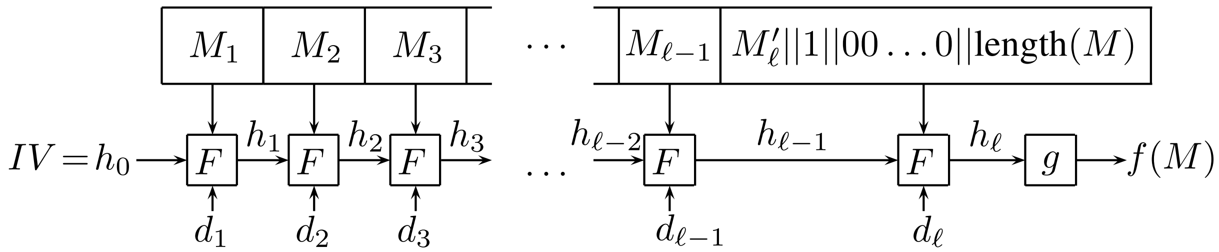
Factor complexity $\mathcal{C}_{\mathbf{d}}(m) : \mathbb{N} \to \mathbb{N}$ of the word **d** is function:

$$\mathcal{C}_{\mathbf{d}}(m) = \#\mathcal{L}_m(\mathbf{d}).$$

# Dithering

The improvement of second preimage resistance.
We add the letter $d_i$ of infinite word $\mathbf{d}$ to input of compression function:

$$h_i = F(h_{i-1}, M_i, d_i)\,.$$



- Square-free word **d** disables attack using expandable message.
- The high-complexity word or the word over large alphabet makes more difficult the other attacks.

# Attacks on the dithered hash functions

- Collision tree of depth $l$
  - $4\sqrt{2\ln 2}\cdot\sqrt{l}\cdot 2^{\frac{n+l}{2}} + C_{\mathbf{d}}(l+1)\cdot 2^{n-k} + 2^{n-l}$

Note: $M_{target}$ has length $2^k$ blocks.

# Collision tree attack

- The nodes of collision tree are marked by hashes, the edges by message blocks with the dither letter according to level of the tree.
- Every hash is colliding hash of its leaves and blocks in the edges.
- The root of collision tree is due to birthday paradox linked to the target message.
- Prefix of required length is linked to one of leaves.
- Complexity is $4\sqrt{2\ln 2}\cdot\sqrt{l}\cdot 2^{\frac{n+l}{2}} + \mathcal{C}_{\mathbf{d}}(l+1)\cdot 2^{n-k} + 2^{n-l}$ calls of compresion function.

# Construction of dither sequence

**Theorem**

*Let* $\mathbf{u} = u_1 u_2 u_3 \cdots$ *be the word with complexity* $\mathcal{C}_\mathbf{u}(m)$ *over the alphabet* $\mathcal{A}$*, the word* $\mathbf{v} = v_1 v_2 v_3 \cdots$ *is square-free over the alphabet* $\mathcal{B}$ *and* $\mathcal{A} \cap \mathcal{B} = \emptyset$*.*
*Then* $\mathbf{u} = u_1 v_1 u_2 v_2 u_3 v_3 \cdots$ *is square-free and it holds* $\mathcal{C}_\mathbf{d}(2m) \geq \mathcal{C}_\mathbf{u}(m)$*.*

We get the word $\mathbf{u}$ with the exponential complexity $\mathcal{C}_{\mathbf{u}}(m) = 2^m$ by concatenation of binary expansions of natural numbers:

$$\mathbf{u} = 11011100101110111 \cdots .$$

The word $\mathbf{v}$ is generated as follows:
We suppose morfism $\tau$ defined over the alphabet $\{A, B, C, D\}$ as

$$\tau(A) = AB, \quad \tau(B) = CA, \quad \tau(C) = CD, \quad \tau(D) = AC\,.$$

Morfism $\mu$ is applied on the fixed point $\tau^{\infty}(A)$:

$$\mu(A) = 4, \quad \mu(B) = 3, \quad \mu(C) = 2, \quad \mu(D) = 3\,.$$

The word $\mathbf{v} = \mu(\tau^{\infty}(A)) = 432423432342\cdots$ is Thue's square-free word.

By shuffling **u** and **v** is formed square-free word

$$\mathbf{d} = 14130214121304031203141 21\cdots$$

over the alphabet $\{0,1,2,3,4\}$ with the complexity $\mathcal{C}_{\mathbf{d}}(m) \geq 2^{\frac{m}{2}}$.
The complexity of the collision tree attack using dither sequence **d** is

$$4\sqrt{2\ln 2}\cdot\sqrt{l}\cdot 2^{\frac{n+l}{2}} + \mathcal{C}_{\mathbf{d}}(l+1)\cdot 2^{n-k} + 2^{n-l} \geq$$
$$\geq 4\sqrt{2\ln 2}\cdot\sqrt{l}\cdot 2^{\frac{n+l}{2}} + 2^{\frac{l+1}{2}}\cdot 2^{n-k} + 2^{n-l}.$$

That means to increase $k$ from $\frac{n}{3}$ to $\frac{n}{2}$ to keep the same complexity as the classical hash function.

Thanks for Your attention.

# References I

F. J. Brandenburg.
Uniformly Growing k-th Power-Free Homomorphisms.
*Theoretical Computer Science*, 23:69–82, 1983.

H. Dobbertin.
Cryptanalysis of MD4.
*Journal of Cryptology*, 11(4):253–271, 1998.

J. Kelsey and B. Schneier.
Second Preimages on n-bit Hash Functions for Much Less than $2^n$ Work.
In *Proceedings of the 24th annual international conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'05, pages 474–490, Berlin, Heidelberg, 2005. Springer-Verlag.

# References II

- A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone.
  *Handbook of Applied Cryptography*.
  CRC Press, 1996.
- R. L. Rivest.
  Abelian Square-free Dithering for Iterated Hash Functions.
  *Presented at ECrypt Hash Function Workshop, June 21, 2005, Cracow, and at the Cryptographic Hash workshop, November 1, 2005, Gaithersburg, Maryland (August 2005)*, 2005.