Cryptanalysis of GRINDAHL

Asiacrypt 2007 - Kuching, Malaysia

Thomas Peyrin

Orange Labs

AIST

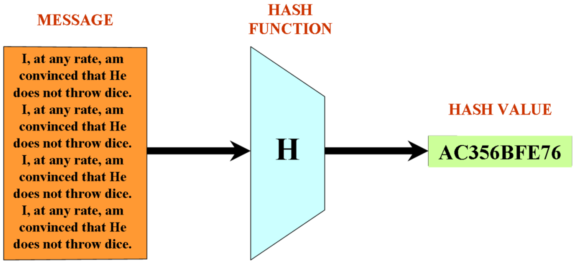
University of Versailles

December 6, 2007

Outline

Outline

What is a hash function ?

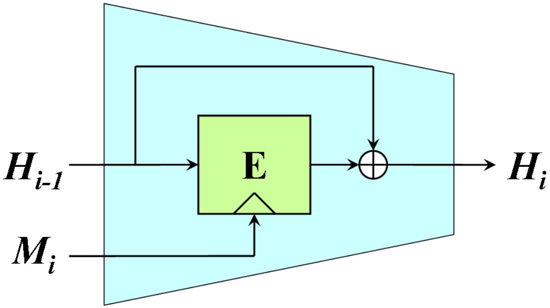


- H maps an **arbitrary length input** (the message M) to a **fixed length output** (typically $n = 128$, $n = 160$ or $n = 256$).
- H must be collision ($2^{n/2}$ function calls), 2nd-preimage (2^n function calls) and preimage resistant (2^n function calls).

How to build a hash function (usually) ?

compression function + domain extension algorithm.

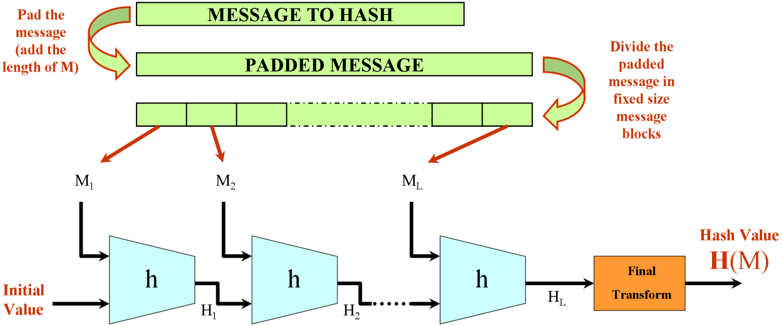
The Davies-Meyer construction



How to build a hash function (usually) ?

compression function + **domain extension algorithm**.

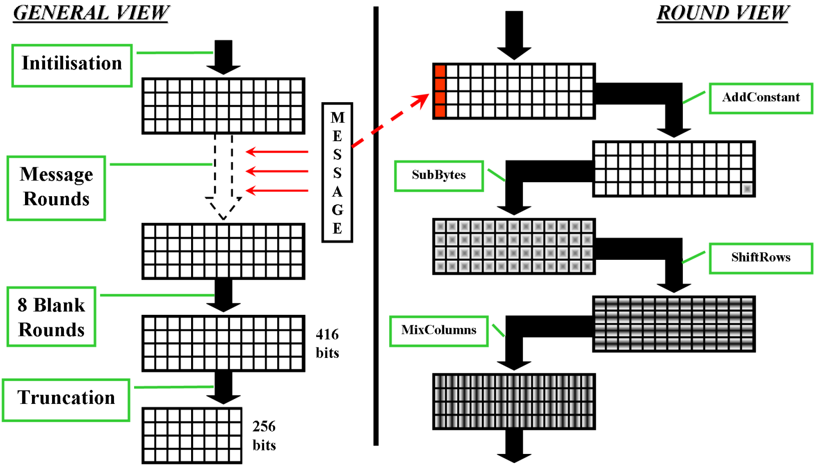
The Merkle-Damgård algorithm



GRINDAHL (Knudsen, Rechberger, Thomsen - 2007)

- 256-bit output (a 512-bit version is also defined).
- no Merkle-Damgård, nor Davies-Meyer construction !
- use a **big internal state S**: 4×13 **matrix of bytes.**
- process 4 new bytes of message each round.
- a round uses **Rijndael** parts: MixColumns, SubBytes, ShiftRows (with rotations 1, 2, 4, 10 for better diffusion) and AddRoundKey is replaced by the addition of a constant.
- **blank rounds** without incoming message after having processed all the message.
- then truncation of S for a 256-bit output.

High-level view of GRINDAHL



Properties of GRINDAHL

- faster than SHA-256 and low memory requirements: can benefit from the fast/small AES implementations.
- collision resistance, 2nd preimage and preimage resistance in $2^{n/2}$ function calls (possibility of meet-in-the-middle attacks for (2nd)-preimage).
- **main security arguments:**
  - a collision requires intermediate states with at least half of the bytes active.
  - an internal collision requires at least 5 rounds.

It is very hard to find a low-weight and-or a small differential path for GRINDAHL.

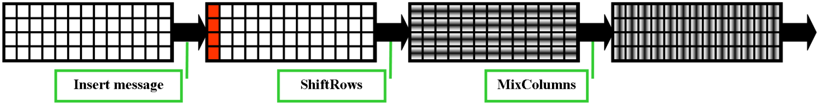
Outline

1. The GRINDAHL Family of Hash Functions
2. First Observations
3. General Strategy
4. The Collision Attack

Truncated differentials

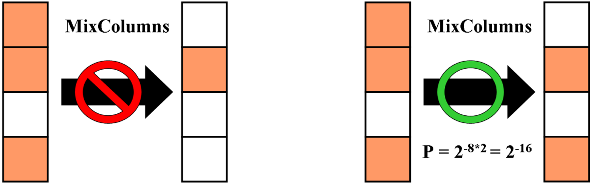
- the scheme is byte oriented.
- let's deal with **truncated differences**: only check if there is a difference in a byte, but don't care about the actual value of the difference.
- we can forget about SubBytes and the constant addition (transparent for truncated differentials).
- *we only deal with ShiftRows, MixColumns and truncation.*

The simplified scheme we consider:



The MixColumns function

- How do the truncated differentials react with the MixColumns function ?
- **Property of MixColumns:** $\sharp\{\text{input byte-differences}\} + \sharp\{\text{output byte-differences}\} \geq 5$.
- **P[valid transitions] =** $2^{-8\times(4-\sharp\{\text{output byte-differences}\})}$.

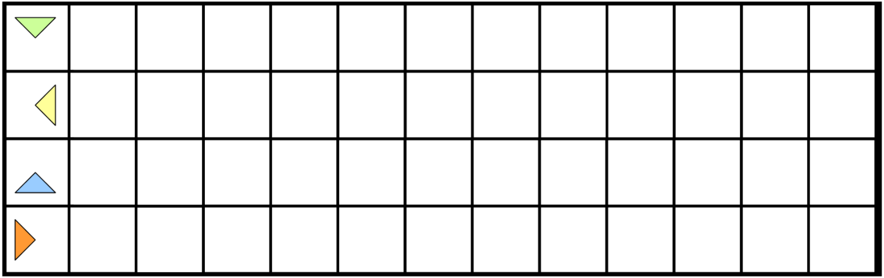


The control bytes (1)

- ShiftRows modified (1, 2, 4, 10) for better diffusion: every state byte depends on every message byte after 4 rounds.
- ... but what happens before those 4 rounds ?
- each message byte inserted affect some subset of the internal state S.
- **this will allow us to control a little bit the difference spreading by forcing some MixColumns differential transitions independently.**
- we call them **control bytes**.

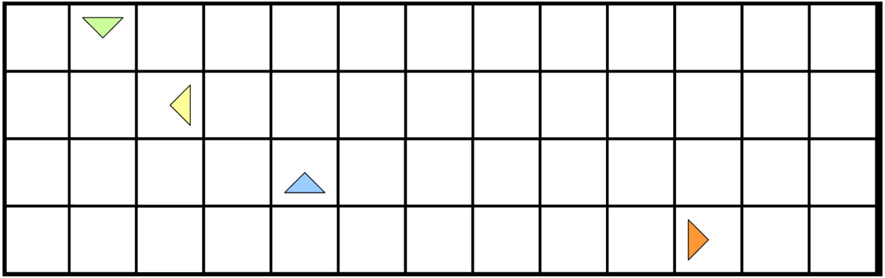
The control bytes (2)

- **Insert the message bytes.**



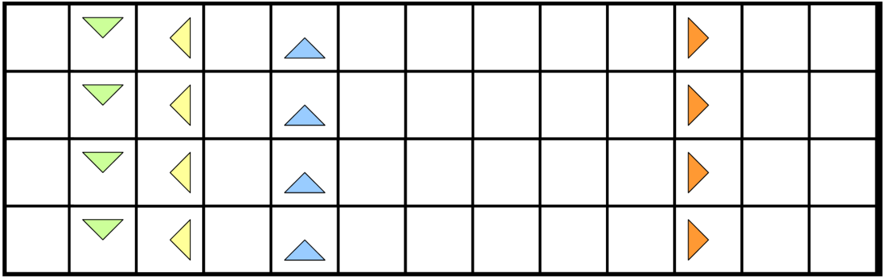
The control bytes (2)

- Do **ShiftRows** (1^{st} round).



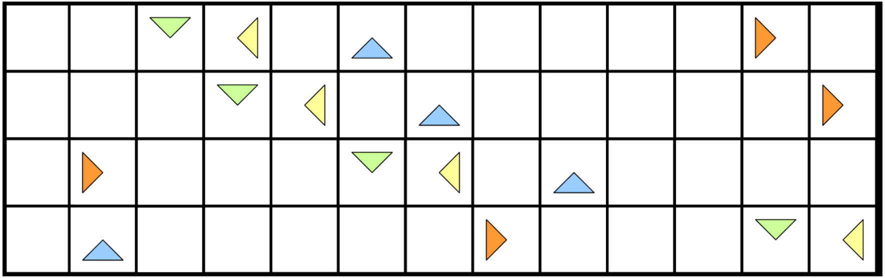
The control bytes (2)

- Do **MixColumns** (1^{st} round).



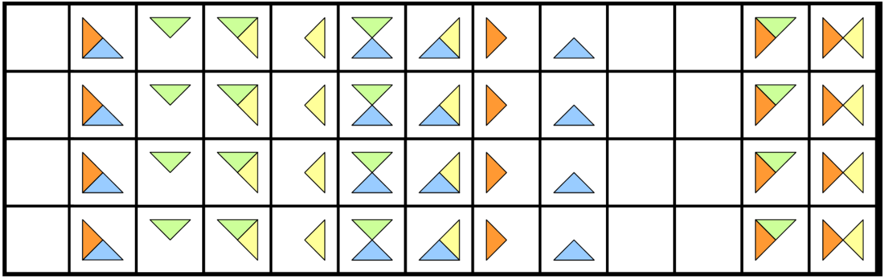
The control bytes (2)

- Do **ShiftRows** (2^{nd} round).



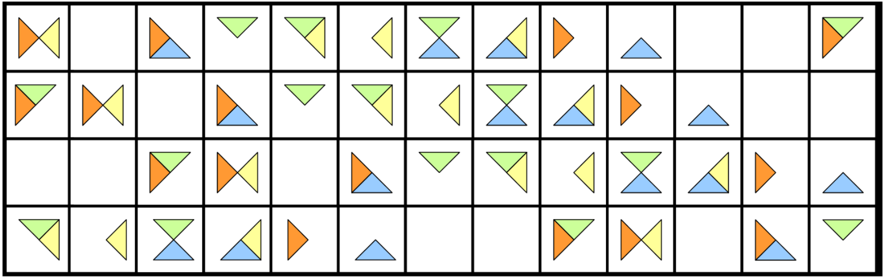
The control bytes (2)

- Do **MixColumns** (2^{nd} round).



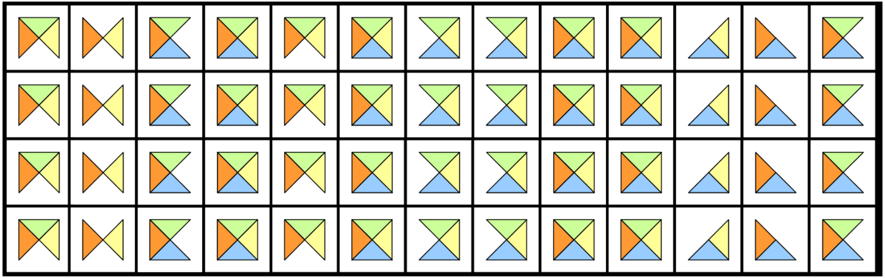
The control bytes (2)

- Do **ShiftRows** (3^{rd} round).



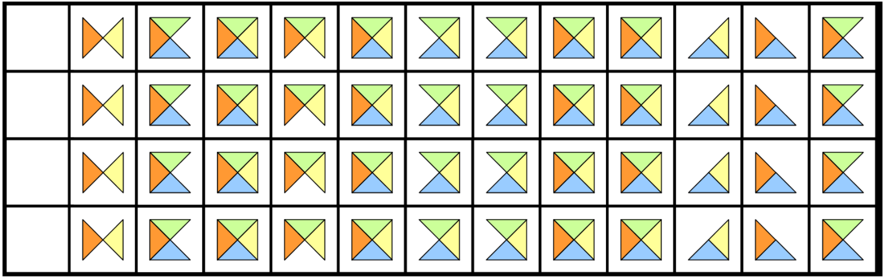
The control bytes (2)

- Do **MixColumns** (3^{rd} round).



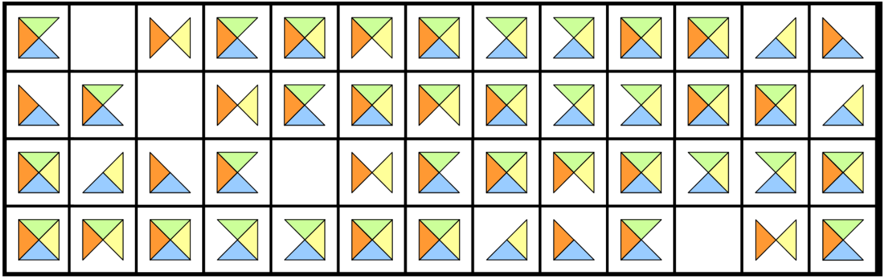
The control bytes (2)

- **Truncation of the first column** (new message bytes).



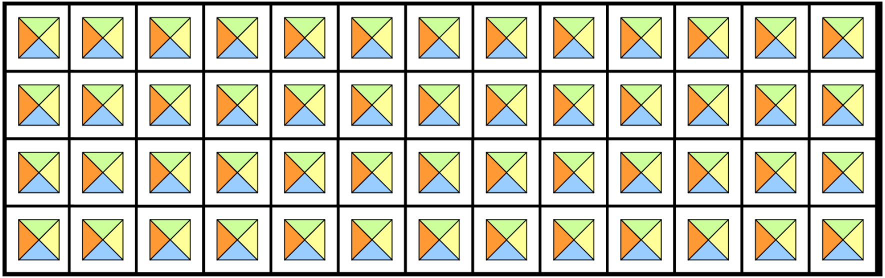
The control bytes (2)

- Do **ShiftRows** (4^{th} round).



The control bytes (2)

- Do **MixColumns** (4^{th} round).



Outline

Internal collisions are better

- 2 possiblilities for a collision: internal or not.
- the blank rounds would make things really hard since we have no more control (no more message byte inserted).
- an **internal collision** seems easier, even if we can not use the final truncation anymore (we'll have a bigger internal state to make collide).
- **2 possibles ways to erase a truncated difference**: with a MixColumns transition (for a cost P^{-1}) or thanks to the truncation during a message insertion (no cost since already planed in the differential path).

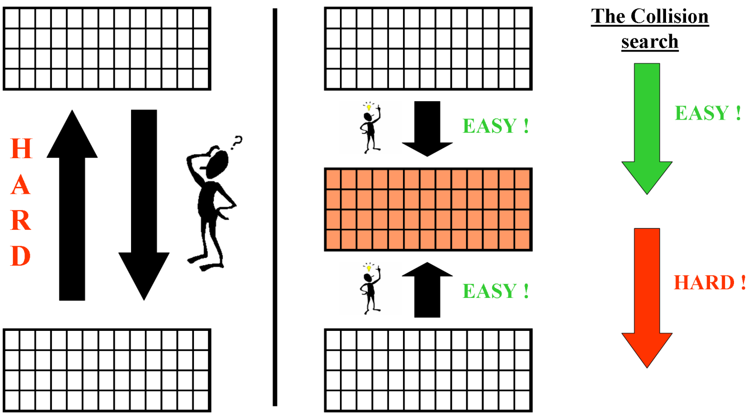
An unintuitive strategy

- Building a differential path is really hard because of the two security properties.
- **idea - take the all-difference state as a check point:**
  - from a no-difference state to an all-difference state: hopefully very easy ! No need for a differential path here.
  - from an all-difference state to a no-difference state: harder ! Build the differential path backward and search for a collision onward.
- the costly part is obviously the second stage !

That is an unintuitive strategy for a hash function cryptanalyst: we deliberately let all the differences spread in the whole state before beginning the collision search !

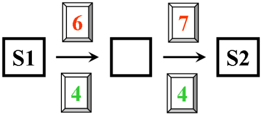
How to build a differential path

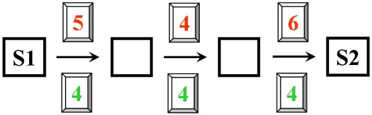
Building a differential path is really hard !



Differential path and control bytes

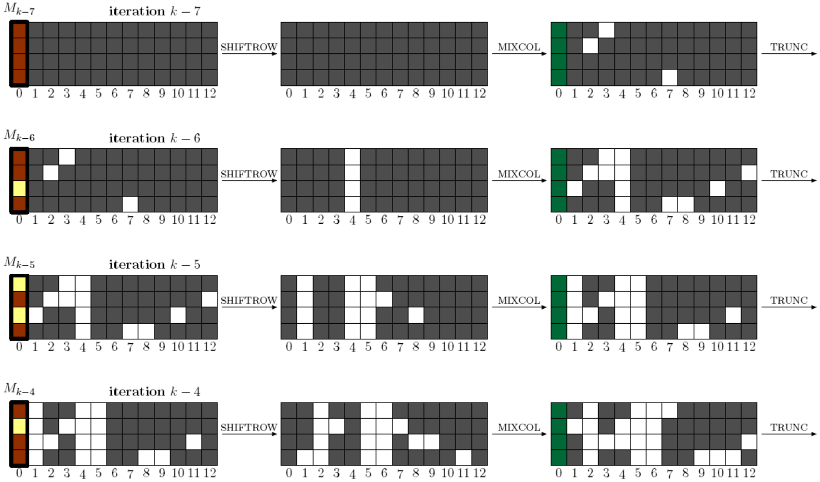
- several differential paths are possibles.
- some give better probability of success than others ... but we will use the control bytes to force some MixColumns independently.
- **dilution effect: it may be better to use less probable paths but longer ones (more message/control bytes gained than probability decrease).**
- this whole differential path trade-off search can be automated.



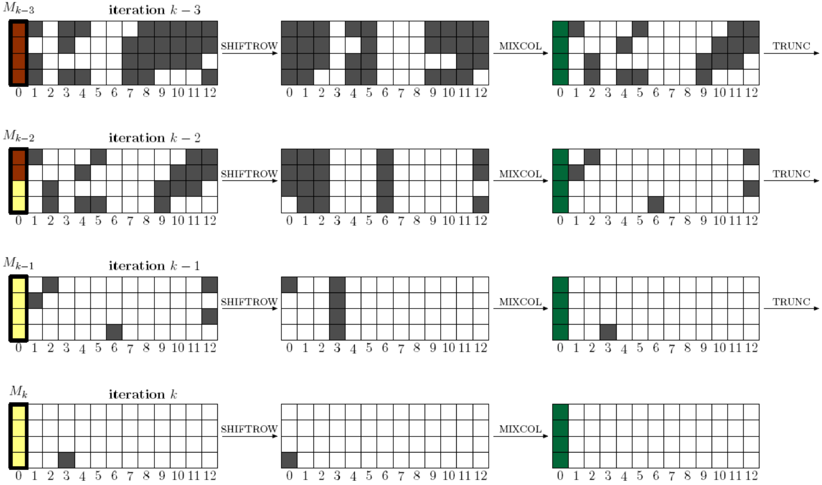


Outline

Our truncated differential path (1)



Our truncated differential path (1)



Results

One can find a collision for the full GRINDAHL with a complexity of 2^{112} functions calls approximatively (2^{128} in the ideal case).

- please read the paper for the details !
- may also work for the 512-bit version but the differential path search tree is too big.
- is the internal state of GRINDAHL too small ? it is possible to patch the scheme to provide good security arguments regarding this kind of attack.

Thank you!