Cryptanalysis of FORK-256

Krystian Matusiewicz[1], Thomas Peyrin[2], Olivier Billet[2], Scott Contini[1] and Josef Pieprzyk[1]

[1]Centre for Advanced Computing Algorithms and Cryptography, Department of Computing, Macquarie University

[2]Network and Services Security Lab, France Telecom Research and Development

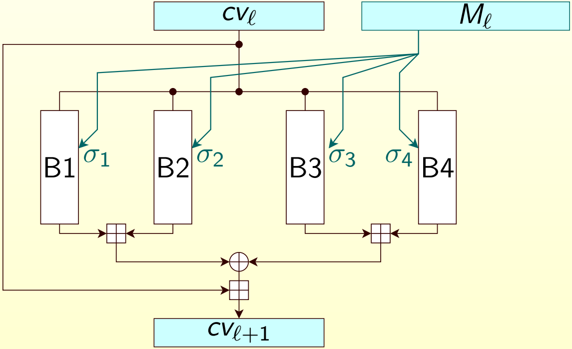
FSE 2007, 26 March 2007

Talk overview

- Short description of FORK-256
- Micro-collisions in the step transformation
- Simple differential path for the compression function
- General method of finding differential paths
- Collisions for the compression function
  - The differential path
  - Complexity analysis
  - Improving efficiency using large memory
  - Achieving collisions for the hash function
- Conclusions

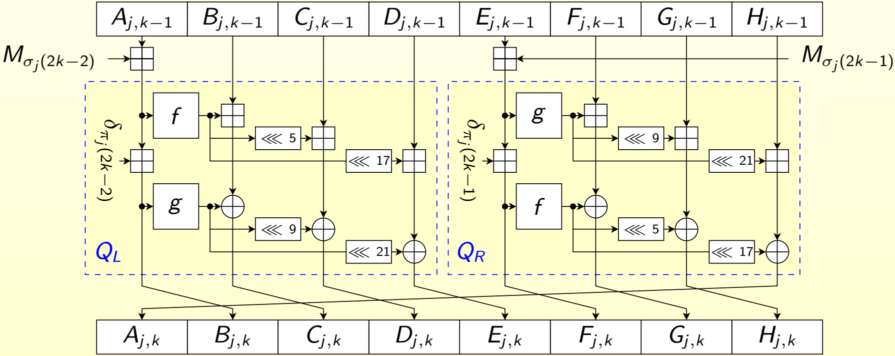
- Short description of FORK-256
- Micro-collisions in the step transformation
- Simple differential path for the compression function
- General method of finding differential paths
- Collisions for the compression function
  - The differential path
  - Complexity analysis
  - Improving efficiency using large memory
  - Achieving collisions for the hash function
- Conclusions

Structure of FORK-256 :: four parallel branches



- 256 bits of chaining variable *cv*
- 512 bits of message *M*
- each branch B1, B2, B3, B4 consists of **8 steps**
- each branch uses a different permutation $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ of message words $M_0, \ldots, M_{15}$

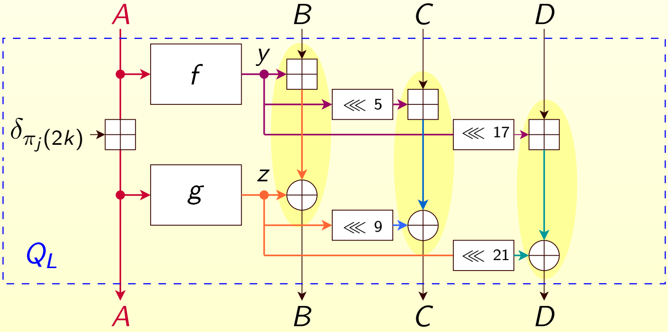
Structure of FORK-256 :: step transformation



- there are 8 steps in each branch
- step transformation – composition of 3 simple operations
  - addition of two different message words
  - two parallel **Q-structures**
  - rotation of registers

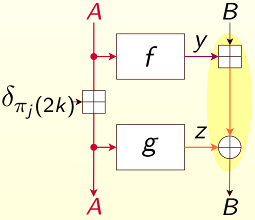
- Short description of FORK-256
- Micro-collisions in the step transformation
- Simple differential path for the compression function
- General method of finding differential paths
- Collisions for the compression function
  - The differential path
  - Complexity analysis
  - Improving efficiency using large memory
  - Achieving collisions for the hash function
- Conclusions

What is a "micro-collision"?



Micro-collision: a difference in register A does not propagate to the selected register B, C or D.

If it does not propagate to more than one other register we have *simultaneous micro-collisions*.



Let us denote

$$y = f(x), \quad y' = f(x') \qquad z = g(x \boxplus \delta), \quad z' = g(x' \boxplus \delta).$$

We have a micro-collision in the first line if the equation

$$(y \boxplus B) \oplus z = (y' \boxplus B) \oplus z' \tag{1}$$

is satisfied for given y, y', z, z' and some constant B.

Our aim is to find the set of all constants B for which (1) is satisfied.

Three representations of a difference

- usual XOR difference:

$$\Delta^{\oplus}(z, z') = (z_0 \oplus z'_0, \ldots, z_{31} \oplus z'_{31}) \quad \in \{0,1\}^{32}$$

- integer difference:

$$\partial y = y' - y \quad \in \{-2^{32}+1, \ldots, 2^{32}-1\}$$

- singed binary difference:

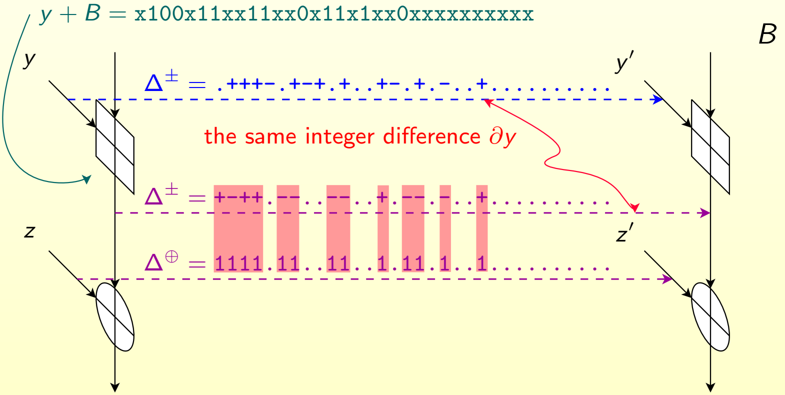
$$\Delta^{\pm}(y, y') = (y_0 - y'_0, \ldots, y_{31} - y'_{31}) \quad \in \{-1,0,1\}^{32},$$

Two useful relationships between different representations

- If $\Delta^{\pm}(y, y') = (r_0, r_1, \ldots, r_{31})$ is a signed binary difference, then the corresponding XOR difference is $(|r_0|, |r_1|, \ldots, |r_{31}|)$.
- Having a signed binary difference we can easily recover the (unique) corresponding integer difference:

$$\partial y = \sum_{i=0}^{31} 2^i \cdot \Delta^{\pm}(y, y')_i \quad .$$

Finding micro-collisions: The principle



XOR difference $\Delta^{\oplus} \longrightarrow 2^{h_w(\Delta^{\oplus})}$ signed binary diffs $\longrightarrow$ $2^{h_w(\Delta^{\oplus})}$ integer diffs $\longrightarrow$ one of them must be $\partial y = y - y'$

Finding micro-collisions: Necessary condition

To test whether the quadruple (y, y', z, z') may yield a micro-collision we have to check whether there exists a signed binary representation corresponding to $\partial y = y - y'$ that "fits" into XOR difference $\Delta^{\oplus}(z, z')$.

This problem can be reduced to an easy (superincreasing) knapsack problem:

> *Having a set of positions* $I = \{k_0, k_1, \ldots, k_m\}$ *(determined by non-zero bits of* $\Delta^{\oplus}(z, z')$*), decide whether it is possible to find a binary signed representation* $r = (r_0, \ldots, r_{31})$ *corresponding to* ∂y *s.t.:*
>
> $$\partial y = \sum_{i=0}^{m} 2^{k_i} \cdot r_{k_i} \quad \text{where } r_{k_i} \in \{-1, 1\} \ .$$

This test can be implemented very efficiently!

```
int micro_possible(WRD y1, WRD y2, WRD dz) {
    WRD tmp, delta_y, sum;
    if ( y2 > y1 ) {
        tmp = y2; y2 = y1; y1 = tmp;
    }
    delta_y = y1 - y2;
    sum = delta_y;
    sum += dz;
    if ( sum < delta_y ) {
        if ( (dz>>31)==0 )
            return 0;
    }
    dz <<= 1;
    return ( (dz|sum) == dz );
}
```

Finding micro-collisions: Also a sufficient condition

In fact we can prove that this condition is also sufficient: if we can find such a representation, we can always find constants B that make the difference "fit" into the prescribed XOR pattern.

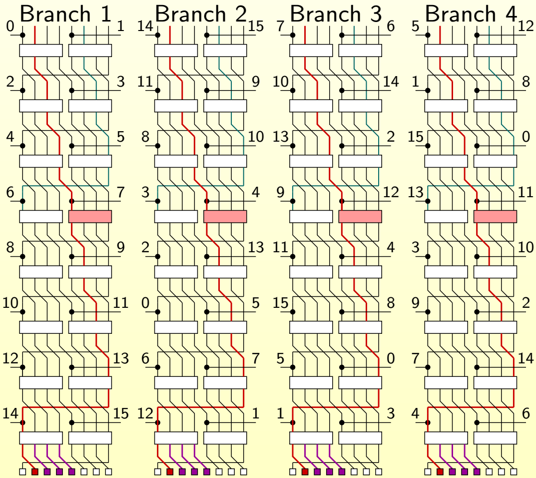
Moreover, the analysis shows that the size of the set of good constants B is equal to

$$2^{32-h_w(z \oplus z')+1} ,$$

with the grey one added if the MSB of $\Delta^{\oplus}(z, z')$ is one.

- Short description of FORK-256
- Micro-collisions in the step transformation
- Simple differential path for the compression function
- General method of finding differential paths
- Collisions for the compression function
  - The differential path
  - Complexity analysis
  - Improving efficiency using large memory
  - Achieving collisions for the hash function
- Conclusions

Simple differential path using micro-collisions



By introducing differences in B_0 and finding simultaneous microcollisions in four Q-structures in step 4 we obtain a differential restricted to 4 registers.

Simple path: complexity analysis

- Once we pass through step 4, we can generate 2^{32} pairs,
- To pass step 4 we have to make a few simple checks for 2^{32} values, altogether equivalent to $2^{32}/4$ of FORK evaluations, we succeed with probability P_d^6, where P_d depends on the difference, for $d = 0x00000404$ we have $P_d \approx 2^{-3}$.
- the average cost of a single solution $\approx 1/4 \cdot P_d^{-6} \approx 2^{16}$.
- an example of a pair with output difference of weight 22:

cv_n	8406e290	5988c6af	76a1d478	0eb60cea	f5c5d865	458b2dd1	528590bf	c3bf98a1
cv'_n	8406e290	5988cab3	76a1d478	0eb60cea	f5c5d865	458b2dd1	528590bf	c3bf98a1
M	396eedd8	0e8c2a93	b961f8a4	f0a06fc6	9935952b	e01d16c9	ddc60aa4	0ac1d8df
	c6fef1d8	4c472ca6	58d9322d	2d087b65	7c8e1a26	71ba5da1	ba5d2bfc	1988f929
cv_{n+1}	9897c70a	4e18862d	b4725ac1	cfc9f92c	9aa0637d	ae772570	74dd4af1	cd444dd7
cv'_{n+1}	9897c70a	4e1880f9	1e677302	4c650966	f4792bf4	ae772570	74dd4af1	cd444dd7

- Short description of FORK-256
- Micro-collisions in the step transformation
- Simple differential path for the compression function
- General method of finding differential paths
- Collisions for the compression function
  - The differential path
  - Complexity analysis
  - Improving efficiency using large memory
  - Achieving collisions for the hash function
- Conclusions

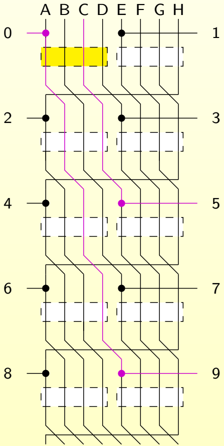
Finding high-level paths: idea and model

Let's be optimistic:

- Assume that we can always avoid mixing introduced by Q-structures (finding micro-collisions is always easy).
- Assume that any two differences cancel each other (i.e. we don't need to worry about many different values, either there is a difference or not and any two differences added together disappear).

So now we are in $\mathbb{F}_2$...

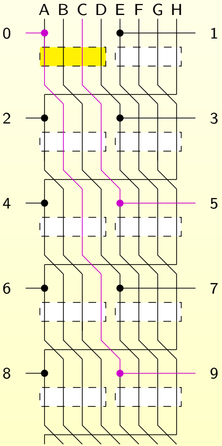
- The model is $\mathbb{F}_2$-linear function L_{out} that maps input differences in M and cv_n to output diffs.
- We can find the kernel of this map to get the set of all input differences that vanish at the output.



Finding high-level paths: going back to reality

The more micro-collisions we have to find and the longer the path the smaller probability that differences in the original function will follow the path.

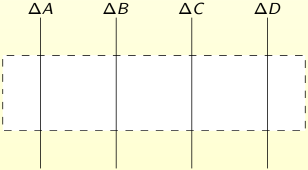
- We look for paths with as few micro-collisions as possible (a few differences in registers A and E)
- Generally, the shorter path the better.
- Let's look at the registers A and E and pick those input differences S that yield only a few non-zero differences in A and E.
- Optimal paths – minimum weight words in a linear code.

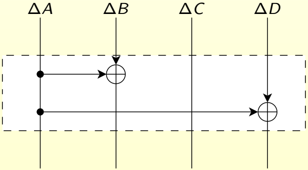


Finding high-level paths: more general model

So far we assumed that differences in A (or E) do not propagate to any other registers in the Q-structure. We can relax this condition.

$(q_B, q_C, q_D) = (0, 0, 0)$ $\qquad$ $(q_B, q_C, q_D) = (1, 0, 1)$



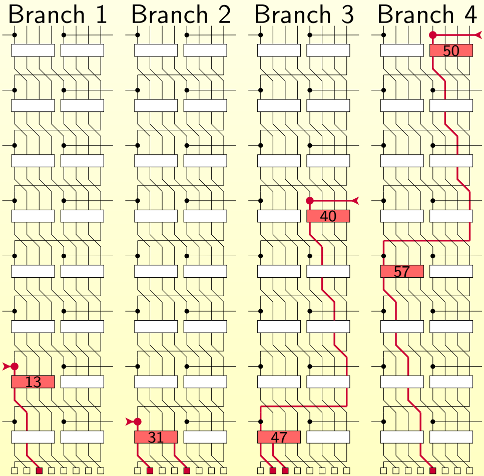


$$
\begin{aligned}
\Delta A_{i+1} &= \Delta A_i \ , \\
\Delta B_{i+1} &= \Delta B_i + q_B \cdot \Delta A_i \ , \\
\Delta C_{i+1} &= \Delta C_i + q_C \cdot \Delta A_i \ , \\
\Delta D_{i+1} &= \Delta D_i + q_D \cdot \Delta A_i \ .
\end{aligned}
$$

For each Q-structure we have $2^3 = 8$ possible configurations. This gives 8^{64} different models for FORK-256 – more freedom to look for short differential paths.

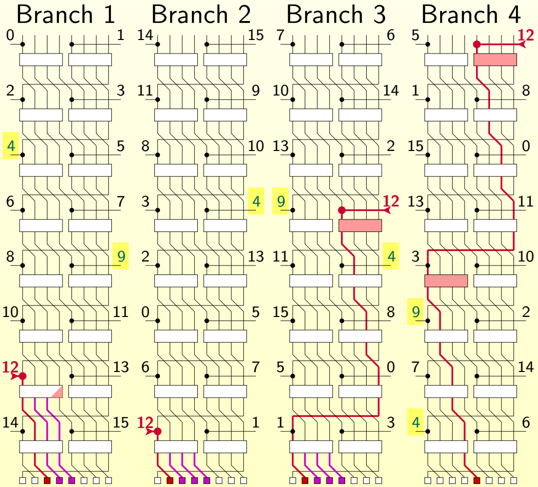
Example of a path: Collisions for all branches

Differences in M_{12}. Configuration of Q-structures: 13:(q_B, q_C, q_D)=000, 31:001, 40:000, 47:100, 50:000, 57:000



- Short description of FORK-256
- Micro-collisions in the step transformation
- Simple differential path for the compression function
- General method of finding differential paths
- Collisions for the compression function
  - The differential path
  - Complexity analysis
  - Improving efficiency using large memory
  - Achieving collisions for the hash function
- Conclusions

Collisions: The differential path



$d = \texttt{0xdd080000}$ or $d = \texttt{0x22f80000}$

- ▶Using a modified path we need micro-colls in only $3\frac{1}{3}$ Q-structures.
- ▶Get 3 micro-collisions in branches 3 and 4 first.
- ▶Using different values of M_4 and M_9 compute branch 1 and hope there is a single micro-collision in Br. 1 step 7.
- ▶Using d with only 13 MSB set only 108 bits are affected.

Collisions: the complexity of getting full collisions

- Complexity of finding a single solution: $2^{18.6}$.
- Now, if the distribution of outputs is close to uniform, we expect to find a collision after testing 2^{108} pairs.

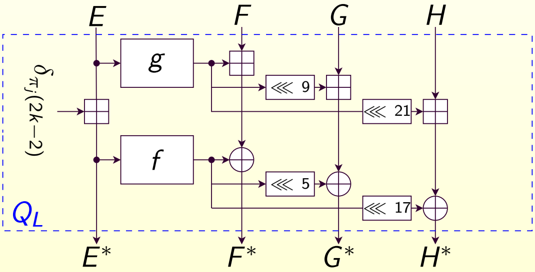
Complexity of finding a collision: $2^{108} \cdot 2^{18.6} = 2^{126.6}$.

- *faster than by birthday paradox*
- *using only small memory (hash table + stored allowable values: 2^{23} 32-bit words in total)*
- *trivially parallelizable*
- *practical for obtaining near-collisions*

Example of a near-collision for the compression function with weight 28

IV	6a09e667	db1bb914	3c6ef372	a54ff53a	510e527f	767b0824	66410f7d	90f7ce64
M	85a83e55	91d3ca9d	a6c2facb	027afd32	000000cb	00000000	9d4a6aba	00000000
	e649c148	4606ae35	6efb18d8	2d6ade8f	1dcb6936	ec995db1	d2ad257b	730f5bb4
M'	85a83e55	91d3ca9d	a6c2facb	027afd32	000000cb	00000000	9d4a6aba	00000000
	e649c148	4606ae35	6efb18d8	2d6ade8f	40c36936	ec995db1	d2ad257b	730f5bb4
diff	**00000000**	**8c300000**	**1d010204**	**52520104**	**c0908122**	**00000000**	**00000000**	**00000000**

Collisions: improving efficiency using large tables



Problem: To what extent can we influence the values of E^*, F^*, G^*, H^* changing only E ?

- We can set E^* to any value (bijective map),
- For any given pair (G, w) we can *very often* find such E that $G^* = w$.
- We can precompute a look-up table T that for any pair (G, G^*) returns the necessary value of E, $T(G, G^*) = E$.

Collisions: improving efficiency using large tables

- We can use such look-up tables to significantly reduce the time spent in branch 1
- Theoretical complexity of finding a single solution: $2^{1.6}$.

Complexity of finding a collision: $2^{108} \cdot 2^{1.6} = 2^{109.6}$.

- *we improved the speed by the factor of* 2^{17},
- *but we assume we can use huge, fast memory,*
- *we use around 512 tables (family parametrized by a), each one of size* 2^{64} *32-bit words, i.e.* 2^{73} *words of memory in total*

Collisions for the full hash function: principle

- We can avoid using B_0 in branch 3 either by using look-up tables or by a smarter scheduling in branch 3 we have to have only three IV words (F_0, G_0, H_0) set to one of the good constants to allow for micro-collisions in step 1 of branch 4.
- Probability that a random IV has all three values (F_0, G_0, H_0) acceptable to the algorithm is bigger than $2^{-3 \cdot 32}$, in fact around 2^{-65} for differences `0xdd080000` and `0x22f80000`.
- At the cost of 2^{65} FORK evaluations we can find a prefix message block that after the first application of the compression function yields IV suitable for the main part of the attack.

Collisions for the full hash function: example

- For other modular differences this probability is much bigger.
- Using “easier” modular difference we’ve got near-collisions for the full hash function with Hamming weight 42.
  However, this modular difference is not as effective when it comes to solving branch 1.

Example of a near-collision for the full hash function with weight 42

M	2d4458a4	57976f57	3e44cfd9	1ab54cb2	7ec11870	173f6573	6141c261	7db20d3e
	2feeb74d	5fac87a6	61a73fa1	3454b23d	451d389b	78f061ec	7c32fb06	57ef1928
	79dcd071	39dc97f0	3a1bff42	031d364c	fef000e6	40873ef5	d0741256	649430cf
	97ef5538	3eab6a7e	b4f9cf72	9eba8257	4e84d457	5a6c49b6	ad1d9711	0f69afa2
M'	2d4458a4	57976f57	3e44cfd9	1ab54cb2	7ec11870	173f6573	6141c261	7db20d3e
	2feeb74d	5fac87a6	61a73fa1	3454b23d	451d389b	78f061ec	7c32fb06	57ef1928
	79dcd071	39dc97f0	3a1bff42	031d364c	fef000e6	40873ef5	d0741256	649430cf
	97ef5538	3eab6a7e	b4f9cf72	9eba8257	8df0c460	5a6c49b6	ad1d9711	0f69afa2
diff	**00000000**	**83480012**	**32b4070c**	**681a1279**	**648600ad**	**00000000**	**00000000**	**00000000**

Conclusions

We exploited a particular weakness of the step transformation of FORK-256 to cryptanalyse the function. We showed

- how to find micro-collisions efficiently,
- how to look for high-level differential paths,
- how to combine those two steps to produce near-collisions efficiently and evaluated the complexity of getting collisions at $2^{126.6}$ using small amount of memory
- that using large memory we can find collisions in $2^{109.6}$,
- how to extend the attack to the full hash function (with predefined IV),
- that using truncated versions of FORK is extremely risky.

You can download our program that finds near-collisions from:

http://www.ics.mq.edu.au/~kmatus/FORK

Thank you!

Additional slides

[just in case someone asked about details]

Functions f and g

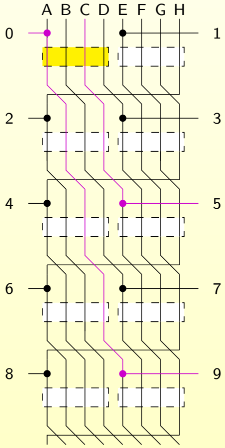
$$f(x) = x \boxplus (x^{\lll 7} \oplus x^{\lll 22}) \ ,$$
$$g(x) = x \oplus (x^{\lll 13} \boxplus x^{\lll 27})$$

Finding micro-collisions

- We can rewrite $(y \boxplus B) \oplus z = (y' \boxplus B) \oplus z'$ as $(y \boxplus B) \oplus (y' \boxplus B) = z \oplus z'$
- This means that the signed difference $\Delta^{\pm}(y \boxplus B, y' \boxplus B)$ has to have non-zero digits in those places where $\Delta^{\oplus}(z, z')$ has ones.
- There are $2^{h_w(\Delta^{\oplus}(z,z'))}$ such signed differences that "fit" into the XOR difference.
- They correspond to $2^{h_w(\Delta^{\oplus}(z,z'))}$ integer differences that may yield a micro-collision
- Integer difference is not changed by adding the constant B !

Finding high-level paths: example

- So now we are in $\mathbb{F}_2$! The whole model is $\mathbb{F}_2$-linear function L_{out} that maps input differences in M and cv_n to output differences.



Example

Input differences
$S = (A, B, C, D, E, F, G, H, M_0, \ldots, M_9)$.
For
$S = (0, 0, 1, 0, 0, 0, 0, 0,\ 1, 0, 0, 0, 0, 1, 0, 0, 0, 1)$ we have $L_{out}(S) = (0, 0, 0, 0, 0, 0, 0, 0)$.

Results of the search

Scenario	Branches	m	Differences in	active Q-structures
Pseudo-collisions	1,2,3,4	5	H_0, M_2, M_{11}	12:000, 25:000, 35:001, 41:001, 51:010
Collisions	1,2,3,4	6	M_{12}	13:000, 31:001, 40:000, 47:100, 50:000, 57:000
Pseudo-collisions	1,2,3	2	B_0, M_{12}	8:100, 24:000
	1,2,4	3	H_0, M_{11}	3:000, 51:010, 60:000
	1,3,4	3	H_0, M_2	35:001, 44:000, 51:000
	2,3,4	3	D_0, M_9	36:010, 43:000, 52:000
Collisions	1,2,3	3	M_0, M_3, M_9	1:001, 20:010, 39:100
	1,2,4	4	M_1, M_2	2:001, 9:000, 25:100, 51:000
	1,3,4	5	M_9	10:000, 39:001, 42:001 43:010, 59:000
	2,3,4	5	M_3, M_9	20:010, 27:000, 39:000 57:000, 59:010

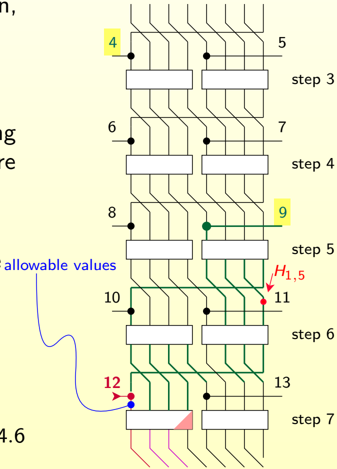
Legend: 47:100 means that the 47-th Q-structure is modelled with coefficients $(q_B, q_C, q_D) = (1, 0, 0)$.

Collisions: the principle of the attack

- Get three micro-collisions in branches 3 and 4.
  This leaves two message words M_4 and M_9 free, the rest is fixed
- Using different values of M_4 and M_9 compute branch 1 and hope that there is a single micro-collision in strand D in step 7.
- If a micro-collision there is found, compute the rest of the function and check the output difference.
  Note that the output differences have weights always ≤ 108

Collisions: the complexity of getting close hashes

1. Compute internal registers up to 7th step. Then, for each allowable value x, set $A_{1,6} = x - M_{12}$, get the corresponding $H_{1,5}$ and store the result into a hash table T.
2. For each value of M_9 compute the corresponding value of $H_{1,5}$ and look for a match in T. If there is a match, go to point 3. When all M_9 are exhausted, increment M_4 and go to point 1.
3. *Check.* If current value of M_9 leads to a single micro-collision in the thread $D_{1,6} \rightarrow E_{1,7}$ then return (M_4, M_9), else continue point 2.



Point 1: $\eta/64 = 2^{15.7}$ FORK evaluations.
Point 2: $2^{32}/64 = 2^{26}$ FORK evaluations.
Since point 3 succeeds with probability $2^{-24.6}$ we get $2^{7.4}$ solutions for a work effort of 2^{26}. Per single solution: about $2^{18.6}$ FORK evaluations.

Collisions: improving efficiency using large tables

We can use such precomputed tables to speed up the algorithm.

- In branch 3 we can use one to control the thread $C_{3,1} \to D_{3,2}$ through M_{10}
- In branch 1 we use a family of such tables T_a for some (best) allowable values a. For a fixed a, $T_a(G_{1,4}, M_{11} + E_{1,5})$ returns the value of M_9 that gives us $A_{1,6} = a - M_{12}$
- For that allowable value a we get a micro-collision with probability $2^{-8} \sim 2^{-9}$. So after 512 lookups we expect to get a micro-collision.
- If 1 look-up = 1 op (e.g. ADD) then this takes 1/2 FORK and we have $\approx 3/2$ FORK per single solution.