Improved Fast Syndrome Based Cryptographic Hash Functions

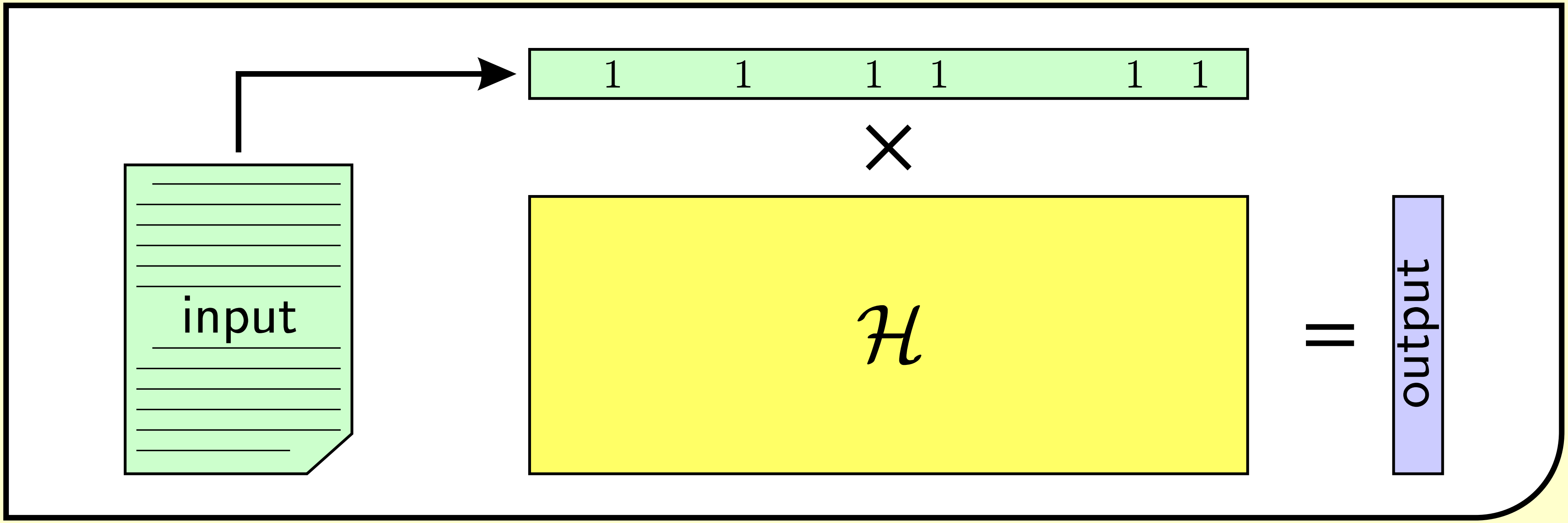
Matthieu Finiasz, Philippe Gaborit,
and Nicolas Sendrier

The Original FSB Hash Function
[Augot, Finiasz, Sendrier - Mycrypt 05]

- Based on the Merkle-Damgård construction
  - ▷ requires a collision resistant compression function.
- Provably secure:
  - ▷ collision search on the compression function requires to solve an instance of an NP-complete problem,
  - ▷ inversion too.
- These problems have been well studied
  - ▷ similar to those of the McEliece cryptosystem.

The core of the function is a binary $r \times n$ matrix $\mathcal{H}$.

- the input (data + chaining) is converted into a binary vector of weight w and length n.
- this vector is multiplied by $\mathcal{H}$ to obtain r bits of output.



The core of the function is a binary $r \times n$ matrix $\mathcal{H}$.

- ▷ the input (data $+$ chaining) is converted into a binary vector of weight w and length n.
- ▷ this vector is multiplied by $\mathcal{H}$ to obtain r bits of output.

- ▶ Constant weight encoding uses regular words

$$\boxed{10\text{——}0 \,\vdots\, 0\text{-}010\text{-}0 \,\vdots\, 0\text{——}01 \,\vdots\, 0\text{-}010\text{-}0 \,\vdots\, 010\text{——}0}$$

  - ▷ much faster than optimal encoding.

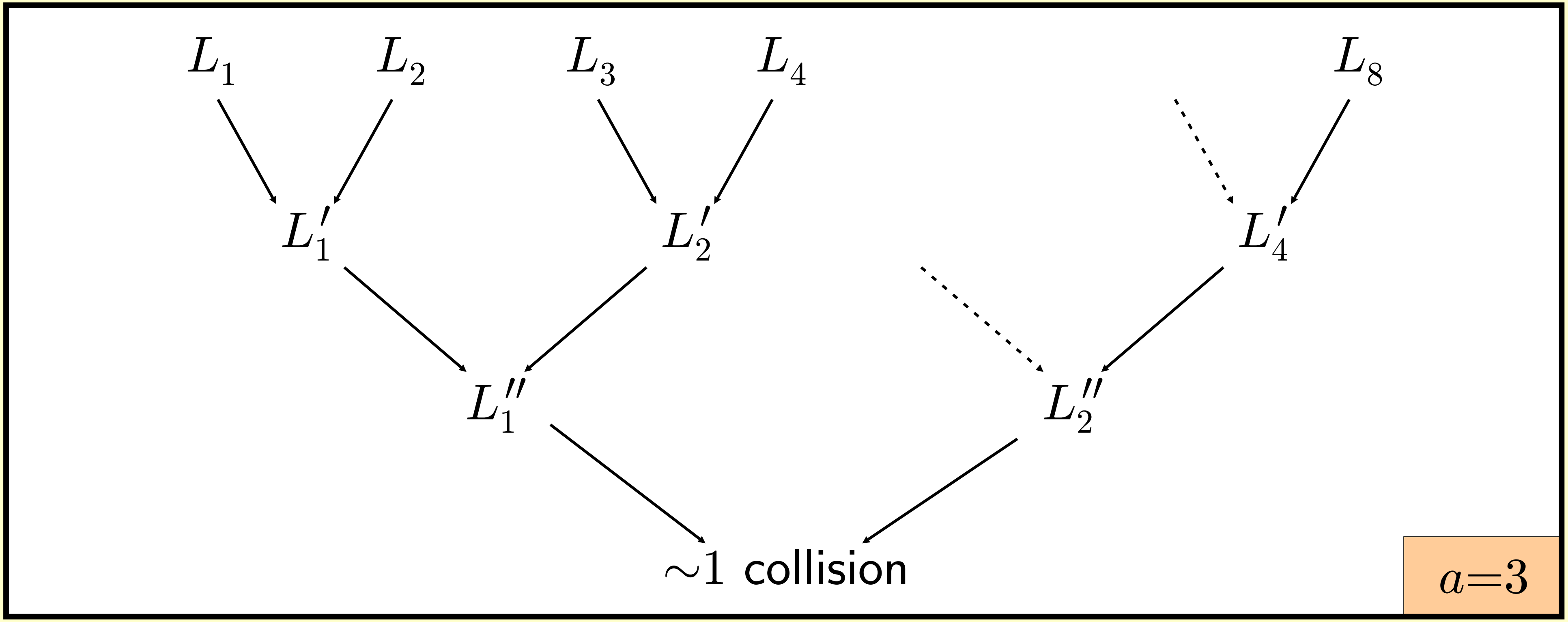
- Inversion:
  - ▷ find a vector of weight w with given image
    $\longrightarrow$ exactly Syndrome Decoding.
- Collision search:
  - ▷ find a vector of weight $\leq 2w$ with null image
    $\longrightarrow$ again Syndrome Decoding.

- With regular words, both of these problems are still NP-complete. [Augot, Finiasz, Sendrier - Mycrypt 05]

- The best attack uses Wagner's generalized birthday technique [Crypto 2002].

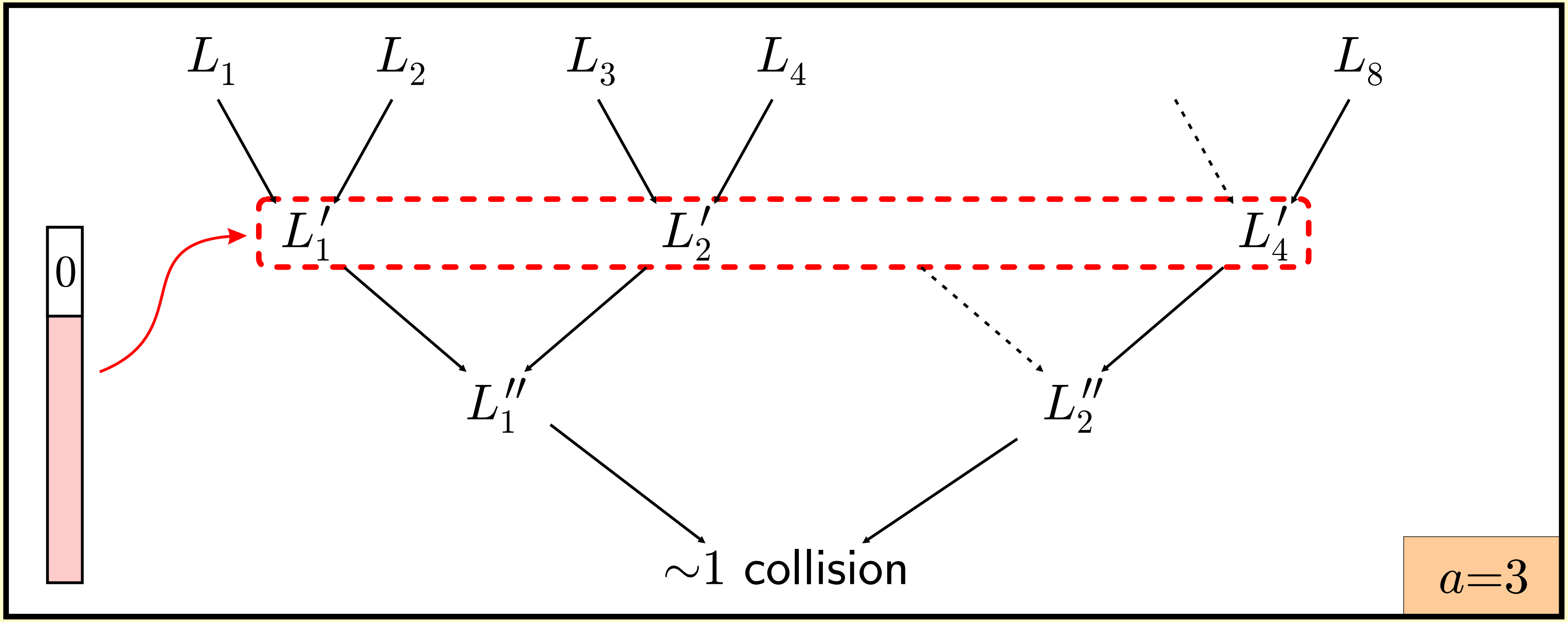
- We look for $2w$ columns of $\mathcal{H}$, XORing to 0.
  - Birthday technique:
    - → build 2 lists of XORs of w columns.
    - → complexity: $\mathcal{O}\left(2^{\frac{r}{2}}\right)$.
  - Wagner's generalized birthday technique:
    - → build 2^a lists of XORs of $\frac{w}{2^{a-1}}$ columns.
    - → complexity: $\mathcal{O}\left(2^{\frac{r}{a+1}}\right)$.

Wagner's Generalized Birthday Technique



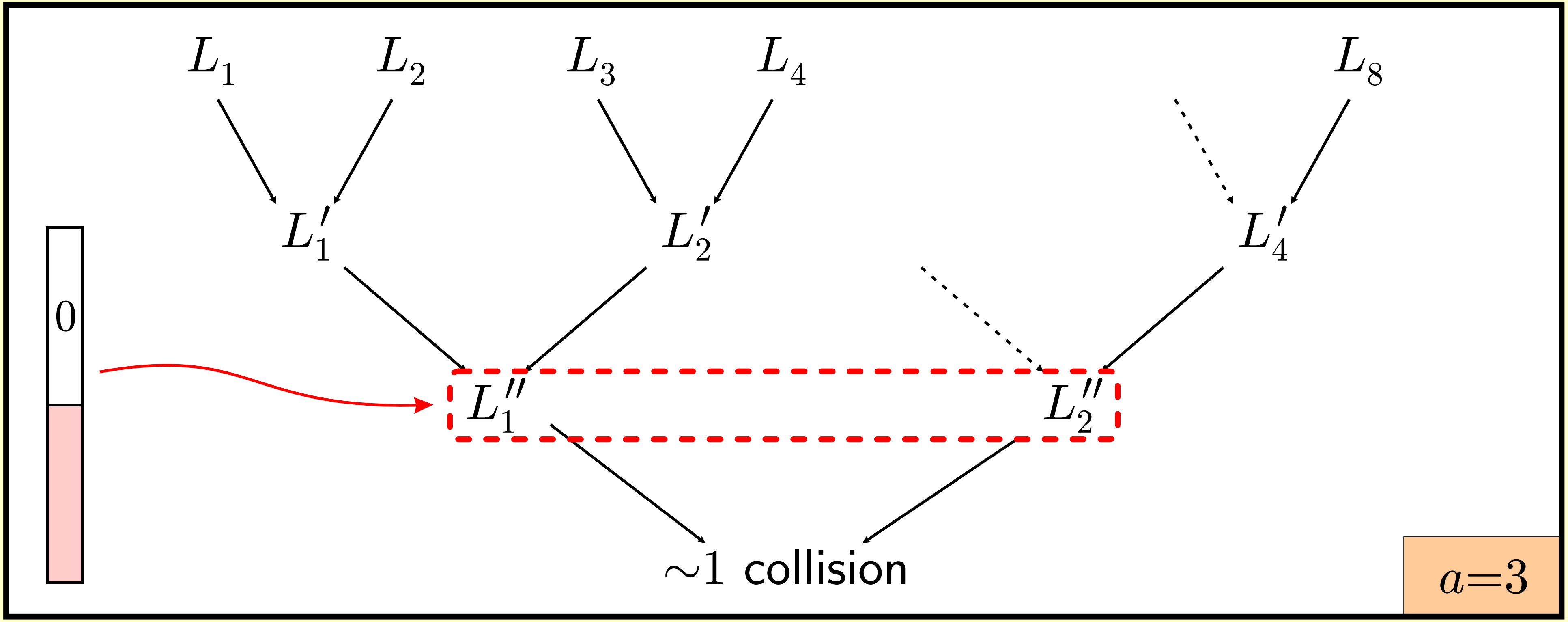
- L_i are lists of $2^{\frac{r}{4}}$ elements
  - each element is the XOR of $\frac{w}{4}$ columns.

Wagner's Generalized Birthday Technique



- L'_i are lists of $2^{\frac{r}{4}}$ elements
  - each element is the XOR of $\frac{w}{2}$ columns.
  - each element starts with $\frac{r}{4}$ zeroes.

Wagner's Generalized Birthday Technique



- L''_i are lists of $2^{\frac{r}{4}}$ elements
  - each element is the XOR of w columns.
  - each element starts with $\frac{r}{2}$ zeroes.

- Efficient parameters always allow to choose $a = 4$ in Wagner's technique,
  - for a security of 2^{80} we need $r = 400$.
- The choice of w and n is flexible:
  - tradeoff between the matrix size and the hash speed.

Example parameters:

$$r = 400, w = 85, n = 256 \times w = 21760.$$

$\rightarrow$ speed: 70Mbits/s, matrix size: 1MB.

- The original FSB construction is:
  - practical,
  - quite fast,
  - provably collision resistant.

- However it suffers from a few drawbacks:
  - the output size is too large,
  - the block size is quite large,
  - the matrix is large,

    $\longrightarrow$ does not fit in a CPU cache.

Improvements to the Original FSB

Addition of a Final Transform

- For a security against collision of 2^λ operations, one expects a hash of 2λ bits:
  - requires to add a final compression round.

- Used in many other constructions.
  - If the final compression is collision resistant, then the combination is also collision resistant.
  - What about provable security?
    $\longrightarrow$ Must the last round be provably collision resistant?
  - Use the same construction with other parameters?

Addition of a Final Transform

- Suppose we used a linear transform L from r to r' bits:
  - compute $\mathcal{H}' = L \times \mathcal{H}$ and use Wagner's attack on $\mathcal{H}'$.
    - $\longrightarrow$ The complexity of decreases to $2^{\frac{r'}{a+1}}$.

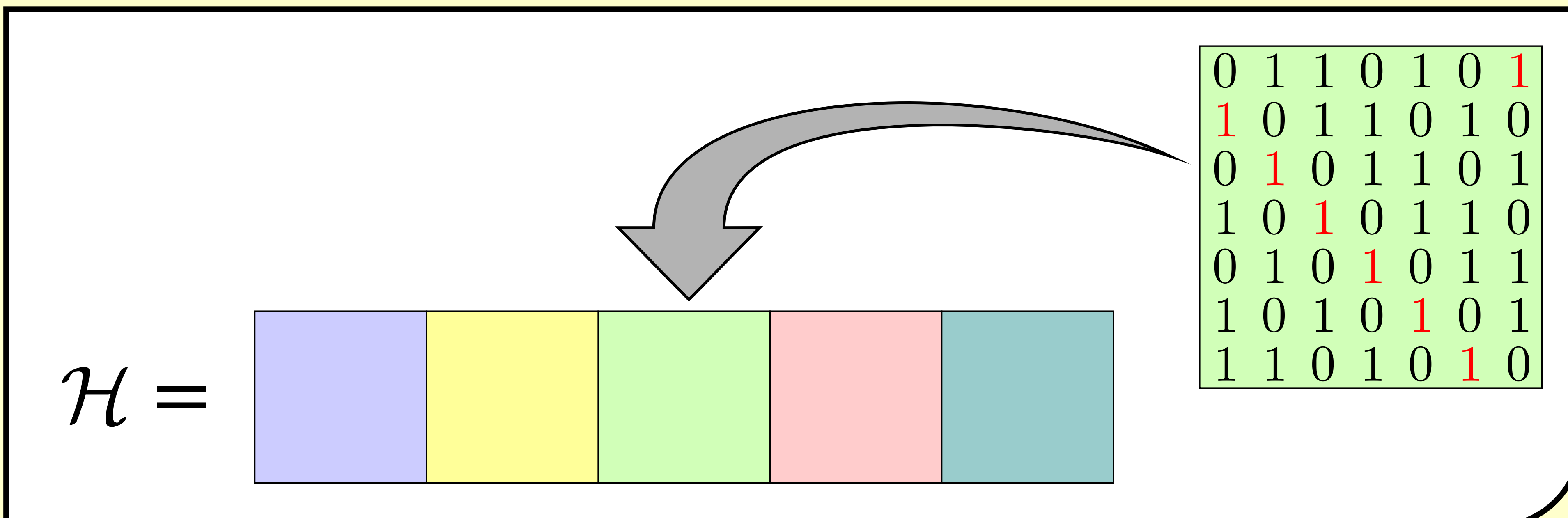
If the final transform is non-linear this won't be possible.

- We propose to use another hash function like Whirlpool:
  - it is designed to be as much as possible non-linear,
  - we loose provable security,
  - chances are that attacks on Whirlpool won't affect our construction.

Use of a Quasi-cyclic Matrix

Basic idea

- The matrix $\mathcal{H}$ is too large:
  - store a small amount of data and generate $\mathcal{H}$ from it,
  - must fit in the CPU cache
    - $\longrightarrow$ generation is done at runtime.
- Use a quasi-cyclic (QC) matrix:



Use of a Quasi-cyclic Matrix

Basic idea

- The matrix $\mathcal{H}$ is too large:
  - ▷ store a small amount of data and generate $\mathcal{H}$ from it,
  - ▷ must fit in the CPU cache
    $\longrightarrow$ generation is done at runtime.
- Use a quasi-cyclic (QC) matrix:
  - ▷ storing the first line is enough,
  - ▷ other lines are blockwise cyclic shifts,
  - ▷ cyclic shifts can be efficient
    $\longrightarrow$ no need to rebuild $\mathcal{H}$ completely before hashing.

- Syndrome Decoding of a QC matrix is NP-complete
  - not proven for regular words.
- QC codes have been extensively studied:
  - no known efficient decoding algorithm,
  - any attack would yield such a decoding algorithm.
- For some specific sizes the outputs are proven to be uniformly distributed.

- From a practical point of view:
  - no clue how to improve Wagner's birthday technique.

Implementation

					standard FSB			new improved variant		
secu.	r	w	n	$\frac{n}{w}$	size of $\mathcal{H}$	time	cyc./byte	size of $\mathcal{H}$	time	cyc./byte
64	512	512	131 072	256	8 388 608	28.8s	390.6	16 384	6.6s	89.3
	512	450	230 400	512	14 745 600	43.1s	587.9	28 800	12.1s	165.1
	1 024	2^{17}	2^{25}	256	2^{32}	–	–	4 194 304	25.0s	339.8
80	512	170	43 520	256	2 785 280	37.7s	517.0	**5 440**	**20.5s**	**281.1**
	512	144	73 728	512	4 718 592	42.6s	581.6	9 216	17.6s	239.8
128	1 024	1 024	262 144	256	33 554 432	48.6s	669.6	**32 768**	**8.9s**	**121.0**
	1 024	904	462 848	512	59 244 544	72.4s	989.9	57 856	27.2s	371.2
	1 024	816	835 584	1 024	106 954 752	53.4s	727.6	104 448	11.8s	162.6
64	MD5				best known implementations from					3.7
80	SHA-1				[Nakajima, Matsui - Eucrocrypt 2002]					8.3
128	SHA-256									20.6

- Our implementation is not optimised:
  - we obtain a speed of 180Mibts/s with 128 bits security.

Conclusion

We propose a new variant of the FSB hash function:

- no large matrix to handle,
- standard output size,
- twice as fast as the original construction,

- not completely proven to be collision resistant:
  – use of regular words,
  – use of the final compression transform.