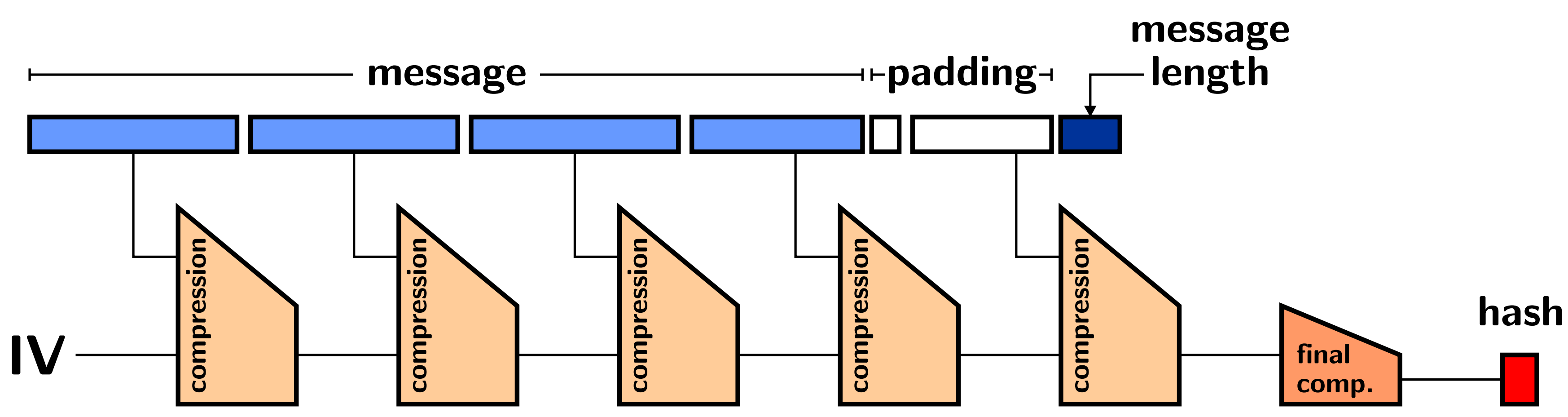
# SHA-3 Proposal: FSB

D. Augot, M. Finiasz, P. Gaborit,
S. Manuel, and N. Sendrier

# High Overview of FSB

◇ FSB uses the Merkle-Damgård construction (chaining and padding), with a large internal state:
→ it uses a final compression function.

◇ the main compression function uses a one-way function from coding theory:
→ security reduction for inversion and collision search.

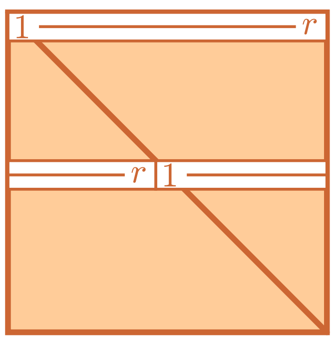


- The compression function of FSB is made of two steps:
  - a non-linear bijective step,
  - a linear compression step.

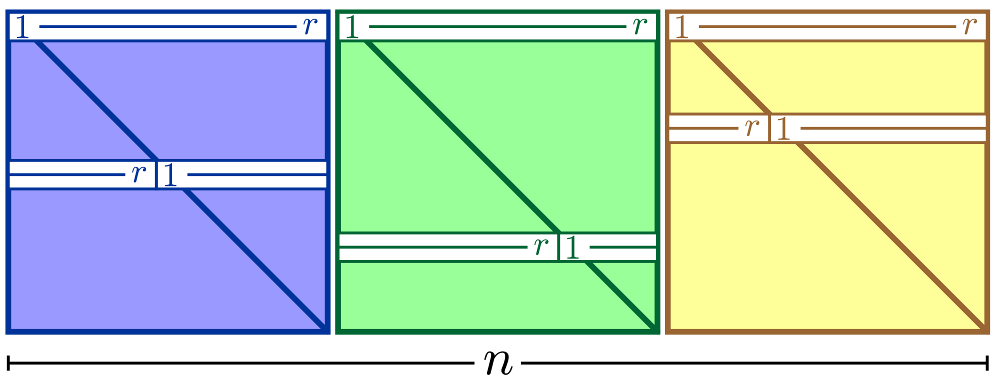
- First the $s$ input bits are transformed in a binary vector of length $n$ and Hamming weight $w$:
  - for efficiency we use regular words.
- Then this vector is multiplied by a binary matrix $\mathcal{H}$
  - $w \ll n$ so this is simply the XOR of $w$ columns of $\mathcal{H}$.

In practice $\mathcal{H}$ is a truncated quasi-cyclic matrix

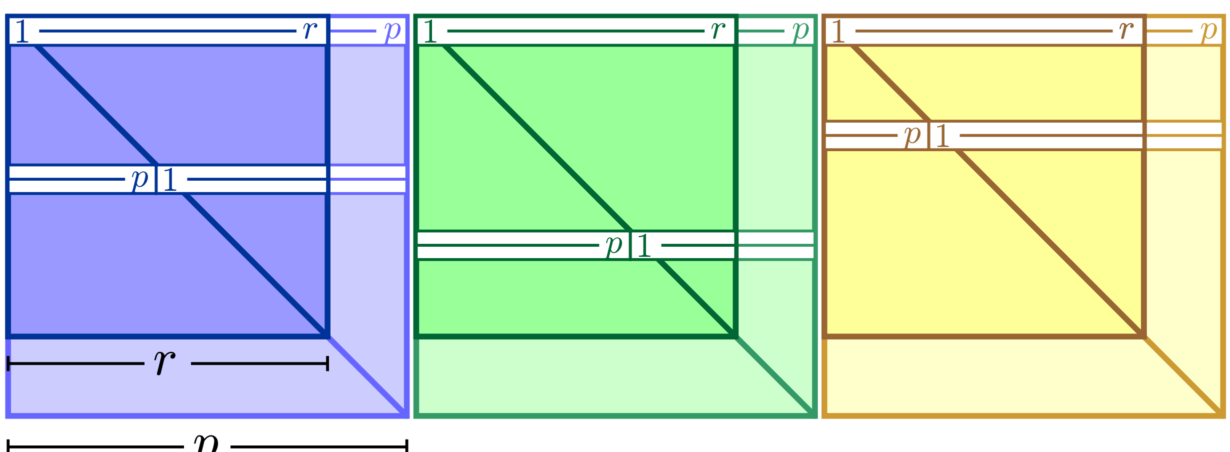
**circulant**




**quasi-cyclic**




**truncated quasi-cyclic**



In practice $\mathcal{H}$ is a truncated quasi-cyclic matrix

- $\mathcal{H}$ is described by its first line: $\frac{n}{r}$ vectors of $p$ bits.
- columns of $\mathcal{H}$ are truncated cyclic shifts of these binary vectors.
- which vectors to choose and how much they should be shifted depends on the input:
  - $w$ indexes are derived from 13 or 14 input bits each,
  - 8 IV/chaining bits and 5 or 6 message bits,
  - the $i$-th index is taken in the interval $\left[i\frac{n}{w}, (i+1)\frac{n}{w}-1\right]$,
  - the $w$ indexes correspond to the $w$ columns to XOR.

The best algorithms that can be used to attack FSB are:

Generalized birthday algorithm

- best algorithm for inversion and second preimage,
- requires a lot of memory.

Information set decoding

- best algorithm for collision search,
- yields strong constraints on the choice of $r$ and $w$.

Proposed parameters have been chosen according to these algorithms, plus a security margin.

Inverting the compression function requires to find $w$ columns of $\mathcal{H}$ which XOR to a target vector.

▷ this is an instance of the syndrome decoding problem,

▷ this problem is NP-complete for random matrices, but also for truncated quasi-cyclic matrices,

▷ well chosen values of $p$ and $r$ give supposedly hard instances of the problem.

Collisions require $2w$ columns of $\mathcal{H}$ which XOR to 0.

▷ also an instance of the syndrome decoding problem,

▷ an "easier" instance in practice.

An important point is that these reductions are tight.

| adversary | best attack | reduction |
|---|---|---|
| collision | $\text{ISD}(n, r, 2w) \times 1$ | $\text{CSD}(n, r, 2w)/1$ |
| preimage | $\text{GBA}(n, r, w) \times 1$ | $\text{CSD}(n, r, w)/1$ |
| second-preimage | $\text{GBA}(n - w, r, w) \times 1$ | $\text{CSD}(n - w, r - w, w)/1$ |

ISD = Information set decoding

GBA = Generalized birthday algorithm

CSD = Computationnal syndrome decoding.

One call to the adversary solves the CSD problem, one call to ISD/GBA is enough to build an adversary.

# Final Compression Function

Few constraints apply to the final compression function.

it must not weaken the main compression function

▷ any linear function is bad

simple truncation is impossible.

it does not require collision resistance/one-wayness

▷ collisions on the final compression do not directly lead to collisions on FSB

Cryptographers and the NIST need to be convinced...

▷ anything too simple should be avoided.

# Final Compression Function

We propose to use Whirlpool [Rijmen, Barreto 2004]:

The $r$-bit output of the main compression function is input as an $r$-bit message to Whirlpool

▷ the final output is a truncated Whirlpool hash.

This is a safe choice, not an efficiency oriented choice:

▷ Whirlpool is highly non-linear,

▷ we are confident that it is a secure hash function,

▷ attacks on Whirlpool would probably not affect our construction.

The main compression functions is very simple:

- ▷ shift and XOR $w$ times some vectors
  with precomputed shifts, only XORs are required.
- ▷ parameters of FSB are quite large
  the XORs are expensive: 250 to 500 cycles/byte.

The description of FSB is large:

- ▷ 2 millions bits from digits of $\pi$ define the vectors
  this is a problem for constrained environments,
- ▷ using pseudo-random data could improve this but would loosen the security reduction.

The main interest of FSB is its compression function:

- ◇ inversion and collision search reduce to hard problems,
- ◇ it is slow, but much faster than most "similar designs,"
- ◇ it is very simple to describe/implement
  only very basic operations are used,
- ◇ the description of FSB is large as "random bits" are needed.

Security reduction to hard problems comes at a cost, but it can be practical in many contexts.