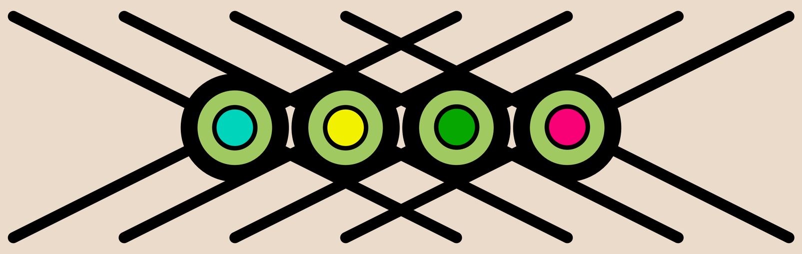
Verified tail-recursive calculation of Classic McEliece control bits

Wrenna Robson ◈ University of Bristol

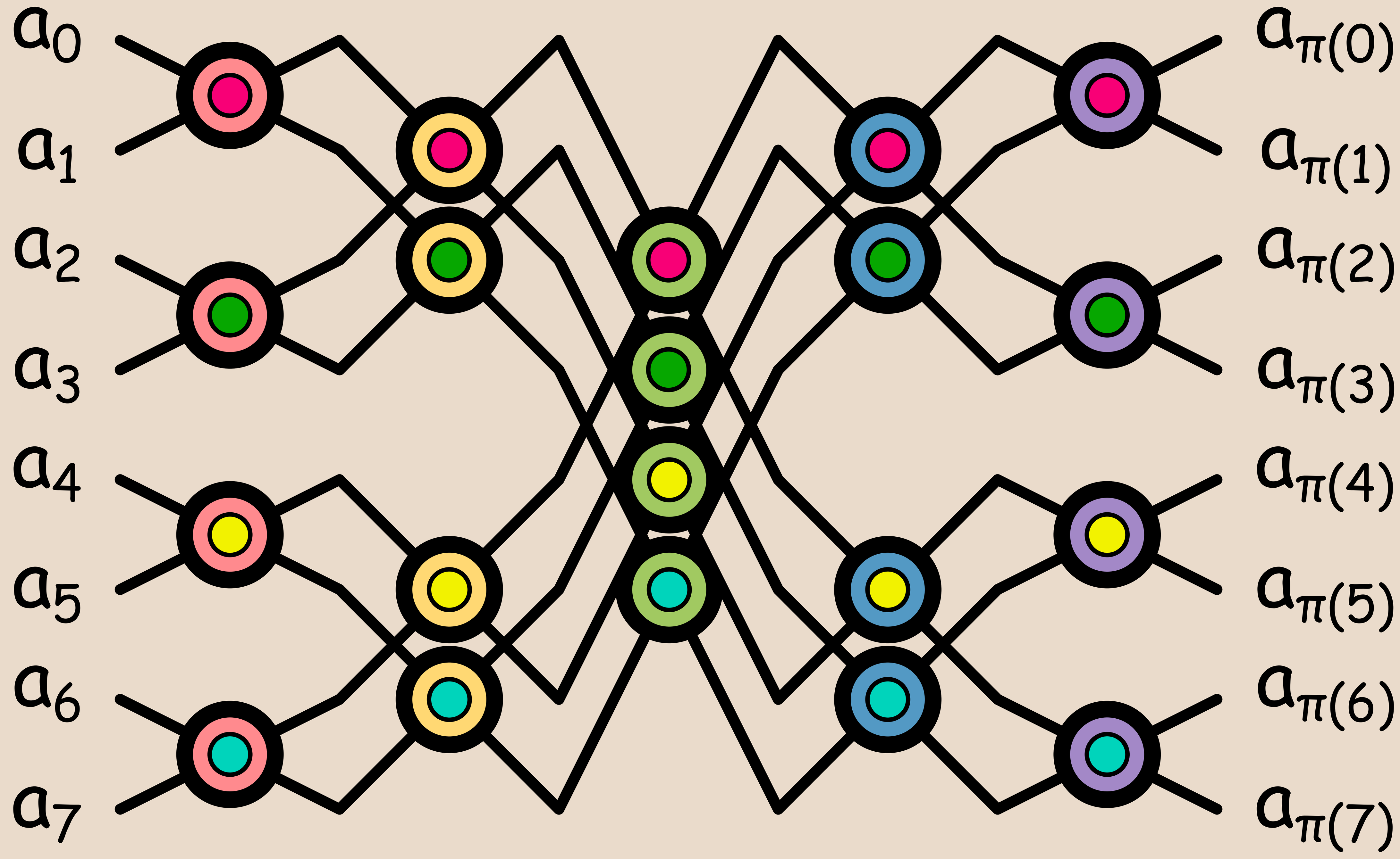
With thanks to Dr Rachel Player and Dr Martin Brain



A sequence of 2^n bits represents a permutation on 2^{n+1} elements. These correspond to conditional flips in a particular bit.

$$X_{n,i} : \{0,1\}^{2^n} \to Sym(2^{n+1})$$

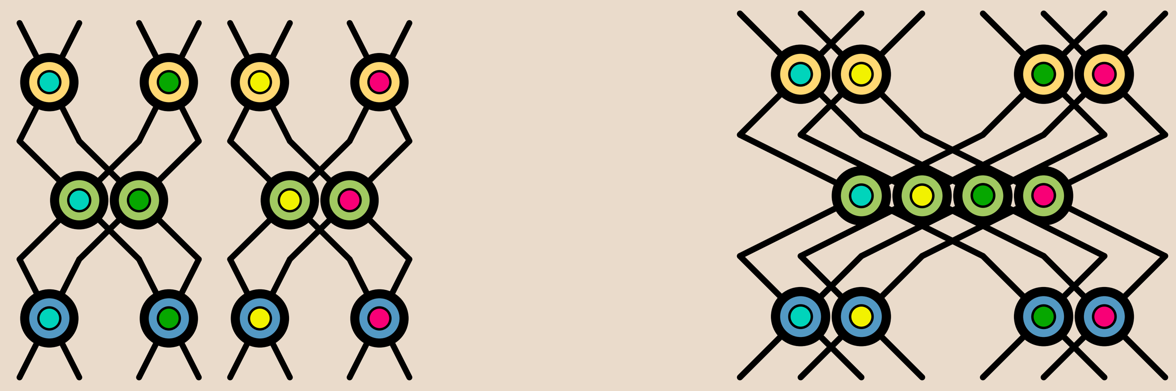
We seek to represent all permutations of 2^{n+1} elements by a sequence of these "conditional swaps", to avoid side-channel leaks.



Using control bit networks

- Permutations are represented as networks of control bits controlling exchanges between fixed elements.
- The Classic McEliece post-quantum key encapsulation mechanism uses these networks as part of its decryption keys.

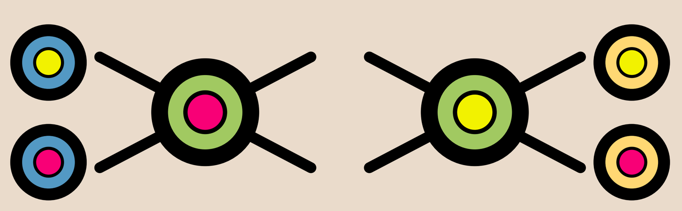
An isomorphism exists between paired permutations on 2^n elements and permutations on 2^{n+1} elements which preserve parity.



$$M_n : Sym(2^n) \times Sym(2^n) \to Sym(2^{n+1})$$

$$C_{n,i} : Sym(2^{n+1}) \to \{0,1\}^{2^n} \times Sym(2^{n+1}) \times \{0,1\}^{2^n}$$

If $C_{n,i}(\pi) = (F, M, L)$, then $\pi = X_{n,i}(F) \cdot M \cdot X_{n,i}(L)$



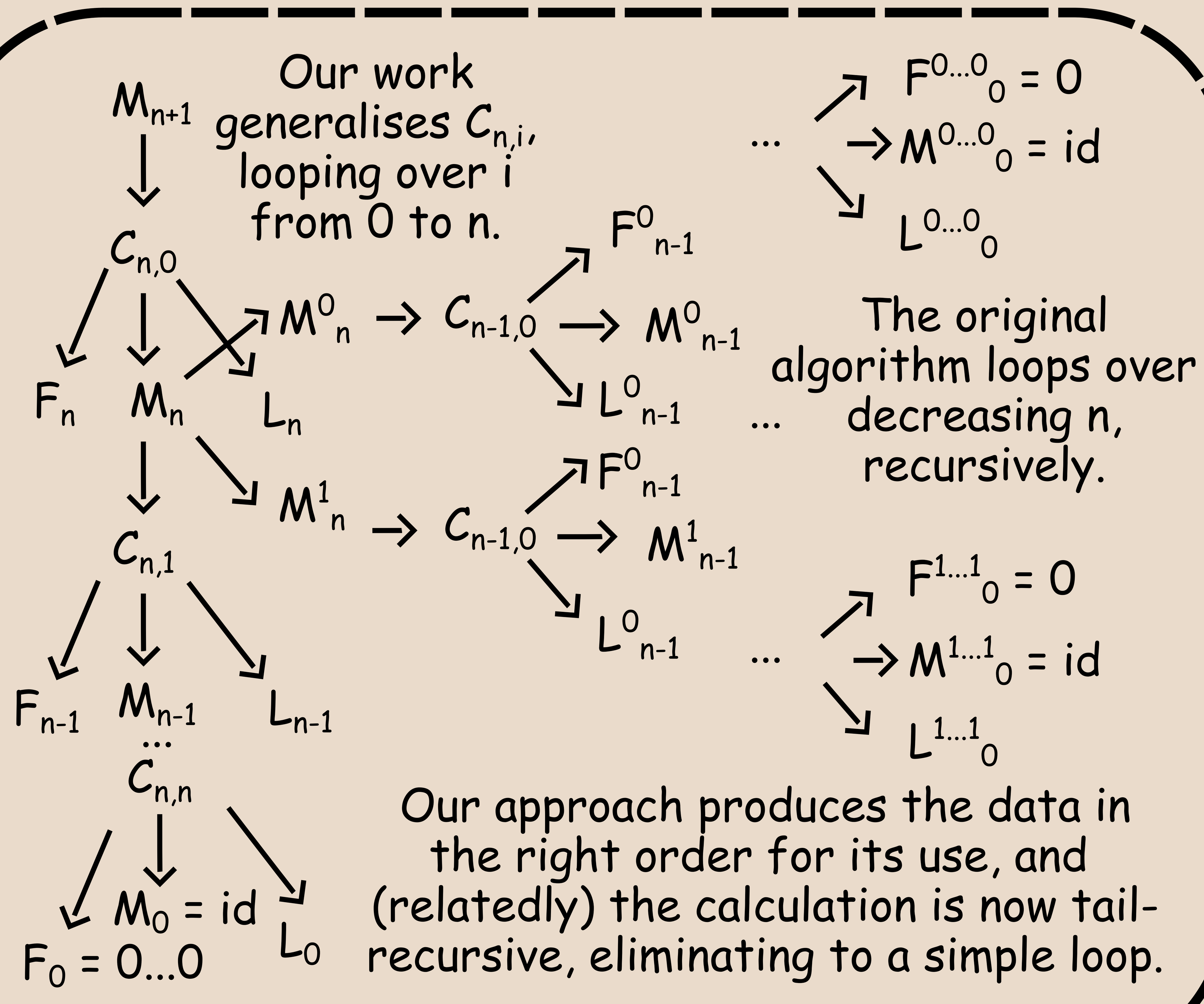
If $C_{n,i}(\pi) = (F, M, L)$, and π bit-invariant for all bits $< i$, then the first 2^i bits of F will be 0, and M is bit-invariant for all bits $\leq i$.

Algorithms for calculating control bits

- There is a key component of both algorithms which we call $C_{n,i}$.
- Computes "outer layer" for valid inputs
- This process is computationally taxing but this can be mitigated.

Original approach

- The Classic McEliece specification uses only $C_{n,0}$ (which it is easier to prove the properties of).
- The isomorphism defined by M_n is used to split "M" into two permutations of half the size, and recursively proceeds.
- The results of the recursion must be interleved after they complete.



Our approach produces the data in the right order for its use, and (relatedly) the calculation is now tail-recursive, eliminating to a simple loop.

Our contributions

- A tail-recursive variant algorithm for calculating the Classic McEliece control bits.
- A verified proof of the correctness of our tail-recursive variant in the Lean theorem prover and functional language, linked to a performant implementation.
- A formal proof of the correctness of the original algorithm, linked to a functional though non-performant implementation, building on existing formalisation work by Dan Bernstein.