Perfect Block Ciphers With Small Blocks

Louis Granboulan[1,2] Thomas Pornin[3]

[1]École Normale Supérieure

[2]EADS

[3]Cryptolog International

March 28th, 2007

Outline

Small blocks

Usual block ciphers operate over blocks of 64 bits or more.
Some applications need shorter blocks (e.g. generation of unique pseudo-random numbers in a short range).

Security issues with block size

Usual block cipher designs (e.g. Feistel scheme) build ciphers from a restricted subset of the permutations over the message space (a Feistel scheme can only be an even permutation). This is *tolerated* thanks to the huge block size.

Non-binary alphabets

Usual block ciphers use messages consisting of bits.
Some applications need messages using another alphabet (e.g. generation of unique *decimal* pseudo-random numbers).

10 is more complex than 2

Decimal alphabets are challenging:

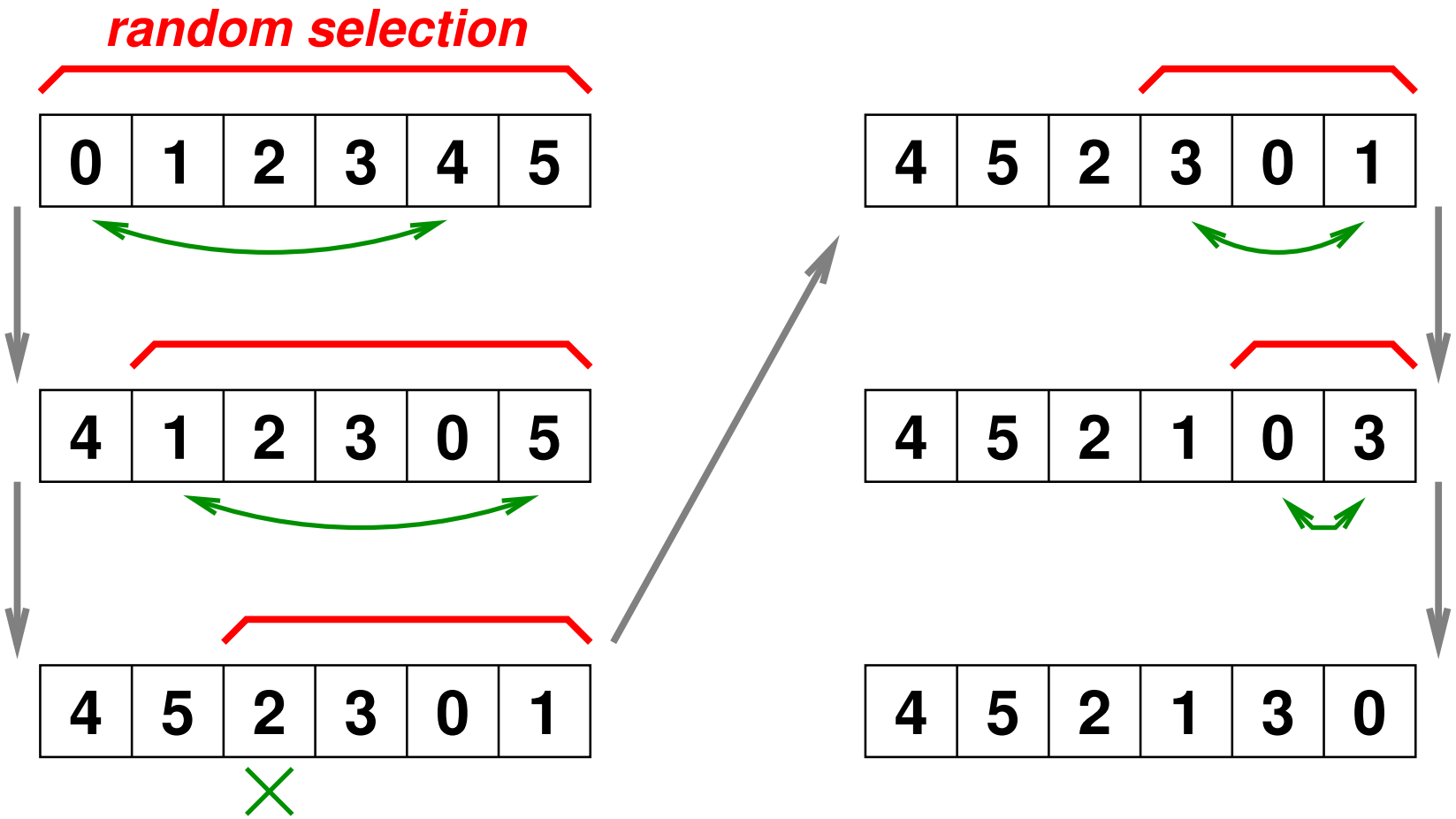
- Several ring structures can be applied to a set of size 10.
- There is no field of size 10.
- There are several types of differentials.

What this paper is about

We describe PERMUTATOR, which is an algorithm for selecting randomly and uniformly a permutation over a set of n elements:

- n is arbitrary;
- the algorithm input is a seekable stream of random bits;
- if the stream is truly random, then all the $n!$ permutations have an equal chance of being selected;
- the permutation and its inverse can be "efficiently" evaluated.

Knuth Shuffle



Knuth Shuffle

- The permutation is defined by an array of size n.
- We need $n-1$ random selections of integers between 0 and r, where r goes down from $n-1$ to 1.
- Cost:
  - $O(n \log n)$ space
  - $O(n \log n)$ CPU (n selections of integers of size $\log n$) for init, then $O(\log n)$ (array lookup) for each evaluation

Applicability

The "Knuth shuffle" solves our problem only for very small values of n (e.g. $n \leq 10000$).

Partial evaluation

Idea

Use a shuffle algorithm, but apply it *partially*: for a given input x, compute only the parts which may have an influence over $\phi(x)$.

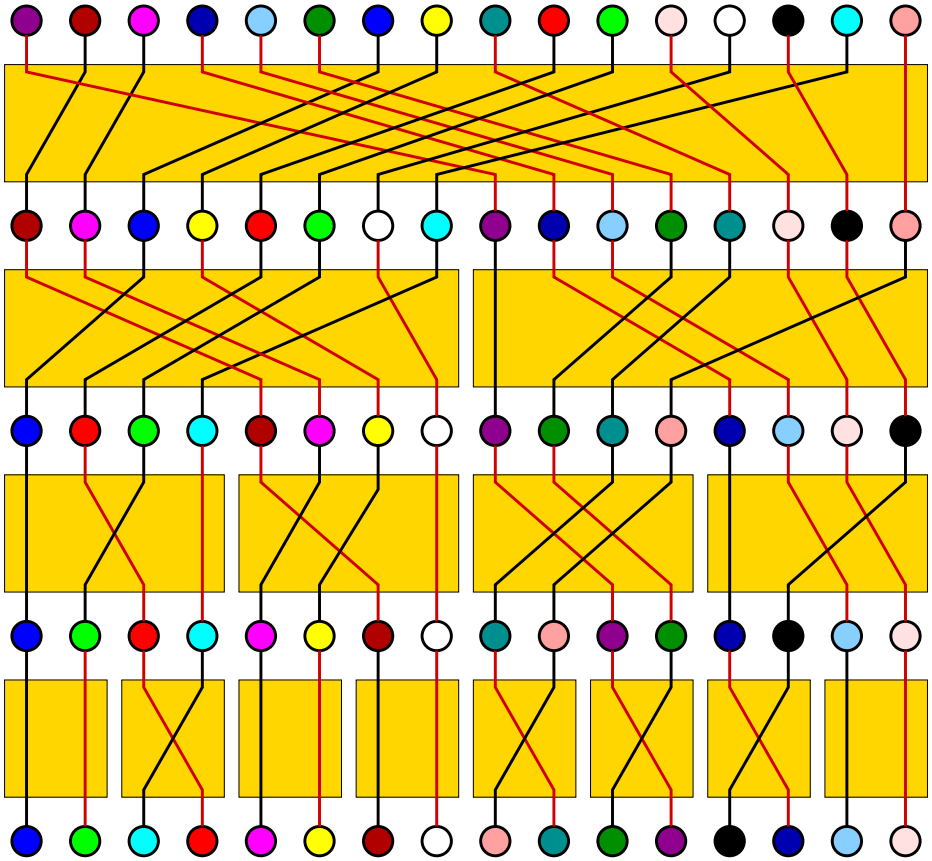
The Knuth shuffle is *not* adequate for partial evaluation: on average, $n/2$ random selection events *may* affect $\phi(x)$.

Overview

"PERMUTATOR" is a shuffle expressed as a binary tree of "SPLITTER" operations. To evaluate $\phi(x)$, one needs follow only one path from the root in that tree ($\log n$ nodes).
"SPLITTER" is implemented as a binary tree of "REPARTITOR" operations. For a given x, we need follow only one path in that tree, for each for the considered SPLITTER nodes. (at most $\log n$ sub-nodes).
"REPARTITOR" is a random selection event, using the hypergeometric distribution, which has cost $O(\log n)$.
Cost:

- $O(\log n)$ space (tree walking, no backtrack needed)
- $O((\log n)^3)$ CPU for each evaluation

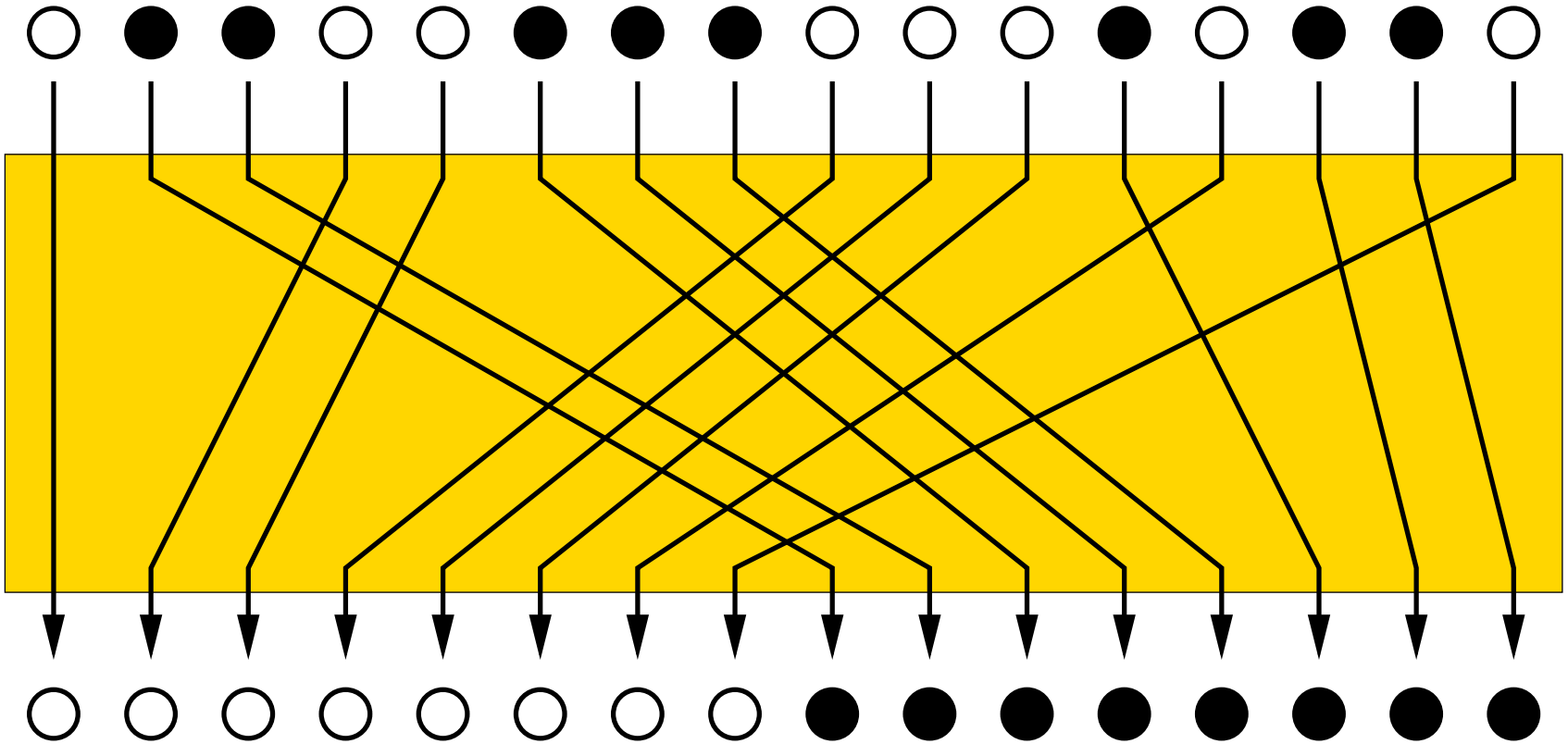
Permutator: a tree of Splitter



Splitter

Splitter is an elementary permutation which splits elements into two groups: each of the n elements goes either into the left half (size $\lfloor n/2 \rfloor$) or the right half (size $\lceil n/2 \rceil$).
Within each half, the element ordering is preserved.
Splitter selects $\lfloor n/2 \rfloor$ "white" elements, which go into the left half; the remaining (black) elements go into the right half.

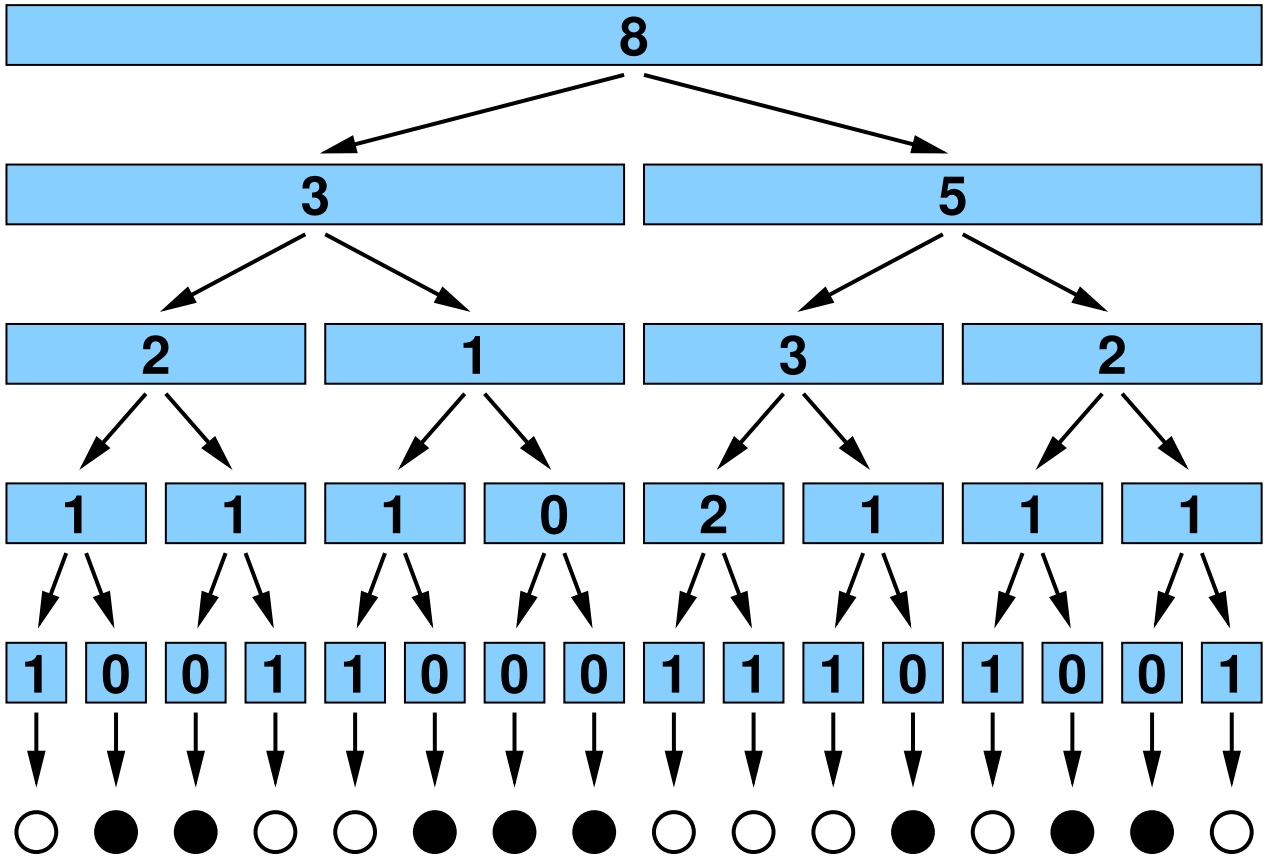
Splitter



Splitter

Each Splitter works over n elements, and must "extract" p white elements. It invokes Repartitor, which tells how many of these white elements come from the left half.
Splitter then invokes itself recursively on both halves. For partial evaluation, only one half is considered.

Splitter: a tree of Repartitor



Repartitor

Repartitor is given n elements, among which p are white (and $n-p$ are black). Repartitor chooses how many of those p white elements come from the $\lfloor n/2 \rfloor$ first elements.
Repartitor, when used in Permutator, selects a uniform permutation if it returns the value u following the hypergeometric distribution:

$$P(u=k) = \frac{\binom{a}{k}\binom{n-a}{p-k}}{\binom{n}{p}}$$

where $a = \lfloor n/2 \rfloor$.

Direct sampling

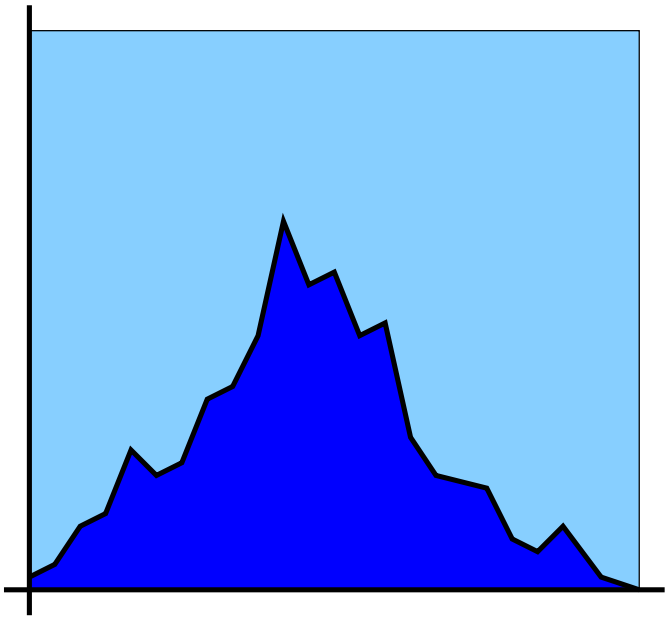
For small values of p, REPARTITOR uses a direct sampling algorithm: the p white elements are directly selected:

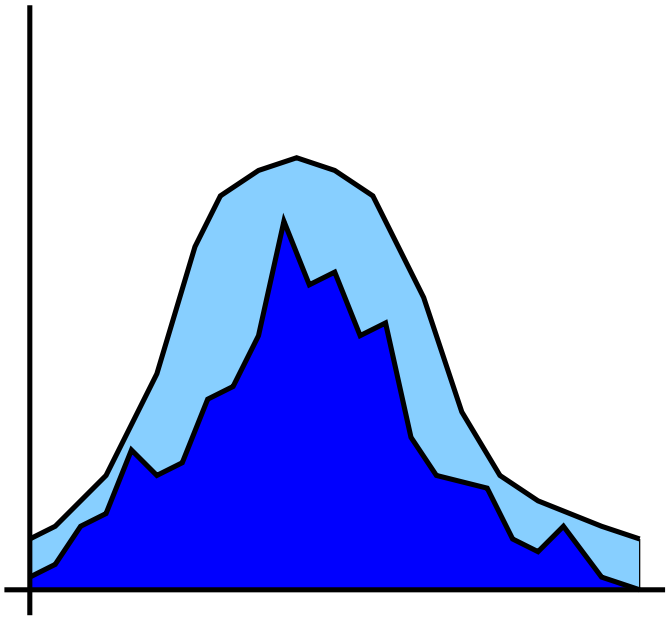
1. $n_1 \leftarrow a$, $n_2 \leftarrow n - a$
2. if $p = 0$, then return $a - n_1$
3. select randomly r between 0 and $n_1 + n_2 - 1$ (inclusive)
4. if $r < n_1$, then $n_1 \leftarrow n_1 - 1$, else $n_2 \leftarrow n_2 - 1$
5. $p \leftarrow p - 1$
6. go to step 2

Limitations

Cost is linear in p. We use this method for $p \leq 10$.

Rejection sampling





Rejection sampling

Principle

We select random points until we find one which lies below the target distribution. The process is hastened by using a carefully chosen area for random point selection (a scaled "easy" distribution).

For REPARTITOR, we use the Cauchy-Lorentz distribution:

$$CL_{\mu,\nu}(x) = \frac{1}{\pi}\left(\frac{\sqrt{\nu}}{(x-\mu)^2+\nu}\right)$$

where $\alpha = \lfloor n/2 \rfloor / n$, $\mu = \alpha p$ and $\nu = 2\alpha(1-\alpha)p$.

Rejection sampling

Precision and performance

- Our method implies using floating point values with arbitrarily extended precision.
- The sampling is exact (the hypergeometric distribution is followed exactly) but slow.

A *slightly biased* REPARTITOR can be tolerated, provided that:

- the proportion of permutations which can no longer be selected is sufficiently small;
- for those permutations which can be selected, their probability of selection is sufficiently close to the theoretical $1/n!$.

Security model

Uniform random selection of a permutation among the $n!$ possible permutations is "the best possible". We define a security model which illustrates that PERMUTATOR fares better than previously known constructions:

- The attacker has unlimited computing power.
- The attacker is given a black box implementing PERMUTATOR with a secret random stream.
- The attacker may perform up to $n-2$ encryption or decryption queries (adaptively).
- The attacker must then predict the output for an input x which he has not seen yet.

Security model

The attacker is an $(n-2)$-*limited adaptive distinguisher* in the model of *super-pseudorandomness* (terminology from Luby and Rackoff).
With a random permutation, probability of success is at most $1/2$.
With a Feistel scheme, probability of success is 1: a Feistel scheme implements an *even* permutation.

Small blocks

The usual security models are not adequate for small blocks: the attacker can plausibly explore a substantial part of the code book.

Security issues

- The random stream is the output of a PRNG: the security of PERMUTATOR depends on the output of the PRNG.
- The REPARTITOR implementation is vulnerable to side channel attacks.
- The tree walk algorithm is potentially vulnerable to timing attacks if n is not a power of 2.

Side channel

A usable (i.e. not too slow) version of PERMUTATOR will use a different REPARTITOR with its own issues with regards to side channels.

Conclusion

- We presented a novel design for building a secure block cipher from a secure PRNG.
- Our construction works over arbitrary input domains, not necessarily bit strings of a fixed length.
- A better method for Repartitor is needed to achieve industrial applicability.