Randomized stopping times and provably secure pseudorandom permutation generators *

Michal Kulis[1], Pawel Lorek[2], and Filip Zagorski[1]

[1] Wroclaw University of Science and Technology
Faculty of Fundamental Problems of Technology
Department of Computer Science

[2] Wroclaw University
Faculty of Mathematics and Computer Science
Mathematical Institute

Abstract. Conventionally, key-scheduling algorithm (KSA) of a cryptographic scheme runs for predefined number of steps. We suggest a different approach by utilization of randomized stopping rules to generate permutations which are indistinguishable from uniform ones. We explain that if the stopping time of such a shuffle is a *Strong Stationary Time* and bits of the secret key are not reused then these algorithms are immune against *timing attacks*.
We also revisit the well known paper of Mironov [16] which analyses a card shuffle which models KSA of RC4. Mironov states that expected time till reaching uniform distribution is $2nH_n - n$ while we prove that $nH_n + n$ steps are enough (by finding a new strong stationary time for the shuffle).
Nevertheless, both cases require $O(n \log^2 n)$ bits of randomness while one can replace the shuffle used in RC4 (and in Spritz) with a better shuffle which is optimal and needs only $O(n \log n)$ bits.
Our analysis gives also insights to the (in)security of Spritz stream cipher.

Keywords: Pseudo-random permutation generator, Markov chains, mixing time, stream cipher, timing-attacks.

1 Introduction

The applicability of card shuffles to cryptography was noticed many years ago by *e.g.,* Naor [18] for Thorp shuffle. The shuffles can be categorized into two groups. The first one are the *oblivious* shuffles, meaning that the trajectory of a card during the shuffle can be traced without tracing trajectories of other cards. Thus oblivious shuffles can be seen as block ciphers. The other group of card shuffles – *non-oblivious* shuffles require

* Authors were supported by Polish National Science Centre contract number DEC-2013/10/E/ST1/00359.

tracing all the cards in order to trace a single one. Since one needs to trace each of the n cards, straightforward application of non-oblivious shuffles as block ciphers would be inefficient. But non-oblivious shuffles are used in cryptographic schemes anyway, just in a slightly different role – often as a building block of a stream cipher.

Let us use the naming convention used by RC4 – a stream cipher designed in 1987 by Ronald Rivest. There is also a long line of stream ciphers: RC4A [24], Spritz [21], RC4+ [22], VMPC [26] – all of them are very similar – they are composed from two algorithms:

1. KSA (Key Scheduling Algorithm) uses a secret key to transform identity permutation of n cards into some other permutation (one can model KSA as a card shuffle).
2. PRGA (Pseudo Random Generation Algorithm) starts with a permutation generated by KSA and outputs random bits from it updating permutation at the same time.

Thus, KSAs of all aforementioned algorithms (RC4, RC4A, Spritz, RC4+, VMPC) can be seen as performing some card shuffling, where a secret key corresponds to/replaces randomness. If we consider a version of the algorithm with purely random secret key of infinite length then we indeed consider a card shuffling procedure. Following [16], we call such a version of the algorithm an *idealized version*. In the case of KSA used by RC4 the idealized version (mathematical model) of the card shuffle is called *Cyclic-to-Random Transpositions* shuffle which indeed is an example of *non-oblivious* shuffle. Recently, in 2013, Rivest and Schuldt presented a new version of an RC4-like cipher (Spritz [21]) which has a new sponge-like KSA which performs more complicated shuffle: $6N$ steps of *Cyclic-to-Random Transpositions* (as part of Whip procedure, see Figure 7; compared to only N steps of in RC4) and in between, partial sorting (so called Crush) of elements in the internal state is performed twice (after 2-nd and 4-th shuffling).

The KSAs of mentioned ciphers perform shuffling for some **predefined number of steps**. The security of such a scheme is mainly based on analyzing idealized version of the algorithm and corresponds to the "quality of a shuffling". Roughly speaking, shuffling is considered as a Markov chain on permutations, all of them converge to uniform distribution (perfectly shuffled cards). Then we should perform as many steps as needed to be close to this uniform distribution, what is directly related to the so-called *mixing time*. This is one of the main drawbacks of RC4: it performs *Cyclic-to-Random Transpositions* for n steps, whereas the mixing time is of order $n \log n$.

There is a long list of papers which point out weaknesses of the RC4 algorithm. Attacks exploited both weaknesses of PRGA and KSA or the way RC4 was used in specific systems [13,11,10,14,5]. As a result, in 2015 RC4 was prohibited in TLS by IETF, Microsoft and Mozilla.

In the paper we use so-called **Strong Stationary Times** (SST) for Markov chains. The main area of application of SSTs is studying the rate of convergence of a chain to its stationary distribution. However, they may also be used for *perfect sampling* from stationary distribution of a Markov chain, consult [20] (on Coupling From The Past algorithm) and [9] (on algorithms involving Strong Stationary Times and Strong Stationary Duality).

1.1 Our contribution

(1) Strong stationary time based KSA algorithm(s) Instead of running a KSA algorithm (*i.e.,* performing the shuffle) for some predefined number of steps, we make it randomized (Las Vegas algorithm). To be more specific we suggest utilization of so-called **Strong Stationary Times** (SST) for Markov chains. We use SST to obtain samples from uniform distribution on all permutations (we actually perform perfect sampling). We show benefits of such approach:

1. Use of SST may allow to close the gap between theoretical models and practice. As a result of Mironov's [16] work, one knows that idealized version of RC4's KSA would need keys of length $\approx 23\ 037$ in order to meet the mathematical model. In fact one may use a better shuffling than the one that is used in RC4 *i.e., time-reversed riffle shuffle* which requires (on average) 4096 bits – not much more than 2048 bits which are allowed for RC4 (see Section 4.1).
2. Coupling methods are most commonly used tool for studying the rate of convergence to stationarity for Markov chains. They allow to bound so-called *total variation distance* between the distribution of given chain at time instant k and its stationary distribution. However, the (traditional) security definitions require something "stronger". It turns out that bounding *separation distance* is what one actually needs. It fits perfectly in the notion of Strong Stationary Times we are using (see Section 4.2).
3. By construction, the running time of our model is key dependent. In extreme cases (very unlikely) it may leak some information about the key but it **does not leak any** information about the resulting permutation to an adversary. We also discuss how one can mask such a leakage (see Section 4.3).

(2) Better SST for RC4's KSA Our complementary contribution (Section 5.2) is the analysis of RC4 showing a new upper bound on number of steps the algorithm should perform. Similarly as in [16], we propose SST which is valid for *Cyclic-to-Random Transpositions* and *Random-to-Random Transpositions*, for the latter one we calculate the mixing time, which is however "faster" than the one given in [16]. It is known that *Random-to-Random Transpositions* card shuffling needs $\frac{1}{2}n\log n$ steps to mix. It is worth mentioning that although *Random-to-Random Transpositions* and *Cyclic-to-Random Transpositions* are similar in the *spirit* it does not transfer automatically that the latter one also needs $\frac{1}{2}n\log n$ steps to mix. Mironov [16] states that expected time till reaching uniform distribution is upper bounded by $2nH_n - n$, we show in Lemma 3 that the expected running time for this SST in *Random-to-Random Transpositions* is equal to:

$$E[T] = nH_n + n + O(H_n)$$

and empirically check that the result is similar for *Cyclic-to-Random Transpositions*. This directly translates into the required steps that should be performed by RC4's KSA.

(3) Note on Spritz construction. We also have a look at Spritz (Section 6), a newer sponge-like construction. We explain why the *Sign distinguisher* attack cannot be successful and provide arguments why the KSA algorithm in Spritz may not perform enough steps.

2 Preliminary

Throughout the paper, let $\mathcal{S}_n$ denote a set of all permutations of a set $\{1,\ldots,n\} =: [n]$.

2.1 Markov chains and rate of convergence

Consider ergodic Markov chain $\mathbf{X} = \{X_k, k \geq 0\}$ on finite state space $\mathbb{E} = \{0,\ldots,M-1\}$ with stationary distribution ψ. Let $\mathcal{L}(X_k)$ denote the distribution of a chain at time instant k. By the *rate of convergence* we understand the knowledge on how fast a distribution of a chain converges to its stationary distribution. We have to measure it according to some distance $dist$. Define

$$\tau_{mix}^{dist}(\varepsilon) = \inf\{k : dist(\mathcal{L}(X_k), \psi) \leq \varepsilon\},$$

which is called *mixing time* (w.r.t. given distance *dist*). In our case the state space is the set of all permutations of $[n]$, *i.e.,* $\mathbb{E} := \mathcal{S}_n$. The stationary distribution is the uniform distribution over $\mathbb{E}$, *i.e.,* $\psi(\sigma) = \frac{1}{n!}$ for all $\sigma \in \mathbb{E}$. In most applications the mixing time is defined w.r.t. total variation distance:

$$d_{TV}(\mathcal{L}(X_k), \psi) = \frac{1}{2} \sum_{\sigma \in \mathcal{S}_n} \left| Pr(X_k = \sigma) - \frac{1}{n!} \right|.$$

The **separation distance** is defined by

$$sep(\mathcal{L}(X_k), \psi) := \max_{\sigma \in \mathbb{E}} \left(1 - n! \cdot Pr(X_k = \sigma)\right).$$

It is relatively easy to check that $d_{TV}(\mathcal{L}(X_k), \psi) \leq sep(\mathcal{L}(X_k), \psi)$.

Strong Stationary Times The definition of separation distance fits perfectly into notion of Strong Stationary Time (SST) for Markov chains. This is a probabilistic tool for studying the rate of convergence of Markov chains allowing also *perfect sampling*. We can think of stopping time as of a running time of algorithm which observes Markov chain $\mathbf{X}$ and which stops according to some stopping rule (depending only on the past).

Definition 1. *Random variable T is a randomized stopping time if it is a running time of the* Randomized Stopping Time *algorithm.*

Algorithm Randomized Stopping Time

1: $k := 0$
2: *coin* :=`Tail`
3: **while** *coin* ==`Tail` **do**
4: At time $k, (X_0, \ldots, X_k)$ was observed
5: Calculate $f_k(X)$, where $f_k : (X_0, \ldots, X_k) \to [0, 1]$
6: Let $p = f_k(X_0, \ldots, X_k)$. Flip the coin resulting in `Head` with probability p and in `Tail` with probability $1 - p$. Save result as *coin*.
7: $k := k + 1$
8: **end while**

Definition 2. *Random variable T is* ***Strong Stationary Time (SST)*** *if it is a randomized stopping time for chain $\mathbf{X}$ such that:*

$$\forall (i \in \mathbb{E})\ Pr(X_k = i | T = k) = \psi(i).$$

Having SST T for chain with uniform stationary distribution lets us bound the following (cf. [3])

$$sep(\mathcal{L}(X_k), \psi) \leq Pr(T > k). \tag{1}$$

We say that T is an optimal SST if $sep(\mathcal{L}(X_k), \psi) = Pr(T > k)$.

2.2 Distinguishers and security definition

We consider three distinguishers: Sign distinguisher and Position distinguisher which are exactly the same as defined in [16], we also consider Permutation distinguisher – we consider his advantage in a traditional cryptographic security definition. That is, Permutation distinguisher is given a permutation π and needs to decide whether π is a result of a shuffle or if it is a permutation selected uniformly at random from the set of all permutations of a given size. The important difference is that the upper bound on Permutation distinguisher's advantage is an upper bound on any possible distinguisher – even those which are not bounded computationally.

`Position distinguisher`

Input: S, t, table $(p_{i,j}^{(t)})$,
threshold A
Output: b
p:=0
for i:=0 to $n-1$ do
  p:=$p + \log(np)_{i,S[i]}^{(t)}$
  if $p < A$
    then return *false*
    else return *true*

`Sign distinguisher`

Input: S, t
Output: b
if sign(S)=$(-1)^t$
  then return *false*
  else return *true*

Position distinguisher Because of the nature of the process (in fact both: *Random-to-Random Transpositions* and *Cyclic-to-Random Transpositions*), the probability that ith card is at jth position depends on t: $p_{i,j}^{(t)} = P(S[j] = i \text{ at time } t)$ which can be pre-computed according to recursion: $p_{i,j}^{(0)} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$ and for $t > 0$:

$$p_{i,j}^{(t)} = \begin{cases} p_{i,j}^{(t-1)} \left(1 - \frac{1}{n}\right) + \frac{1}{n} p_{t_0,j}^{(t-1)} & \text{if } i \neq t_0, \\ \frac{1}{n} & \text{otherwise,} \end{cases}$$

where $t_0 = t \mod n$. The advantage of Position distinguisher dissolves in time – we will upper bound the time needed for this distinguisher to lose his advantage.

Sign distinguisher For a permutation π which has a representation of non-trivial transpositions $\pi = (a_1 b_1)(a_2 b_2) \dots (a_m b_m)$ the sign is defined as: $sign(\pi) = (-1)^m$. So the value of the sign is $+1$ whenever m is even and is equal to -1 whenever m is odd.

Permutation distinguisher The distinguishability game for the adversary is as follows:

Definition 3. *The permutation indistinguishability* $\mathsf{Shuffle}_{\mathsf{S},\mathcal{A}}(n, r)$ *experiment.*

Algorithm $\mathsf{Shuffle}_{\mathsf{S},\mathcal{A}}(n, r)$

Let S be a shuffling algorithm which in each round requires m bits.

1. S is initialized with:
   (a) a key generated uniformly at random $\mathcal{K} \sim \mathcal{U}(\{0,1\}^{rm})$,
   (b) $S_0 = \pi_0$ (identity permutation)
2. S is run for r rounds: $S_r := \mathsf{S}(\mathcal{K})$ and produces a permutation π_r.
3. We set:
   - $c_0 := \pi_{rand}$ a random permutation from uniform distribution is chosen,
   - $c_1 := \pi_r$.
4. A challenge bit $b \in \{0, 1\}$ is chosen at random, permutation c_b is sent to the Adversary.
5. Adversary replies with b'
6. The output of the experiment is defined to be 1 if $b' = b$, and 0 otherwise.

In the case when adversary wins the game (if $b = b'$) we say that $\mathcal{A}$ succeeded. Adversary wins the game if she can distinguish the random permutation from the permutation being a result of the PRPG algorithm.

Definition 4. *A shuffling algorithm* S *generates indistinguishable permutations if for all adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$Pr\left[\mathsf{Shuffle}_{\mathsf{S},\mathcal{A}}(n, r) = 1\right] \leq \frac{1}{2} + \mathsf{negl}(n).$$

The above translates into:

Definition 5. *A shuffling algorithm* S *generates indistinguishable permutations if for any adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that:*

$$\left| \Pr_{K \leftarrow \{0,1\}^{keyLen}} [\mathcal{A}(\mathsf{S}(K)) = 1] - \Pr_{R \leftarrow \mathcal{U}(\mathcal{S}_n)} [\mathcal{A}(R) = 1] \right| \leq \mathsf{negl}(n).$$

3 Related Work

3.1 RC4 algorithm

RC4 is a stream cipher, its so-called *internal state* is (S, i, j), where S is a permutation of $[n]$ and i, j are some two indices. As input it takes $L-$byte message $m_1, \ldots, m_L$ and a secret key K and returns ciphertext $c_1, \ldots, c_L$. The initial state is the output of KSA. Based on this state PRGA is used to output bits which are XORed with the message. The actual KSA algorithm used in RC4 is presented in Figure 1 together with its *idealized version* KSA^* (where a secret key-based randomness is replaced with pure randomness) and our version of the algorithm KSA^{**} (where, in addition to KSA^*, it does not run pre-defined number of steps, but the number depends on a key and is determined by some stopping procedure ST). The details on KSA^{**} will be given in Section 4.1.

KSA(K)	KSA^*	KSA^{**}

for $i := 0$ to $n-1$ **do**
  $S[i] := i$
end for
 	
$j := 0$
for $i := 0$ to $n-1$ **do**
  $j := j + S[i] + K[i \bmod l]$
  swap$(S[i], S[j])$
end for
$i, j := 0$
	for $i := 0$ to $n-1$ **do**
	  $S[i] := i$
	end for
	
	for $i := 0$ to $n-1$ **do**
	  $j :=$**random**(n)
	  swap$(S[i], S[j])$
	end for
	$i, j := 0$
		for $i := 0$ to $n-1$ **do**
		  $S[i] := i$
		end for
		
		while $(\neg$ ST$)$ **do**
		  $j :=$**random**(n)
		  swap$(S[i], S[j])$
		  $i := i + 1 \bmod n$
		end while

Fig. 1. KSA of RC4 algorithm and its idealized version KSA^*. The KSA^{**} has some additional procedure ST (stopping time) which is computed during the execution of the algorithm (for original RC4 simply ST is: stop after n steps).

A closer look at KSA^* reveals that it is actually so-called *Cyclic-to-Random Transpositions*. If we identify elements $[n]$ with cards then we do the following: at step t exchange card t mod n with randomly chosen one.

Throughout the paper, let $\mathbf{Z} = \{Z\}_{t\geq 0}$ denote the chain corresponding to this shuffling and let $\mathcal{L}(Z_t)$ denote the distribution of the chain at time t.

3.2 Sign distinguisher for RC4's KSA

It was observed in [16] that the sign of the permutation at the end of KSA algorithm is not uniform. And as a conclusion it was noticed that the number of discarded shuffles (by PRGA) must grow at least linearly in n. Below we present this result obtained in a different way than in [16], giving the exact formula for advantage at any step t. This form will be used by us to draw conclusions about Spritz algorithm in Section 6.1.

One can look at the sign-change process for the *Cyclic-to-Random Transpositions* as follows: after the table is initialized, sign of the permutation is $+1$ since it is identity so the initial distribution is concentrated in $v_0 = (Pr(\text{sign}(Z_0) = +1), Pr(\text{sign}(Z_0) = -1)) = (1, 0)$.

Then in each step the sign is unchanged if and only if $i = j$ which happens with probability $1/n$. So the transition matrix M_n of a sign-change process induced by the shuffling process is equal to:

$$M_n := \begin{pmatrix} \frac{1}{n} & 1-\frac{1}{n} \\ 1-\frac{1}{n} & \frac{1}{n} \end{pmatrix}.$$

This conclusion corresponds to looking at the distribution of the sign-change process after t steps: $v_0 \cdot M_n^t$, where v_0 is the initial distribution. The eigenvalues and eigenvectors of M_n are $(1, \frac{2-n}{n})$ and $(1,1)^T, (-1,1)^T$ respectively. The spectral decomposition yields

$$v_0 \cdot M_n^t = (1,0) \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \frac{2-n}{n} \end{pmatrix}^t \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} = \left(\frac{1}{2} + \frac{1}{2}\left(\frac{2}{n} - 1\right)^t, \frac{1}{2} - \frac{1}{2}\left(\frac{2}{n} - 1\right)^t \right).$$

For $n = 256$ (which corresponds to the value of n used in RC4) and initial distribution being identity permutation after $t = n = 256$ steps one gets: $v_0 \cdot M_{256}^{256} = (0.567138, 0.432862)$.

In [14] it was suggested that the first 512 bytes of output should be dropped. The Table 3 in Appendix A presents the advantage ϵ of a sign-adversary after dropping k bytes of the output (so after $n+k$ steps of the shuffle, for the mathematical model).

3.3 Position distinguisher for RC4's KSA

Mironov suggested analysis of *idealized* version of KSA algorithm. Being in permutation $S \in \mathcal{S}_n$ at step i, the idealized version swaps element

$S[i]$ with purely random $S[j]$. Treating the permutation as a permutation of a deck of cards, this is exactly a known *Cyclic-to-Random Transpositions* card shuffling. On the other hand if both, $S[i]$ and $S[j]$ are chosen uniformly at random, the procedure is called *Random-to-Random Transpositions* card shuffling. It is known that Random Transposition requires around $\frac{1}{2}n \log n$ to reach uniform distribution, see [8]. Moreover, authors showed that *"most of the action"* actually happens at this step – the process exhibit so called cut-off phenomena. The analysis of Position distinguisher uses Strong Stationary Time (called *Strong Uniform Times* in [16]), based on Broder's construction for *Random-to-Random Transpositions*. Unfortunately Mironov's "estimate of the rate of growth of the strong uniform time T is quite loose" and results "are a far cry both from the provable upper and lower bounds on the convergence rate". He:

- proved an upper bound $O(n \log n)$. More precisely Mironov showed that there exists some positive constant c such that $P[T > cn \log n] \to 0$ when $n \to \infty$. Author experimentally checked that $P[T > 2n \lg n] < 1/n$ for $n = 256$ which corresponds to $P[T > 4096] < 1/256$.
- experimentally showed that $E[T] \approx 11.16n \approx 1.4n \lg n \approx 2857$ (for $n = 256$) – which translates into: on average one needs to drop ≈ 2601 initial bytes.

Later Mosel, Peres and Sinclair [17] proved a matching lower bound establishing mixing time to be of order $\Theta(n \log n)$. However, the constant was not determined.

4 Randomized stopping times and cryptographic schemes

4.1 Strong stationary time based KSA algrorithms

We propose to use the $\mathsf{KSA}^{**}_{\texttt{Shuffle},\texttt{ST}}(n)$ algorithm which works as follows. It starts with identity permutation. Then at each step it performs some card shuffling procedure `Shuffle`. Instead of running it for a pre-defined number of steps, it runs until an event defined by a procedure `ST` occurs. The procedure `ST` is designed in such a way that it guarantees that the event is a Strong Stationary Time. At each step the algorithm uses new randomness – one can think about that as of an idealized version but when the length of a key is greater than the number of random bits required by the algorithm then we end up with a permutation which cannot be distinguished from a random (uniform) one (even by a computationally unbounded adversary).

Algorithm $\mathsf{KSA}^{**}_{\texttt{Shuffle},\texttt{ST}}(n)$

Require: Card shuffling `Shuffle` procedure, stopping rule `ST` which is a Strong Stationary Time for `Shuffle`.

for $i := 0$ to $n-1$ **do**
  $S[i] := i$
end for

while ($\neg$ `ST`) **do**
  `Shuffle`(S)
end while

Notational convention: in $\mathsf{KSA}^{**}_{\texttt{Shuffle},\texttt{ST}}(n)$ we omit parameter n. Moreover, if `Shuffle` and `ST` are omitted it means that we use *Cyclic-to-Random Transpositions* as shuffling procedure and stopping rule is clear from the context (as in KSA^{**} given earlier in Fig. 1). Note that if we use for stopping rule `ST` "stop after n steps" (which of course is *not* SST), it is equivalent to $\mathsf{RC4}$'s KSA^{*} (also in Fig. 1).

Given a shuffling procedure `Shuffle` one wants to have a "fast" stopping rule `ST` (perfectly one wants an optimal SST which is stochastically the smallest). The stopping rule `ST` is a parameter, since for a given shuffling scheme one can come up with a better stopping rule(s). This is exactly the case with *Cyclic-to-Random Transpositions* and *Random-to-Random Transpositions*, we recall Mironov's [16] stopping rule as well as new "faster" rule called `StoppingRuleKLZ` is given (in Section 5.2).

4.2 RST and security guarantees

Coupling method is a commonly used tool for bounding the rate of convergence of Markov chains. Roughly speaking, a coupling of a Markov chain $\mathbf{X}$ with transition matrix $\mathbf{P}$ is a bivariate chain $(\mathbf{X}', \mathbf{X}'')$ such that marginally $\mathbf{X}'$ and $\mathbf{X}''$ are Markov chains with transition matrix $\mathbf{P}$ and once the chains meet they stay together (in some definitions this condition can be relaxed). Let then $T_c = \inf_k\{X'_k = X''_k\}$ *i.e.,* the first time chains meet, called *coupling time*. The *coupling inequality* states that $d_{TV}(\mathcal{L}(X_k), \psi) \leq Pr(T_c > k)$.

On the other hand separation distance is an upper bound on total variation distance, *i.e.,* $d_{TV}(\mathcal{L}(X_k), \psi) \leq sep(\mathcal{L}(X_k), \psi)$. At first glance it seems that it is better to directly bound d_{TV}, since we can have d_{TV} very small, whereas *sep* is (still) large. However, knowing that *sep* is small gives us much more than just knowing that d_{TV} is small, what turns out

to be **crucial** for proving security guarantees (*i.e.,* Definition 5). In our case ($\mathbb{E} = \mathcal{S}_n$ and ψ is a uniform distribution on $\mathbb{E}$) having d_{TV} small, *i.e.,* $d_{TV}(\mathcal{L}(X_k), \psi) = \frac{1}{2}\sum_{\sigma \in \mathcal{S}_n} \left|Pr(X_k = \sigma) - \frac{1}{n!}\right| \leq \varepsilon$ **does not** imply that $|Pr(X_k = \sigma) - \frac{1}{n!}|$ is uniformly small (*i.e.,* of order $\frac{1}{n!}$). Knowing however that $sep(\mathcal{L}(X_k), \psi) \leq \varepsilon$ implies

$$\forall(\sigma \in \mathbb{E}) \ \left|Pr(X_k = \sigma) - \frac{1}{n!}\right| \leq \frac{\varepsilon}{n!}. \tag{2}$$

Above inequality is what we need in our security definitions and shows that the notion of separation distance is an adequate measure of mixing time for our applications.

It is worth noting that $d_{TV}(\mathcal{L}(X_k),\mathcal{U}(\mathbb{E})) \leq \varepsilon$ implies (see Theorem 7 in [2]) that $sep(\mathcal{L}(X_{2k}),\mathcal{U}(\mathbb{E})) \leq \varepsilon$. This means that proof of security which bounds directly total variation distance by ε would require twice as many bits of randomness compared to the result which quarantees ε bound on separation distance.

4.3 RST and timing-attacks

One of the most serious threats to any cryptographic scheme are side-channel attacks. One type of such attacks are *timming-attacks* where an attacker by observing the running time of the execution of a cryptosystem derives information about the key used. Timing attacks are especially powerful [1,23] since an attacker may perform them remotely, over the network (while most of other types of side-channel attacks can be performed only when an attacker is nearby). In order to limit threat of timing-attacks, attempts to implement constant-time cryptographic schemes are made. The problem is that such attempts are usually unsuccessful [19] even if the underlying architecture ("claims") allows for that [25,12].

The running time of an SST-based algorithm strictly depends on the secret key. However, in this section we explain why algorithms using randomized stopping times are immune to timing-attacks, we discuss separately security of two assets: (1) resulting permutation, (2) secret key.

Timing-attacks and the security of the resulting permutation We already defined SST (Definition 2) in Section 2.1 but one can define SST differently.

Definition 6. *Random variable T is **Strong Stationary Time (SST)** if it is a randomized stopping time for chain X such that:*

$$X_T \textit{ has distribution } \psi \textit{ and is independent of } T.$$

Corrolary 1 *The information about the number of rounds that an SST-algorithm performs does not reveal any information about the resulting permutation.*

Corrolary 1 comes from the fact that the Definition 2 which defines SST as a certain randomized stopping time is equivalent to the Definition 6 which defines SST as a variable independent of the resulting distribution. For the proof of the equivalence see [4].

Timing attacks and the security of the secret key Unfortunately, although no information about the resulting permutation leaks, some information about the secret key may leak. Shuffling may reveal randomness through the running time (see Example 2 in Appendix D). In practical implementations, one may use some function of a key instead of pure randomness in each step. Then (at least) two following cases may happen:

1. Bits of the keystream are re-used: the running time of the algorithm (SST) may leak both: information about key and the information about permutation (compare with Example 1 in Appendix D).
2. Bits of the keystream are "fresh" (never re-used): the running time of the algorithm (SST) may leak information about the key but it does not leak any information about the produced permutation! (compare with the Example 2 in Appendix D).

Masking SST One can prevent obtaining information about the secret key by timing-attacks by performing a simple masking. For a stopping rule ST that results in expected running time ET one runs the algorithm for at least ET steps even if the ST occurred earlier.

This eliminates very short executions which could reveal information about the key.

On the other hand, for practical implementation one may want to eliminate the extremely long executions. This can be done by letting the algorithm to run for *e.g.,* $ET + c \cdot \sqrt{VarT}$ (where c is a parameter and $VarT$ is the variance for the ST).

5 (Not so) random shuffles of RC4 – revisited

5.1 Mironov's stopping rule – details

The goal of KSA of original RC4 is to produce a pseudorandom permutation of $n = 256$ cards. The original algorithm performs 256 steps of *Cyclic-to-Random Transpositions*. However it is known that the mixing

time of *Cyclic-to-Random Transpositions* is $\Theta(n \log n)$. Then performing only 256 (*i.e.,* n) steps seems much too less. In fact, it is recommended to perform at least 3072 steps, see Mironov [16]. Generally, the more steps are performed, the closer to uniformity the final permutation is. Mironov considered idealized version of the algorithm together with the following marking rule:

> "At the beginning all cards numbered $0, \ldots, n-2$ are *unchecked*, the $(n-1)^{th}$ card is *checked*. Whenever the shuffling algorithm exchanges two cards, $S[i]$ and $S[j]$, one of the two rules may apply before the swap takes place:
> a. If $S[i]$ is unchecked and $i = j$, check $S[i]$.
> b. If $S[i]$ is unchecked and $S[j]$ is checked, check $S[i]$.
> The event T happens when all cards become checked. "

Then the author proves that this is a SST for *Cyclic-to-Random Transpositions* and shows that there exists constant c (can be chosen less than 30) such that $Pr[T > cn \log n] \to 0$ when $n \to \infty$. Empirically, for $n = 256$ he shows that $Pr[T > 2n \log n] < 1/n$. Note that this marking scheme is also valid for *Random-to-Random Transpositions* shuffling.

Lemma 1. *The expected running time of Random-to-Random Transpositions shuffling with Mironov stopping rule is:*

$$ET = 2nH_n - n + O(H_n).$$

Proof. We start with one card checked. When k cards are checked, then probability of checking another one is equal to $p_k = \frac{(n-k)(k+1)}{n^2}$. Thus, the time to check all the cards is distributed as a sum of geometric random variables and its expectation is equal to:

$$\sum_{k=1}^{n-1} \frac{1}{p_k} = 2\frac{n^2}{n+1} H_n - n = 2nH_n - n + O(H_n).$$

5.2 Better stopping rule

We suggest another "faster" SST which is valid for both *Cyclic-to-Random Transpositions* and *Random-to-Random Transpositions*. We will calculate its expectation and variance for *Random-to-Random Transpositions* and check experimentally (see Appendix C) that it is similar if the stopping rule is applied to *Cyclic-to-Random Transpositions*. As a result (proof given at the end of this Section) we have:

Theorem 1. *Let $\mathcal{A}$ be an adversary. Let $K \in \{0,1\}^{rn}$ be a secret key. Let $\mathcal{S}(K)$ be $\mathsf{KSA}^{*}_{\mathtt{RTRT}}$ (i.e., with Random-to-Random Transpositions shuffling) which runs for*

$$r = n(H_n + 1) + \frac{\pi n}{2}\frac{1}{\sqrt{n!\varepsilon}}$$

steps with $0 < \varepsilon < \frac{1}{n!}$. Then

$$\left| \Pr_{K \leftarrow \{0,1\}^{rm}} [\mathcal{A}(\mathcal{S}(K)) = 1] - \Pr_{R \leftarrow \mathcal{U}(\mathcal{S}_n)} [\mathcal{A}(R) = 1] \right| \leq \varepsilon$$

The stopping rule is given in `StoppingRuleKLZ` algorithm.

Algorithm `StoppingRuleKLZ`

Input set of already marked cards $\mathcal{M} \subseteq \{1, \ldots, n\}$, round r, $Bits$
Output {`YES`,`NO`}

$j =$`n-value`$(Bits)$
if there are less than $\lceil (n-1)/2 \rceil$ marked cards **then**
  if both $\pi[r]$ and $\pi[j]$ are unmarked **then**
    mark card $\pi[r]$
  end if
else
  if ($\pi[r]$ is unmarked and $\pi[j]$ is marked) OR ($\pi[r]$ is unmarked and $r = j$) **then**
    mark card $\pi[r]$
  end if
end if

if all cards are marked **then**
  `STOP`
else
  `CONTINUE`
end if

Lemma 2. *The resulting permutation of KSA^{**} with $\mathtt{ST} =$`StoppingRuleKLZ` has a uniform distribution over $\mathcal{S}_n$.*

Proof. We will show that the running time of the algorithm is a SST, *i.e.,* that the card marking procedure specified in `StoppingRuleKLZ` is a SST for *Cyclic-to-Random Transpositions*. First phase of the procedure (*i.e.,* the case when there are less than $\lceil (n-1)/2 \rceil$ cards marked) is constructing a random permutation of marked cards by placing unmarked cards on randomly chosen unoccupied positions, this is actually first part of Matthews's marking [15] scheme. Second phase is simply a

Broder's construction. Theorem 9. of [16] shows that this is a valid SST for *Cyclic-to-Random Transpositions*. Both phases combined produce a random permutation of all cards.

Remark 1. One important remark should be pointed. Full Matthews's marking [15] scheme is "faster" than ours. However, although it is a SST for *Random-to-Random Transpositions*, this is not SST for *Cyclic-to-Random Transpositions*.

Calculating ET or $VarT$ seems to be a challenging task. But note that marking scheme `StoppingRuleKLZ` also yields a valid SST for *Random-to-Random Transpositions*. In next Lemma we calculate ET and $VarT$ for this shuffle, later we experimentally show that ET is very similar for both marking schemes.

Lemma 3. *Let T be the running time of KSA** with Random-to-Random Transpositions shuffling and ST=StoppingRuleKLZ. Then we have*

$$\begin{aligned} E[T] &= nH_n + n + O(H_n), \\ Var[T] &\sim \tfrac{\pi^2}{4} n^2, \end{aligned} \tag{3}$$

where H_n is the $n-th$ harmonic number and $f(k) \sim g(k)$ means that $\lim_{k\to\infty} \frac{f(k)}{g(k)} = 1$.

The details of the proof of the Lemma 3 are in Appendix B.

Proof. Define T_k to be the first time when k cards are marked (thus $T \equiv T_n$). Let $d = \lceil (n-1)/2 \rceil$. Then T_d is the running time of the first phase and $(T_n - T_d)$ is the running time of the second phase. Denote $Y_k := T_{k+1} - T_k$.

Assume that there are $k < d$ marked cards at a certain step. Then the new card will be marked in next step if we choose two unmarked cards what happens with probability: $p_a(k) = \frac{(n-k)^2}{n^2}$. Thus Y_k is a geometric random variable with parameter $p_a(k)$ and

$$E[T_d] = n^2 \left(H_n^{(2)} - H_{n-d}^{(2)} \right).$$

Now assume that there are $k \geq d$ cards marked at a certain step. Then, the new card will be marked in next step with probability:

$$p_b(k) = \frac{(n-k)(k+1)}{n^2}$$

and Y_k is a geometric random variable with parameter $p_a(k)$. Thus:

$$E[T_n - T_d] = nH_n - \frac{n}{n+1}H_n + \frac{n^2}{n+1}\left(H_{n-d} - H_d\right).$$

For variance we have:

$$Var[T_n] = Var[T_d] + Var[T_n - T_d] \sim \frac{\pi^2}{4}n^2.$$

□

From Lemma 3 we immediately have the following:

Corrolary 2 *Consider the chain corresponding to* KSA** *with Random-to-Random Transpositions shuffling and* `ST=StoppingRuleKLZ`*. Then we have*

$$\tau_{mix}^{sep}(\varepsilon) \leq n(H_n + 1) + \frac{\pi n}{2}\frac{1}{\sqrt{\varepsilon}}.$$

Proof (of Theorem 1). In Theorem 1 we perform *Random-to-Random Transpositions* for $r = \tau_{mix}^{sep}(n!\varepsilon)$ steps, *i.e.,* $sep(\mathcal{L}(X_r), \psi) \leq n!\varepsilon$ Inequality (2) implies that $|Pr(X_r = \sigma) - \frac{1}{n!}| \leq \varepsilon$ for any permutation σ and thus completes the proof. □

5.3 Predefined number of steps vs SST-based algorithms

There is a subtle difference between the randomized stopping time (like the one suggested in the paper) and an algorithm that performs a predefined number of steps. If one wants to achieve security level of *e.g.,* $\varepsilon = O(1/n^k), k > 2$ then the number of steps that would assure that the advantage is smaller than ε would need to be equal to: $\tau_{mix}(\varepsilon) \leq n(H_n + 1) + \frac{\pi n}{2}n^{k/2} = O\left(n^{1+k/2}\right)$.

As we can see the estimated running time of *Cyclic-to-Random Transpositions* with both stopping rules is similar to the theoretical results for *Random-to-Random Transpositions*. Recall that for *Cyclic-to-Random Transpositions* it is known that the mixing time is of order $\Theta(n \log n)$, see [17], however the constant was not determined. Based on our new SST and simulations (see Appendix C)one can conjecture the following

Conjecture 1. The mixing time τ_{mix}^{sep} for *Cyclic-to-Random Transpositions* converges to $n \log n$ as $n \to \infty$.

6 A note on Spritz

Spritz [21] is a new stream cipher that was proposed in 2014 by Rivest and Schuldt as a possible replacement for RC4.

The first cryptanalytic results were already achieved: inefficient state recovery attack [6] (with 2^{1400} complexity) later improved by [7] (with 2^{1247} steps). The second paper presents also a devastating distinguishing attacks of complexity $2^{44.8}$ (multiple key-IV setting) and $2^{60.8}$ (single key-IV setting).

Here we analyze the distribution of the internal state of Spritz after the main part of the scheme (procedure SHUFFLE()) is run. Similarly to the previous approach we replace the deterministic part of UPDATE() function: $j := k+S[j+S[i]]$, with its idealized version: $j := \mathbf{random}(n)$).

The definitions of Spritz' procedures that are of our interest are presented in Appendix E.

6.1 Sign distinguisher

Although we did not find strong stationary time for "KSA" part of Spritz algorithm, one can easily notice that the Sign Distinguisher has no advantage at all. This property is achieved thanks to Crush procedure. During this procedure, the table S is partially sorted *i.e.,* elements at positions v and $n-1-v$ (for $v = 0 \dots \lfloor N/2 \rfloor - 1$) are swapped whenever $S[v] > S[n-1-v]$. So this corresponds to multiplying the sign process by:

$$M_{crush} := \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

If Spritz is used as stream cipher, as part of Squeeze procedure, at least one call to Shuffle is made. So the distribution of sign can be described as

$$v_0 \cdot M_n^{2n} M_{crush}^{n/2} M_n^{2n} M_{crush}^{n/2} M_n^{2n} = \left(\frac{1}{2}, \frac{1}{2}\right).$$

This means that advantage of Sign Distinguisher for Spritz equals to 0.

6.2 Position distinguisher

Let us recall what was one of the main drawback of original RC4: it performed n steps instead of $cn \log n$. The underlying mathematical model is simply a *Cyclic-to-Random Transpositions* card shuffling. This is somehow similar to *Random-to-Random Transpositions* for which it takes of

order of $n \log n$ steps. More exactly, there is so-called cutoff phenomena at $\frac{1}{2} n \log n$. Roughly speaking, lower and upper bounds are of this order. Analysis of *Cyclic-to-Random Transpositions* seemed to be harder, recently [17] the matching lower bound was established showing that mixing time is of order $\Theta(n \log n)$.

Recall that Spritz performs: in total $6n$ steps of *Cyclic-to-Random Transpositions* (as part of Whip procedure) and partial sorting of elements (Crush procedure) in the internal state is performed twice (after 2-nd and 4-th shuffling). This is of course more complicated shuffling than just repeating *Cyclic-to-Random Transpositions*.

Clever use of Crush lets Spritz to get rid of the Sign Distinguisher but at the same time it seems that it may badly influence the mixing time.

Imagine that there exists some SST which during the Spritz execution performs marking of the elements. Marked elements satisfy property that their mutual position is equally distributed. Now, take a look at the step when Crush is performed and there are two marked elements at positions v and $N-1-v$. Then after Crush their relative position will be uniquely determined! This observation suggests that mixing time for Spritz would be greater than $n \log n$.

7 A note on optimal shuffling

Cyclic-to-Random Transpositions is the shuffle used in RC4 and in Spritz. To reach stationarity (*i.e.,* produce random permutation), as we shown, one needs to perform $O(n \log n)$ steps. In each step we use a random number from interval $[0, \ldots, n-1]$, thus this shuffling requires $O(n \log^2 n)$ random bits.

One can ask the following question: Is this shuffling optimal in terms of required bits? The answer is no. The entropy of the uniform distribution on $[n]$ is $O(n \log n)$ (since there are $n!$ permutations of $[n]$), thus one could expect that optimal shuffling would require this number of bits.

We will shortly describe (time reversal of) *Riffle Shuffle*, for details see [2]. For a given permutation $\sigma \in \mathcal{S}_n$ we assign each element a random bit. Then we put all the elements (cards) with assigned bit 0 to the top *keeping* their relative ordering. The following is a known SST for this shuffle: At the beginning all $\binom{n}{2}$ pairs of cards are unmarked. At each step one marks a pair (i, j) if elements i and j were assigned different bits. Let T be the first time all pairs are marked.

The above SST of *Riffle Shuffle* has $ET = 2 \lg n$, at each step n random bits are used, thus this shuffling requires $2n \lg n$ random bits, matching the requirement of optimal shuffle (up to a constant).

In this paper we mainly focused on RC4 and thus on *Cyclic-to-Random Transpositions* shuffle. However, we wanted to point out that using *Riffle Shuffle* (or other shuffling schemes) can result in better efficiency of the whole scheme.

8 Conclusions

We presented the benefits of using Strong Stationary Times in cryptographic schemes (pseudo random permutation generators). These algorithms have a "health-check" built-in and guarantee the best possible properties (when it comes to the quality of randomness of the resulting permutation). We showed that use of SST does not lead to timing attacks. We showed that algorithms using SST achieve better security guarantees than any algorithm which runs predefined number of steps.

	RC4	Mironov	KSA**	RS
#bits used	40 to 2 048			
#bits asymptotics		$(2nH_n - 1) \lg n$	$n(H_n + 1) \lg n$	$2n \lg n$
#bits required		23 037	14 590	4 096

Fig. 2. Comparison between number of bits used by RC4 (40 to 2048) and required by mathematical models (Mironov [16] and ours) versus length of the key for the time-reversed riffle shuffle. *Bits asymptotics* approximates the number of fresh bits required by the mathematical model (number of bits required by the underlying markov chain to converge to stationary distribution). *Bits required* is (rounded) value of *# bits asymptotics* when $n = 256$.

Complementarily, we proved better bound for the mixing-time of the *Cyclic-to-Random Transpositions* shuffling process which is used in RC4 and showed that different, more efficient shuffling methods (*i.e.,* time reversal of *Riffle Shuffle*) may be used as KSA. This last observation shows that the gap between mathematical model (4096 bits required) and reality (2048 allowed as maximum length of RC4) is not that big as previously thought (bound of 23037 by Mironov [16]).

References

1. Martin R. Albrecht and Kenneth G. Paterson. Lucky Microseconds: A Timing Attack on Amazon's s2n Implementation of TLS. *Advances in Cryptology - EUROCRYPT*, 9665:622—-643, 2016.
2. David Aldous and Persi Diaconis. Shuffling cards and stopping times. *American Mathematical Monthly*, 1986.
3. David Aldous and Persi Diaconis. Strong Uniform Times and Finite Random Walks. *Advances in Applied Mathematics*, 97:69–97, 1987.
4. David Aldous and Persi Diaconis. Strong uniform times and finite random walks. *Advances in Applied Mathematics*, 1987.
5. Nadhem AlFardan, Daniel J Bernstein, Kenneth G Paterson, Bertram Poettering, and Jacob C N Schuldt. On the Security of RC4 in TLS. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 305–320, Washington, D.C., 2013. USENIX.
6. Ralph Ankele, Stefan Kölbl, and Christian Rechberger. State-Recovery Analysis of Spritz. pages 204–221. Springer International Publishing, 2015.
7. Subhadeep Banik and Takanori Isobe. Cryptanalysis of the Full Spritz Stream Cipher. In *FSE 2016*, pages 63–77. Springer Berlin Heidelberg, 2016.
8. Persi Diaconis and Mehrdad Shahshahani. Generating a random permutation with random transpositions. *Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57(2):159–179, 1981.
9. James Allen Fill. An interruptible algorithm for perfect sampling via Markov chains. *The Annals of Applied Probability*, 8(1):131–162, feb 1998.
10. S Fluhrer, I Mantin, and A Shamir. Weaknesses in the key scheduling algorithm of RC4. *Selected areas in cryptography*, 2001.
11. Scott R Fluhrer and David a. McGrew. Statistical Analysis of the Alleged RC4 Keystream Generator. *Fast Software Encryption, 7th International Workshop*, pages 19–30, 2000.
12. Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware. *IACR Eprint*, 2016.
13. J Golic. Linear Statistical Weakness of Alleged RC4 Keystream Generator. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, jul 1997.
14. Itsik Mantin and Adi Shamir. A Practical Attack on Broadcast RC4. *Fast Software Encryption, 8th International Workshop, Yokohama, Japan*, pages 152–164, 2001.
15. Peter Matthews. A strong uniform time for random transpositions. *J. Theoret. Probab.*, 1(4):411–423, 1988.
16. Ilya Mironov. (Not So) Random Shuffles of RC4. *Advances in Cryptology—CRYPTO 2002*, 2002.
17. Elchanan Mossel, Yuval Peres, and Alistair Sinclair. Shuffling by semi-random transpositions. *Foundations of Computer Science*, pages 572–581, 2004.
18. Moni Naor and Omer Reingold. On the construction of pseudo-random permutations. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing - STOC '97*, pages 189–199, New York, New York, USA, may 1997. ACM Press.
19. Cesar Pereida García, Billy Bob Brumley, and Yuval Yarom. Make Sure DSA Signing Exponentiations Really are Constant-Time.

20. J G Propp and D B Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9:223–252, 1996.
21. Jacob C. N. Schuldt; Ronald L. Rivest. Spritz—a spongy RC4-like stream cipher and hash function, 2014.
22. Bart Preneel Souradyuti Paul. A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher. In Bimal Roy and Willi Meier, editors, *Fast Software Encryption*, volume 3017 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
23. Francois-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage resilient cryptography in practice. *Towards Hardware-Intrinsic Security*, pages 99–134, 2010.
24. Goutam Paul Subhamoy Maitra. Analysis of RC4 and Proposal of Additional Layers for Better Security Margin. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *Progress in Cryptology - INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
25. Yuval Yarom, Daniel Genkin, and Nadia Heninger. CacheBleed : A Timing Attack on OpenSSL Constant Time RSA. *CHES*, 2016.
26. Bartosz Zoltak. VMPC One-Way Function and Stream Cipher. In *Fast Software Encryption*, pages 210–225. 2004.

Appendix

A Sign distinguisher advantage

k	$+1$	-1	ϵ
0	.5671382998250798	.4328617001749202	$2^{-3.89672}$
256	.509015	.490985	$2^{-6.79344}$
512	.5012105173235390	.4987894826764610	$2^{-9.69016}$
768	.500163	.499837	$2^{-12.5869}$
1024	.5000218258757580	.4999781741242420	$2^{-15.4836}$
2048	.5000000070953368	.4999999929046632	$2^{-27.0705}$
4096	.5000000000000007	.4999999999999993	$2^{-50.2442}$
8192	.5	.5	$2^{-96.5918}$

Fig. 3. The advantage (ϵ) of Sign distinguisher of RC4 after discarding initial k bytes.

B Detailed proof of Lemma 3

Proof. Define T_k to be the first time when k cards are marked (thus $T \equiv T_n$). Let $d = \lceil (n-1)/2 \rceil$. Then T_d is the running time of the first phase and $(T_n - T_d)$ is the running time of the second phase. Denote $Y_k := T_{k+1} - T_k$.

Assume that there are $k < d$ marked cards at a certain step. Then the new card will be marked in next step if we choose two unmarked cards what happens with probability:

$$p_a(k) = \frac{(n-k)^2}{n^2}.$$

Thus Y_k is a geometric random variable with parameter $p_a(k)$ and

$$\begin{aligned} E[T_d] &= \sum_{k=0}^{d-1} E[Y_k] = \sum_{k=0}^{d-1} \frac{1}{p_a(k)} = \sum_{k=0}^{d-1} \frac{n^2}{(n-k)^2} = n^2 \sum_{k=n-d+1}^{n} \frac{1}{k^2} \\ &= n^2 \left(H_n^{(2)} - H_{n-d}^{(2)} \right) = n^2 \left(\frac{1}{n} + O\left(\frac{1}{n^2} \right) \right) = n + O(1). \end{aligned}$$

Now assume that there are $k \geq d$ cards marked at a certain step. Then, the new card will be marked in next step with probability:

$$p_b(k) = \frac{(n-k)(k+1)}{n^2}$$

and Y_k is a geometric random variable with parameter $p_a(k)$. Thus:

$$\begin{aligned} E[T_n - T_d] &= \sum_{k=0}^{d-1} E[Y_k] = \sum_{k=d}^{n-1} \frac{1}{p_b(k)} = \sum_{k=d}^{n-1} \frac{n^2}{(n-k)(k+1)} \\ &= \frac{n^2}{n+1} \sum_{k=d}^{n-1} \left(\frac{1}{n-k} + \frac{1}{k+1} \right) = \frac{n^2}{n+1} \left(\sum_{k=1}^{n-d} \frac{1}{k} + \sum_{k=d+1}^{n} \frac{1}{k} \right) \\ &= \frac{n^2}{n+1} \left(H_{n-d} + H_n - H_d \right) = \frac{n^2}{n+1} H_n + \frac{n^2}{n+1} \left(H_{n-d} - H_d \right) \\ &= nH_n - \tfrac{n}{n+1} H_n + \tfrac{n^2}{n+1} \left(H_{n-d} - H_d \right) = nH_n + O(H_n) + O(1) = nH_n + O(H_n). \end{aligned}$$

For variance we have:

$$Var[T_d] = \sum_{k=0}^{d-1} Var[Y_k] = \sum_{k=0}^{d-1} \frac{1-p_a(k)}{(p_a(k))^2} = \sum_{k=0}^{d-1} \frac{1-\frac{(n-k)^2}{n^2}}{\left(\frac{(n-k)^2}{n^2}\right)^2} \approx \int_0^{\frac{n}{2}} \frac{1-\frac{(n-x)^2}{n^2}}{\left(\frac{(n-x)^2}{n^2}\right)^2} dx = \frac{4}{3}n.$$

$$Var[T_n - T_d] = \sum_{k=d}^{n-1} Var[Y_k] = \sum_{k=d}^{n-1} \frac{1-p_b(k)}{(p_b(k))^2} = \sum_{k=d}^{n-1} \frac{1-\frac{(n-k)(k+1)}{n^2}}{\frac{(n-k)^2(k+1)^2}{n^4}}$$

$$= n^2 \sum_{k=d}^{n-1} \frac{n(n-1)+k(1-n)+k^2}{(n-k)^2(k+1)^2} \approx n^2 \cdot \frac{1}{2} \sum_{k=0}^{n-1} \frac{n(n-1)+k(1-n)+k^2}{(n-k)^2(k+1)^2}$$

$$\approx \frac{n^2}{2}\left[\left(\frac{2}{n} - \frac{4}{n^2}\right) H_n + 3H_n^{(2)}\right] \sim \frac{\pi^2}{4}n^2.$$

Finally

$$Var[T_n] = Var[T_d] + Var[T_n - T_d] \sim \frac{\pi^2}{4}n^2.$$

□

C Experimental results

The expected running time of $\mathsf{KSA}^{**}_{\mathtt{RTRT}}$ (*i.e.,* with *Random-to-Random Transpositions* shuffling) is known:

- with `ST`=`StoppingRuleKLZ` it is $n(H_n + 1)$
- with stopping rule used in [16] it is $2nH_n - n$

For both stopping rules applied to *Cyclic-to-Random Transpositions* no precise results on expected running times are known. Instead we estimated them via simulations, simply running 10.000 of them. The results are given in Fig. 4.

		`StoppingRuleKLZ`	Mironov's `ST`
ET	$n = 256$	1811	2854
	$n = 512$	3994	6442
	$n = 1024$	8705	14324
VarT	$n = 256$	111341	156814
	$n = 512$	438576	597783
	$n = 1024$	1759162	2442503

	$n(H_n + 1)$	$2nH_n - n$
$n = 256$	1823.83	2879.66
$n = 512$	4002.05	6468.11
$n = 1024$	8713.39	14354.79

Fig. 4. Simulations' results for Mironov's and `StoppingRuleKLZ` stopping rules.

D Timing-attacks and KSA**

Example 1. (Top to random shuffle T2R – timing attack – re-used randomness)

Consider algorithm $\mathsf{KSA}^{**}_{\mathtt{T2R},\mathtt{ST}}$ with shuffling procedure corresponding to *Top-To-Random* card shuffling (put the card $S[1]$ which is currently on top to the position j defined by the randomness in the current round) and following stopping time ST:

- before the start of the algorithm, mark the last card (*i.e.,* the card n is marked[3]),
- stop one step after the marked card reaches the top of the deck.

The sample execution of the algorithm is given in Fig. 5.

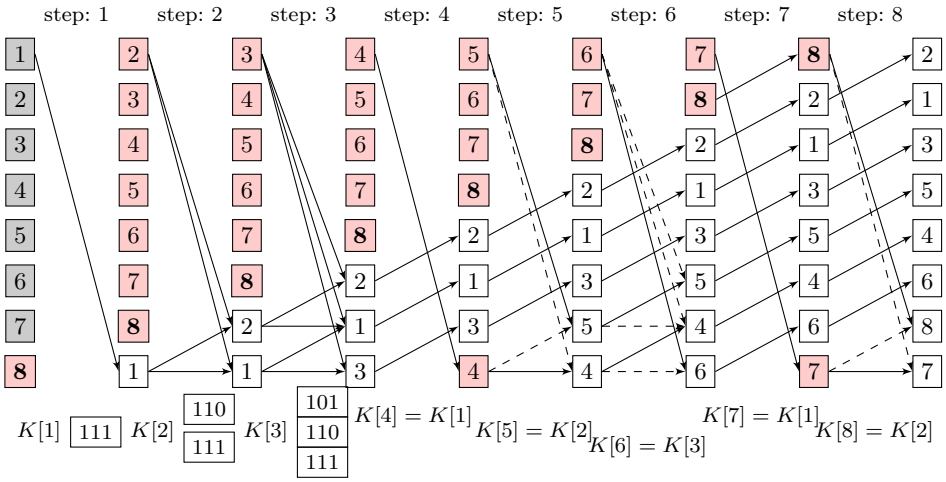

Fig. 5. Example run of *top-to-random* shuffle with "reused randomness" which is taken from the key K of the length equal to three 8-value bytes (9-bits). Let us assume that the running time of SST was exactly 8. Conditioning on the number of steps one can figure out that the first word of the key must be equal to 111 while the second part $K[2] \in \{110, 111\}$ with the same probability. Finally $K[3] \in \{101, 110, 111\}$ and for the step i: $K[i] = K[i \bmod 3]$. So now, instead of possible $2^9 = 512$ permutations which can be generated from 9 bits of key, only 6 are possible (based on the fact that SST has stoped exactly after 8 steps).

[3] It is known [2] that optimal SST for *top to random* initially marks second from the bottom card.

Example 2. (Top-to-random – timing attack – fresh randomness) Let us now consider a very similar situation with one important difference. Now no portion of the key is re-used. An example run of the algorithm is presented on the Figure 6. Based on the knowledge on the number of performed steps, one can learn some information about the secret key (*i.e.,* $K[1] = 000$, $K[2]$ is either 110 or 111) but still no adversary can learn anything about the resulting permutation because any information is generated with exactly the same probability.

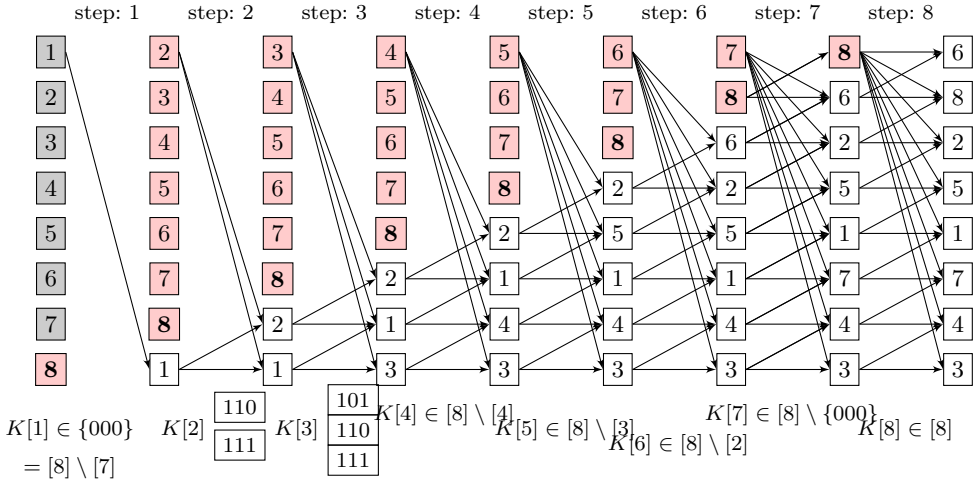

Fig. 6. Example run of *top-to-random* shuffle with "fresh randomness" taken from the key K of the length equal to 8 8-value bytes (24-bits). Conditioning on the number of steps of the SST (in this case 8) one can find out that: out of the possible 2^{24} keys only (exactly) 8! keys are possible (due to the fact that SST stopped after 8 steps) and every of 8! permutations are possible (based on the fact that SST has stopped exactly after 8 steps) – moreover each permutation with exactly the same probability. SST leaks bits of the key *i.e.,* $K[1] = 111$ but does not leak any information about the produced permutation.

E Spritz definition

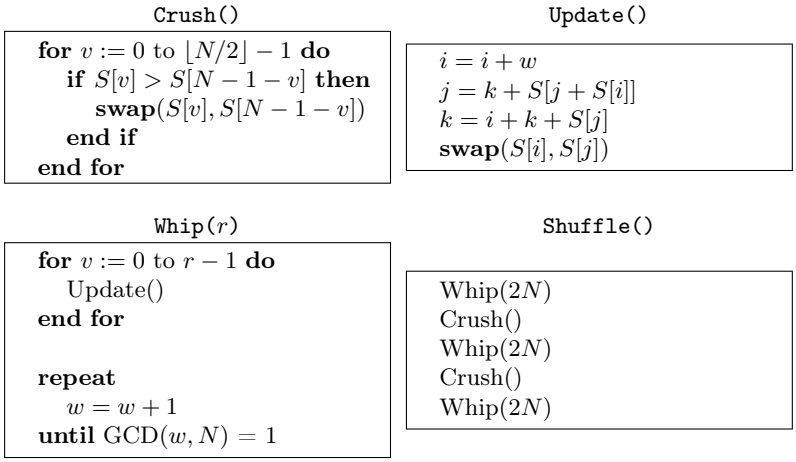

Fig. 7. Building blocks of Spritz