A leakage resilient shuffling *

Anonymous Paweł Lorek[2], Michał Kulis[1], and Filip Zagórski[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. Analysis of various card-shuffles – finding its mixing-time is an old mathematical problem. The results show that *e.g.,* it takes $\mathcal{O}(\log n)$ riffle-shuffles (Aldous and Diaconis, *American Mathematical Monthly*, 1986) to shuffle a deck of n cards while one needs to perform $\Theta(n \log n)$ steps via cyclic to random shuffle (Mossel et al., *FOCS*, 2004). Algorithms for generating pseudo-random permutations play a major role in cryptography. *Oblivious* card shuffles can be seen as block ciphers (and *e.g.,* may be used for format-preserving encryption while *non-oblivious* card shuffles often are a building block for cryptographic primitives (*e.g.,* Spritz, RC4).
Unfortunately, all results about the mixing times of card shuffling algorithms are in the black-box model. The model does not capture real-world capabilities of adversaries who may be able to *e.g.,* obtain some information about the randomness used during the shuffling. In this paper we investigate the impact on the mixing time of the riffle shuffle by an adversary who is able to eavesdrop some portion of the random bits used by the process. More precisely: assuming that each bit of the randomness leaks independently with probability p we show that whenever *RiffleSST* performs $r = \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\frac{1}{\varepsilon n!}\right)$ steps, it cannot be distinguished from a permutation selected uniformly at random with the advantage larger than ε.

Keywords: Leakage resilience, pseudo-random permutation generator, Markov chains, mixing time, card shuffle, riffle shuffle, stream cipher, distinguisher.

1 Introduction

1.1 Card shuffling and cryptography

Shuffling procedures (or card shuffles) are used as cryptographic building blocks. A card shuffle as a way to obtain a permutation can be seen as

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

a block cipher. Due to efficiency reasons, only *oblivious* card shuffles are good candidates for block ciphers (*e.g.,* [14]). *Non-oblivious* card shuffles need to be used differently (*e.g.,* as a key scheduling algorithm [13]). But card shuffles may also help to design/describe higher-level systems *e.g.,* ones which goal is to achieve anonymity like: Private Information Retrieval schemes [22,10] or mixing with application to voting [9,7].

Oblivious shuffles. The applicability of *oblivious* card shuffles to cryptography was noticed many years ago by *e.g.,* Naor and Reingold [16] for Thorp shuffle. Oblivious shuffles can be seen as block ciphers: suppose that a deck has $n = 2^l$ cards then the message space and the ciphertext space is equal to the set of all binary strings of length l. Randomness used by the process corresponds to the trajectory of a given card, and one does not need to trace trajectories of other cards. This point of view led to the proposals [14,8,17,15] (useful for *e.g.,* format preserving encryption schemes) with provable properties in the black-box model (meaning that an adversary has only access to inputs and output of the shuffling algorithm like in CPA-experiment [chosen-plaintext attack]).

Non-oblivious shuffles. In *non-oblivious* shuffles one needs to trace a trajectory of every of the n cards to be able to tell what is the final position of a single card. Because of that non-oblivious shuffles are used as building blocks of cryptographic schemes (in *e.g.,* [12], and especially as Key Scheduling Algorithms in *e.g.,* RC4, Spritz [18], *etc.*) rather than being used as encryption schemes.

Then the security of a cryptographic scheme which uses some card shuffling as a building block depends on the quality of the shuffle. This can be measured by how a given shuffling is close to the uniform distribution (over all possible permutations). This depends on:

1. the rate of convergence to the stationary distribution (depends on the shuffling algorithm itself);
2. the number of steps made by the algorithm. (In particular we are interested in the number of steps needed so that the distribution of the chain at the given step is close to uniform one.)

One of the weaknesses found in RC4 is that its Key Scheduling Algorithm (KSA) makes only n steps while the rate of convergence is $O(n \log n)$ [13,11].

1.2 Leakage

Classically, in (black-box) cryptography, the security definitions assume that an attacker can have only access to inputs and outputs of a cryptographic scheme – for instance, for encryption one considers CPA-security (Chosen Plaintext Attack) or CCA-security (Chosen Cipertext Attack). These definitions assume that no information about the secret-key (or some internal computations) is leaked. In reality however, a device (or particular scheme or protocol implementation) may expose to an adversary lots of additional information, an adversary may measure all kinds of side-channels *e.g.,* electromagnetic [6], acoustic [5] *etc.*. One of the most powerful kind of side-channel attacks are *timing attacks* (because an adversary may perform them remotely), see [1,21]; or the combination of techniques [23].

Practice shows that an adversary may obtain some direct information about the secret key: assume that $\mathbf{b} = (b_1, \ldots, b_t)$ bits of key (or of a function of key) are used in a given round of the algorithm. Then the Hamming weight $HW(\mathbf{b}) = \sum_{i=1}^{t} b_i$ can leak. More precisely Hamming weight with some Gaussian noise leaking was considered *e.g.,* in [21,20,19].

1.3 Our contribution

In this paper we consider the model where each bit b_i of the randomness used by the shuffling algorithm leaks independently with some prescribed probability $p \in [0, 1)$. We analyze a single run of the Riffle Shuffle and our goal is to find the number of rounds the algorithm needs to make in order not to reveal *any* information about the resulting permutation (in the presence of an eavesdropping adversary).

We analyze a non-oblivious shuffling algorithm called *RiffleSST* that is leakage resilient. We show that even if an adversary $\mathcal{A}$ learns each bit of the key K with probability p (knowledge of bits of the key are denoted by $\Lambda_p(K)$) it cannot tell much about the resulting permutation. Putting this in other words: even if an adversary knows some bits of the key $\Lambda_p(K)$, it cannot distinguish the permutation produced by r rounds of the *RiffleSST* algorithm from the permutation sampled from the uniform distribution with probability better than ε.

The contribution of this paper is the first analysis of a card shuffle algorithm in the presence of a randomness-eavesdropper. The result is formulated as Theorem 1.

Theorem 1. *Let $\mathcal{A}$ be an adversary. Let $K \in \{0, 1\}^{rn}$ be a secret key. Let $\Lambda_p(K)$ be the random variable representing the leakage of the key such* *that $\mathcal{A}$ learns each bit of the key independently at random with probability p. Let $\mathsf{S}_{r,n}(K)$ be RiffleSST shuffle of n cards which runs for*

$$r = \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\frac{1}{\varepsilon n!}\right)$$

steps with $0 < \varepsilon < 1/n!$, then

$$\left| \Pr_{K \leftarrow \{0,1\}^{rn}} [\mathcal{A}(\Lambda_{p,r}, \mathsf{S}_{r,n}(K)) = 1] - \Pr_{R \leftarrow \mathcal{U}(\mathcal{S}_n)} [\mathcal{A}(\Lambda_{p,r}, R) = 1] \right| \leq \varepsilon.$$

2 Preliminary

2.1 Security definition

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

We would like to model an adversary whose goal is to distinguish a permutation which is a result of PRPG algorithm from a permutation sampled from uniform distribution.

PRPG algorithm. The PRPG algorithm starts with identity permutation of n elements π_0. In each round PRPG has access to a portion of n bits of the key stream (*i.e.,* in round l reads a portion of the key: $\mathcal{K}_l \in \{0,1\}^n$).

Leakage. We consider adversaries $\mathcal{A}$ which in the lth round can learn a fraction p of $\mathcal{K}_l$, namely $\Lambda_{p,l}(\mathcal{K}) = f_{l,p}(\mathcal{K}_l)$, where a function $f_{l,p}$ has range $\{*, \square\}^n$. More precisely, $f_{l,p} = a_1 a_2 \ldots a_n$ where $a_i \in \{*, \square\}$. If $a_i = \square$ then the adversary sees the corresponding bit of the key stream (learns $\mathcal{K}_{l,i}$) and if $a_i = *$ then the adversary does not learn the bit at position i.

Example 1. (Adversary view) Let $\mathcal{K}_3 = 101110$ and $f_3 = 110100 = \square\square * \square * *$ then adversary's view is: $\Lambda_3 = \boxed{1}\boxed{0} * \boxed{1} * *$. Which means that the adversary learns that $\mathcal{K}_{3,1} = 1, \mathcal{K}_{3,2} = 0, \mathcal{K}_{3,4} = 1$.

We restrict our analysis only to the adversaries for which each bit can be eavesdropped independently with probability p, *i.e.,* $1 - P(a_i = *) = P(a_i = \square) = p$. This means that number of leaking bit has $Bin(n,p)$ distribution in each round (np bits are leaking in each round on average). Let $\mathtt{view}_r$ denote the view of the adversary at the end of the algorithm:

$$\mathtt{view}_{r,p}(\mathcal{K}) = [\Lambda_{1,p}, \ldots, \Lambda_{r,p}].$$

The distinguishability game for the adversary is as follows:

Definition 1. *The* LEAK *indistinguishability experiment* $\mathsf{Shuffle}^{\mathsf{LEAK}}_{\mathsf{S},\mathcal{A}}(n,p,r)$*:*

1. S *is initialized with:*
   (a) *a key generated uniformly at random* $\mathcal{K} \sim \mathcal{U}(\{0,1\}^{rn})$*,*
   (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. *Adversary obtains leaked bits of the key* $\mathsf{view}_{r,p}(\mathcal{K})$*.*
4. *We set:*
   - $c_0 := \pi_{rand}$ *a random permutation from uniform distribution is chosen,*
   - $c_1 := \pi_r$*.*
5. *A challenge bit* $b \in \{0,1\}$ *is chosen at random, permutation* c_b *is sent to the Adversary.*
6. *Adversary replies with* b'
7. *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 based on the leakage she saw.

Definition 2. *A shuffling algorithm* S *generates indistinguishable permutations in the presence of leakage if for all adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

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

The above translates into:

Definition 3. *A shuffling algorithm* S *generates indistinguishable permutations in the presence of leakage if for all adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

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

2.2 Markov chains and rate of convergence

Consider ergodic Markov chain $\{X_k, k \geq 0\}$ 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 k. Stating some results about the *rate of convergence* of a chain to its stationary distribution means having some knowledge on some distance $dist$ (or a bound on it) between $\mathcal{L}(X_k)$ and ψ. By mixing time we mean the value of k making $dist$ small, since it depends on the measure of the distance we define it as

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

In our applications the state space is a set of permutations of $[n]$, *i.e.,* $\mathbb{E} := \mathcal{S}_n$ (thus $|\mathbb{E}| = n!$) and stationary distribution is a uniform one on $\mathbb{E}$ (we denote $\psi = \mathcal{U}(\mathbb{E})$).
Typically in literature the mixing time is defined for $dist$ being total variation distance, i.e.,

$$d_{TV}(\mathcal{L}(X_k), \mathcal{U}(\mathbb{E})) = \frac{1}{2} \sum_{\sigma \in \mathcal{S}_n} |Pr(X_k = \sigma) - Pr(\psi = \sigma)|,$$

which in our case is equivalent to:

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

Note however that knowing that d_{TV} is small for some k does not imply that $\left|Pr(X_k = \sigma) - \frac{1}{n!}\right|$ are "uniformly" small, *i.e.,* that it is of order $1/n!$. This is very important observation, since it means that $\tau_{mix}^{d_{TV}}(\varepsilon)$ is not an adequate measure of mixing time for our applications (i.e., indistinguishability given in Definition 2). Instead we consider so-called **separation distance** defined by

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

which is:

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

If $sep(\mathcal{L}(X_k), \mathcal{U}(\mathbb{E})) \leq \varepsilon$ for some k (i.e., we know $\tau_{mix}^{sep}(\varepsilon)$), then

$$\left| Pr(X_k = \sigma) - \frac{1}{n!} \right| \leq \frac{\varepsilon}{n!}, \tag{1}$$

what will be crucial for showing our results.

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.

Definition 4. *Random variable T is* ***Strong Stationary Time (SST)*** *if it is randomized stopping time for chain $\{X_k, k \geq 0\}$ 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 the separation distance (cf. [3])

$$sep(\mathcal{L}(X_k), \mathcal{U}(\mathbb{E})) \leq Pr(T > k). \tag{2}$$

We say that T is an optimal SST if $sep(\mathcal{L}(X_k), \mathcal{U}(\mathbb{E})) = Pr(T > k)$.
Remark. It is easy to show that separation distance is an upper bound on total variation distance, i.e. that $d_{TV}(\mathcal{L}(X_k), \mathcal{U}(\mathbb{E})) \leq sep(\mathcal{L}(X_k), \mathcal{U}(\mathbb{E}))$.

3 *RiffleSST*– leakage resilient shuffle

3.1 General pseudo-random permutation generator

We identify elements of $[n]$ with cards. We consider the following general pseudo-random permutation generator (PRPG) for generating a permutation of $[n]$. Initially we start with identity permutation π_0. At each round (step) we perform procedure `Shuffle` which takes the current permutation S and uses some "*randomness*" (based on secret key K) and updates the permutation. After r rounds the permutation is denoted by π_r. The algorithm stops depending on some stopping rule represented by procedure `StoppingRule`. We also model some leakage of information (to be specified later).

3.2 Description of *RiffleSST* algorithm

Roughly speaking, it uses card shuffling scheme corresponding to time reversal of Riffle Shuffle (see [4]).

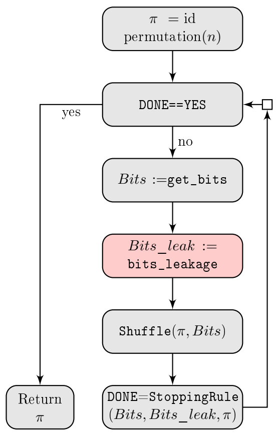
π = id permutation(n)
DONE==YES
yes
no
$Bits$:=get_bits
$Bits\_leak$:= bits_leakage
Shuffle(π, $Bits$)
Return π
DONE=StoppingRule ($Bits$, $Bits\_leak$, π)

Fig. 1. General pseudo-random permutation generator (PRPG)

RiffleSST

`get_bits`	:=	(not specified)
`bits_leakage`	:=	`get_bits_leakage_indep`
`Shuffle`	:=	`RiffleShuffle`
`StoppingRule`	:=	`StoppingRuleRiffle`

Fig. 2. Leakage resilient shuffle *RiffleSST* algorithm.

We do not specify here the details of `get_bits`, this should be a procedure which returns n bits from key K, can depend on current round number r, current permutation π, etc.

`RiffleShuffle` procedure performs the following: for given permutation of cards $\pi \in \mathcal{S}_n$ and given $Bits[i], i = 1, \ldots, n$ (think of assigning bit $Bits[i]$ to card on position i) we put all the cards with assigned bit 0 to the top *keeping* their relative ordering. Sample execution is given in Fig. 4: For example, for initial permutation $(1, 2, 3, 4, 5, 6)$ we assign bit 0 to cards 1,2 and 4, whereas we assign bit 1 to cards 3, 5 and 6. Thus, the resulting permutation is $(1, 2, 4, 3, 5, 6)$.

Leakage model. We assume that at each step and at each position i (independently) a value of $Bit[i]$ is leaking with probability p. Function $\mathsf{bits\_leakage}(p, n)$ generates n dimensional vector of zeros and ones. We assume that each coordinate is chosen independently being 1 with probability p and 0 with the remaining probability. Note that the number of leaking bits has $Bin(n, p)$ distribution, thus on average np bits are leak-

```
procedure RiffleShuffle
    Input permutation $\pi$, round $r$, $Bits$ (of length $n$)
    Output updated permutation $\pi$

    s0:=1
    s1:=sum($Bits$)
    tmp=vector($n$)
    for $i := 0$ to $n-1$ do
        card=S[$i$]
        if($Bits$[$i$]=1) do tmp[s1]=card; s1=s1+1
        else tmp[s0]=card; s0=s0+1
        end if
    end for
    $\pi$:=tmp
end procedure
```

ing. The leakage is modeled by `get_bits_leakage_indep` procedure (here $unif(0,1)$ denotes a random variable uniformly distributed on $[0,1]$).

```
procedure GET_BITS_LEAKAGE_INDEP
    Input $n, p$
    Output vector $leak$ of $n$ bits
    for $j = 1$ to $n$ do
        $leak(j) = \mathbf{1}\{unif(0,1) < p\}$
    end forreturn $leak$
end procedure
```

We simply run the algorithm for pre-defined number of steps expressed by procedure `StoppingRuleRiffle`

```
procedure StoppingRuleRiffle
    Input $n$, $p$ (leakage level), $\varepsilon$
    Output {YES,NO}

    if $r < \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\frac{1}{\varepsilon n!}\right)$ then return NO
    else return YES
    end if
end procedure
```

4 Proofs

As already mentioned, assuming random keys, the algorithm can be regarded as (time reversal of) Riffle Shuffle scheme. The idea is similar to approach presented in [13], following author's notation we will call a version of the algorithm with random keys as *idealized* one. Showing that after the execution of the algorithm the adversary has no non-negligible knowledge (in both, leakage and no-leakage version) corresponds to showing that after shuffling cards as many times as the number of steps of the algorithm, the resulting permutation is close to uniform one. In other words, proving the theorem reduces to studying the rate of convergence of corresponding Markov chains. However, the typical bounds on the rate of convergence involving total variation distance do not imply that the shuffling algorithm generates permutation which is indistinguishable from random permutation according to Definition 2. That is why we focus on bounds for separation distance what is achieved by using Strong Stationary Times technique.

```
RiffleSST*

get_bits      := get_bits_rand
bits_leakage  := get_bits_leakage_indep
Shuffle       := RiffleShuffle
StoppingRule  := StoppingRuleRifflePairs
```

Fig. 3. Idealized version of leakage resilient shuffle *RiffleSST*

Consider the *idealized* version of *RiffleSST* (call it `RiffleSST`*) which is defined by the specification from Figure 4. Roughly speaking, there are two differences compared to `RiffleSST`: i) in each round we take new n random bits. ii) instead of running it pre-defined number of steps, we use `StoppingRuleRifflePairs` as stopping rule. The stopping rule works as follows: Initially we set all $\binom{n}{2}$ pairs $(i, j), i, j = 1, \ldots, i < j$ as *not-marked*. (This can be simply represented as $\binom{n}{2}$-dimensional vector with all entries set to 0). Given the current permutation $\pi \in \mathcal{S}_n$ and $Bits[i], Bits\_leak[i], i = 1, \ldots, n$ we mark the pair (i, j) (or equivalently, we say that pair (i, j) is *updated*) if $(Bits[i] \oplus Bits[j] = 1)$ **and** $(Bits\_leak[i] \vee Bits\_leak[j] = 0)$ (*i.e.,* cards $S[i]$ and $S[j]$ were assigned different bits and none is leaking). Formally, this is given in `StoppingRuleRifflePairs` procedure.

```
procedure StoppingRuleRifflePairs
    Input set of already updated $pairs(i,j), i<j$, $Bits, Bits\_leak$
    Output {YES,NO}

    for each pair $(i,j)$ do
        if ($Bits[i] \oplus Bits[j] = 1$) and ($Bits\_leak[i] = Bits\_leak[j] = 0$) then
            mark pair $(i,j)$
        end if
    end for
    if all $\binom{n}{2}$ pairs are marked then return YES
    elsereturn NO
    end if
end procedure
```

The main ingredients of the proof of Theorem 1 are the following Lemma 1 and Theorem 2.

Lemma 1. *The resulting permutation of RiffleSST * has a uniform distribution over $\mathcal{S}_n$.*

Proof (of Lemma 1). For leakage level $p = 0$ the procedure `RiffleSST`* is exactly the Markov chain corresponding to *time-reversed Riffle Shuffle* card shuffling. At each step we consider all $\binom{n}{2}$ pairs (i,j) and we "mark" each pair if either card i was assigned 0 and card j was assigned 1, or vice-versa. Since these two events have equal probability, thus relative ordering of these two cards is random at such step. Let T be the first time all pairs are "marked". Then all the pairs are in relative random order and thus the permutation is also random. In other words, the distribution of X_k given $T = k$ is uniform. This means that the running time T of the algorithm is a Strong Stationary Time for riffle shuffle procedure and the distribution of the chain at time T is uniform.

Note that we exchangeably use the term "mixed" and "updated".

For general $p \in [0, 1)$ the situation is not much different: We update only pairs which are assigned different bits and such that both cards are not leaking. Thus, once the pair is updated it means that it is in random relative order and adversary has no knowledge about this order. After updating all the pairs she has no knowledge about relative order of all the pairs, thus, from her perspective, resulting permutation is random.

We note that the knowledge about the permutation can already "vanish" earlier (*i.e.,* before updating all the pairs), but it is for sure that time till updating all the pairs is enough.

Note that the no leakage version (*i.e.,*, when $p = 0$) of the algorithm can be written in a more compact way (similarly as in [4], the pairs are not involved there directly), however this notation lets us relatively easy extend the algorithm into a leakage resilient version.

Theorem 2. *Let $\{X_k\}_{k\geq 0}$ be the chain corresponding to RiffleSST*. The mixing time τ_{mix}^{sep} of the chain is given by*

$$\tau_{mix}^{sep}(\varepsilon) \leq \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\varepsilon^{-1}\right).$$

Proof (of Theorem 2). In Lemma 1 we showed that $T := \inf_k\{X_k = 0\}$ (*i.e.,* first moment when all pairs are updated) is an SST.
Let T_{ij} be the first time when cards i and j are updated. In one step, the probability that given pair will not be updated is $1 - \frac{1}{2}(1-p)^2 = \frac{2-(1-p)^2}{2}$. We have

$$Pr(T > k) = Pr\left(\bigcup_{1\leq i<j\leq n} \{T_{ij} > k\}\right) \leq \sum_{1\leq i<j\leq n} Pr(T_{ij} > k) =$$
$$= \textstyle\sum_{1\leq i<j\leq n} \left(1 - (1-p)^2 \cdot \frac{1}{2}\right)^k = \binom{n}{2}\left(\frac{2-(1-p)^2}{2}\right)^k.$$

For $k = \log_{\frac{2}{2-(1-p)^2}} \binom{n}{2} + \log_{\frac{2}{2-(1-p)^2}} \left(\varepsilon^{-1}\right)$ we have $Pr(T > k) \leq \varepsilon$, using (2) finishes the proof.

In Theorem 1 we perform *RiffleSST* for $r = \tau_{mix}^{sep}(\varepsilon n!)$ steps, what means that separation distance is less or equal to $\varepsilon n!$. From (1) we thus have that $\left|Pr(X_r = \sigma) - \frac{1}{n!}\right| \leq \varepsilon$ for any permutation σ. This together with Lemma 1 and Theorem 2 completes the proof of Theorem 2.

Remark 1. Note that if we replace τ_{mix}^{sep} with $\tau_{mix}^{d_{TV}}$ in Theorem 2, we could not conclude Theorem 1. This is because knowing that separation distance is smaller than ε is much stronger than knowing that total variation distance is smaller than ε, in particular (1) holds. (Note that typically, *e.g.,* coupling methods provide directly bounds on total variation distance). However, knowing that total $d_{TV}(\mathcal{L}(X_k), \mathcal{U}(\mathbb{E})) \leq \varepsilon$ implies (under some mild conditions - see Theorem 7 in [2]) that $sep(\mathcal{L}(X_{2k}), \mathcal{U}(\mathbb{E})) \leq \varepsilon$ what means that twice as many steps would be needed to achieve security claimed in Theorem 1.

5 Sample execution of *RiffleSST* algorithm

In Fig. 4 the sample execution with and without leakage is presented. At each step, the left column represents the current permutation, whereas the right one currently assigned bits. The leaking bits are represented by red-shaded boxes. In Fig. 5 updated pairs for leakage and no leakage versions are given.

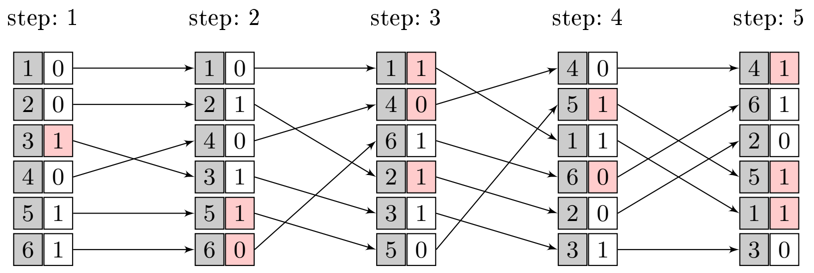

Fig. 4. Sample execution of our PRPG `RiffleSST`* algorithm for $n = 6$. Current permutation at each step is the left column (grayed) whereas right column are the chosen bits. Red shaded bits are leaking.

For example:

- *No leakage version*: At step 3. the current permutation is (1,4,6,2,3,5) and assigned bits are $Bits = (1, 0, 1, 1, 1, 0)$. Thus, *e.g.,* card on position 1 (1) and card on position 2 (4) have different bits assigned (respectively 1 and 0), thus this pair **(1,4)** is updated (it is bolded since it is first step when different bits were assigned for this pair)
- *Leakage version*: At the same step 3. bits assigned to cards 1, 4 and 2 are leaking. Thus all the pairs involving any of these cards are not considered, what results only in updating pairs **(3,5)**, **(5,6)**.

6 Conclusions

We presented the first analysis of the rate of convergence of the riffle-shuffle in the presence of leakage. We proved that no adversary can distinguish permutations produced by the *RiffleSST* (after enough number of steps – Theorem 1) from permutations sampled from uniform distribution.

	step: 1	step: 2	step: 3	step: 4	step: 5
No leakage	**(1,3), (1,5),(1,6) (2,3), (2,5), (2,6) (3,4), (4,5), (4,6)** **sum**$(pairs)$=9	**(1,2)**, (1,3), (1,5) **(2,4)**,(2,6) (3,4), **(3,6)**, (4,5) **(5,6)** **sum**$(pairs)$=13	**(1,4)**, (1,5) (2,4), (2,6) (3,4), **(3,5)**, (5,6) **sum**$(pairs)$=15 STOP		
Leakage	**(1,5), (1,6) (2,5), (2,6) (4,5), (4,6)** **sum**$(pairs)$=6	**(1,2), (1,3) (2,4) (3,4)** **sum**$(pairs)$=10	**(3,5), (5,6)** **sum**$(pairs)$=12	(1,2),**(1,4) (2,3)** (3,4) **sum**$(pairs)$=14	(2,6) **(3,6)** **sum**$(pairs)$=15 STOP

Fig. 5. Pairs "*mixed*" at each step of execution of PRPG given in Fig. 4. New pairs are **bolded**. The idealized algorithm (both, in non leakage and leakage version) stops when $\binom{6}{2} = 15$ pairs are mixed.

Open problems. Since the number of permutations is of $[n]$ is $n!$ the entropy of the uniform distribution on $[n]$ is $O(n \log n)$. The time-reversed riffle-shuffle is optimal (up to a constant factor) shuffle since its mixing-time is $O(\log n)$ and each round consumes n bits of randomness, so in total $O(n \log n)$ bits are used. In case of no leakage ($p = 0$) the mixing time of *RiffleSST* * - by Theorem 2 - is $O(n \log n)$ and thus is optimal.

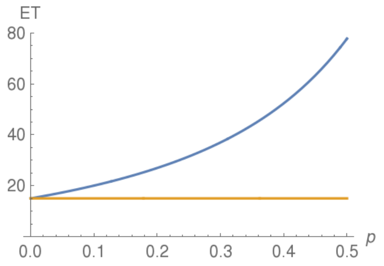

Fig. 6. Comparison of the expected number of rounds for $n = 256$ and Riffle-Shuffle (without leakage – orange) and *RiffleSST* with the leakage level p – blue.

Question: Is *RiffleSST* * optimal in the presence of leakage with rate $p > 0$? More precisely, can we use fewer bits than $O\left(n \log_{\frac{2}{1-(1-p)^2}} n\right)$ bits to achieve the security claimed in Theorem 1?

7 Open problems

- analiza w przypadku: wycieka waga hamminga slowa (oraz running time) - wydaje sie, ze powinno byc tak samo jak dla timing attacks(!) - czyli cos moze wycieka, ale dla riffle-shuffle i tak jest dobrze, bo permutacja jest jednostajna
- przy okazji hamming leakage uzasadnic to DPA dla implementacji DESa
- poprawic/uzasadnic, ze strong adversary nie moze nic zyskac ponad to co uzyskuje weak adwersary (dla riffle shuffle)

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*, 93(5):333–348, 1986.
3. David Aldous and Persi Diaconis. Strong Uniform Times and Finite Random Walks. *Advances in Applied Mathematics*, 97:69–97, 1987.
4. Persi Diaconis and Mehrdad Shahshahani. Generating a random permutation with random transpositions. *Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 57(2):159–179, 1981.
5. Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. Stealing Keys from PCs using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation.
6. Daniel Genkin, Adi Shamir, and Eran Tromer. RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis.
7. Marcin Gomułkiewicz, Marek Klonowski, and Mirosław Kutyłowski. Rapid Mixing and Security of Chaum's Visual Electronic Voting. pages 132–145.
8. VT T Hoang, Ben Morris, and Phillip Rogaway. An enciphering scheme based on a card shuffle. *Advances in Cryptology–CRYPTO 2012*, pages 1–13, 2012.
9. Markus Jakobsson, A Juels, and RL Rivest. Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking. *USENIX security symposium*, 2002.
10. Łukasz Krzywiecki, Mirosław Kutyłowski, Hubert Misztela, and Tomasz Strumiński. Private information retrieval with a trusted hardware unit - Revisited. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6584 LNCS, pages 373–386. Springer, Berlin, Heidelberg, oct 2011.
11. Michal Kulis, Paweł Lorek, and Filip Zagórski. Randomized stopping times and provably secure pseudorandom permutation generators. In *Mycrypt 2016: Paradigm-shifting Crypto*, pages 145—-167, 2016.
12. Pawel Lorek, Filip Zagorski, and Michal Kulis. Strong stationary times and its use in cryptography. *IEEE Transactions on Dependable and Secure Computing*, pages 1–1, 2017.
13. Ilya Mironov. (Not So) Random Shuffles of RC4. *Advances in Cryptology—CRYPTO 2002*, 2002.
14. Ben Morris, Philip Rogaway, and Till Stegers. How to Encipher Messages on a Small Domain Deterministic Encryption and the Thorp Shuffle. In *CRYPTO*, pages 1–19, 2009.
15. Ben Morris and Phillip Rogaway. Sometimes-Recurse Shuffle Almost Random Permutations in Logarithmic Expected Time. 2013.
16. 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, 1997. ACM Press.
17. Thomas Ristenpart and Scott Yilek. The mix-and-cut shuffle: small-domain encryption secure against N queries. In *Advances in Cryptology–CRYPTO 2013*, pages 392–409. Springer Berlin Heidelberg, 2013.

18. Jacob C. N. Schuldt; Ronald L. Rivest. Spritz—a spongy RC4-like stream cipher and hash function. Technical report, 2014.
19. Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. pages 30–46, 2005.
20. François Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8042 LNCS, pages 335–352. Springer Berlin Heidelberg, 2013.
21. 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.
22. Yanjiang Yang, Xuhua Ding, Robert H. Deng, and Feng Bao. An Efficient PIR Construction Using Trusted Hardware. In *Information Security*, pages 64–79. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
23. Yuval Yarom, Daniel Genkin, and Nadia Heninger. CacheBleed : A Timing Attack on OpenSSL Constant Time RSA. *CHES*, 2016.