Reverse Cycle Walking and Its Applications

Sarah Miracle(✉) and Scott Yilek

University of St. Thomas, St. Paul, USA
{sarah.miracle,syilek}@stthomas.edu

Abstract. We study the problem of constructing a block-cipher on a "possibly-strange" set $\mathcal{S}$ using a block-cipher on a larger set $\mathcal{T}$. Such constructions are useful in format-preserving encryption, where for example the set $\mathcal{S}$ might contain "valid 9-digit social security numbers" while $\mathcal{T}$ might be the set of 30-bit strings. Previous work has solved this problem using a technique called cycle walking, first formally analyzed by Black and Rogaway. Assuming the size of $\mathcal{S}$ is a constant fraction of the size of $\mathcal{T}$, cycle walking allows one to encipher a point $x \in \mathcal{S}$ by applying the block-cipher on $\mathcal{T}$ a small *expected* number of times and $O(N)$ times in the worst case, where $N = |\mathcal{T}|$, without any degradation in security. We introduce an alternative to cycle walking that we call *reverse cycle walking*, which lowers the worst-case number of times we must apply the block-cipher on $\mathcal{T}$ from $O(N)$ to $O(\log N)$. Additionally, when the underlying block-cipher on $\mathcal{T}$ is secure against $q = (1-\epsilon)N$ adversarial queries, we show that applying reverse cycle walking gives us a cipher on $\mathcal{S}$ secure even if the adversary is allowed to query all of the domain points. Such fully secure ciphers have been the the target of numerous recent papers.

Keywords: Format-preserving encryption · Small-domain block ciphers · Markov chains

1 Introduction

Suppose we have sets $\mathcal{S}$ and $\mathcal{T}$, with $\mathcal{S}$ a subset of $\mathcal{T}$. Typically, in this paper, the larger set $\mathcal{T}$ will be $\{0, \ldots, 2^n - 1\}$ for some integer n, while the smaller set $\mathcal{S}$ will be an arbitrary set for which we only assume we know how to efficiently test membership. The central problem we study in this paper is, given a cipher with domain $\mathcal{T}$, how can we construct a cipher with domain $\mathcal{S}$.

Format-Preserving Encryption. The above problem arises when constructing *format preserving encryption* (FPE) [1,2,4] schemes for encrypting credit cards numbers, social security numbers, and other relatively short data objects. Suppose we have a customer database containing millions of US social security numbers (SSNs). SSNs are 9 decimal digit numbers with numerous additional restrictions (e.g., the first three digits may not be 666). Now suppose we later decide we need to encrypt the SSNs. One approach would be to use a standard block cipher like AES, representing the SSN as a 30-bit number and then padding

J.H. Cheon and T. Takagi (Eds.): ASIACRYPT 2016, Part I, LNCS 10031, pp. 679–700, 2016.
DOI: 10.1007/978-3-662-53887-6_25

with 0s before encrypting. The resulting ciphertext, however, would have a significantly different format from the original, unencrypted numbers. This could in turn require significant changes to the customer database, as well as to the hardware and software that process the SSNs. For this reason, it is desirable to have format-preserving encryption schemes, in which ciphertexts have the same format as plaintexts. A FPE scheme for SSNs would thus have ciphertexts that are 9 decimal digit numbers with the same restrictions as unencrypted SSNs.

Cycle Walking. A number of recent works [3,11,16–18] describe efficient, provably secure small-domain block ciphers for enciphering either bitstrings or, in most cases, points in the more general domain $\{0,\ldots,N-1\}$. This is already sufficient for many FPE applications. However, if the desired domain for a particular FPE application is not as simple as bitstrings of some length or integers up to N, then these ciphers alone are not sufficient. If we only assume that we can efficiently test membership in our target domain set $\mathcal{S}$,[1] then one approach to the problem is to find a cipher on a larger set $\mathcal{T}$ and transform it into a cipher on the smaller set $\mathcal{S}$. In the case of valid SSNs, for example, we might let the larger set $\mathcal{T}$ be 30-bit strings, since $10^9 < 2^{30}$ and we have many block ciphers that can encipher 30-bit strings. The canonical way to transform a cipher on the more general set $\mathcal{T}$ into a cipher on a subset $\mathcal{S}$ while maintaining the same level of security is to use *cycle walking*. Cycle walking is a folklore technique first formally analyzed by Black and Rogaway [4] that works as follows. Suppose π is a permutation with domain $\mathcal{T}$ and we wish to use it to map a point $x \in \mathcal{S}$ to another point in $\mathcal{S}$. We first compute $\pi(x)$ and test if the result is in $\mathcal{S}$. If so, we map point x to $\pi(x)$. If the result is not in $\mathcal{S}$, we apply π again, computing $\pi(\pi(x))$ and again testing whether or not the result is in $\mathcal{S}$. We repeat this process until we get a point in $\mathcal{S}$. Let CW_π denote this cycle walking algorithm. Black and Rogaway showed that cycle walking maintains the security of π, and in particular showed that if cycle walking is applied to a CCA-secure cipher, then the resulting cipher is also CCA-secure.

If we are unlucky, we may have to apply π numerous times before finally reaching a point in $\mathcal{S}$.[2] In fact, if we consider the *worst-case* running time of cycle walking, we might have to evaluate the permutation $\Theta(N)$ times. Yet, the *expected* running time is much better; if the size of $\mathcal{S}$ is at least half the size of $\mathcal{T}$ and if π is a randomly-chosen permutation on $\mathcal{T}$, then the expected number of times CW_π will need to evaluate π on a particular point is at most 2.

[1] If a set has an efficient way to rank and unrank elements, then instead one can apply the rank algorithm and then a cipher on $\{0,\ldots,|\mathcal{S}|-1\}$. This is the case, for example, with regular languages described by a DFA [1]. For other languages, and even for regular languages described by a regular expression, the situation is more complicated. See [14,15] for more details. Nevertheless, in the current paper we are concerned with more general sets where only testing set membership is assumed to be efficient.

[2] We are guaranteed to eventually land back in the set $\mathcal{S}$, since permutations are made up of cycles and, if we don't hit another point in $\mathcal{S}$ first, we will cycle back around to the same point we started with.

It will be helpful to also examine the cycle structure of CW_π compared to π. Let $\mathcal{S}$ and $\mathcal{T}$ be as they were defined above, and let π again be a permutation on the larger set $\mathcal{T}$. Recall that permutations are made up of disjoint cycles, so our chosen permutation π is made up of disjoint cycles each with points from $\mathcal{T}$. Suppose that one of these cycles is $(t_1\ s_1\ s_2\ s_3\ t_2\ s_4\ s_5\ t_3\ t_4)$, where the s points are all from $\mathcal{S}$ and the t points are from $\mathcal{T} \setminus \mathcal{S}$. Now consider what happens when $\mathsf{CW}_\pi(s_3)$ is evaluated. Notice that $\pi(s_3) = t_2$, so we need to evaluate $\pi(\pi(s_3)) = s_4$. Thus, $\mathsf{CW}_\pi(s_3) = s_4$. In terms of the cycle structure, evaluating $\mathsf{CW}_\pi(s_3)$ corresponds to walking to the right in the cycle from s_3 until we hit another point in $\mathcal{S}$. Similarly, $\mathsf{CW}_\pi(s_5) = s_1$, which we can see since walking to the right from s_5 brings us to t_3, t_4, t_1 (after looping around to the front), and then finally s_1. We can thus determine the cycle structure of CW_π simply by erasing the t points from all of the cycles in π, meaning the cycle above becomes $(s_1\ s_2\ s_3\ s_4\ s_5)$.

A Closer Look at Expected Time. The small expected running time of cycle walking makes it an attractive option in practice for FPE. Yet, from a theoretical perspective, the fact that we do not know how to build permutations for arbitrary sets with worst-case running time better than $\Theta(N)$ is unsatisfying.

Finding alternative algorithms that do not run in expected time is not just an important theoretical question. From a practical perspective, in addition to the unpredictability of execution times potentially bothering practitioners, there is the danger that expected-time cryptographic algorithms can leak timing information that can be exploited by an adversary in an attack. Starting with the work of Kocher [12], there have been numerous examples of how such timing information can lead to subtle and damaging attacks on cryptographic protocols. Thus, generally, it would seem preferable to have cryptographic algorithms whose running time does not vary across different inputs.

Somewhat counter to this, Bellare, Ristenpart, Rogaway, and Stegers [1] analyzed the potential negative effects of the timing information leaked by cycle walking and concluded that the leakage is not damaging. Yet, their result is in a specific model where the adversary has access to the ciphertexts in addition to the number of cycle walking steps needed, which they call the cycle length.[3] This, however, does not preclude the possibility of other scenarios in which this timing information could be useful. As one simple example, suppose an adversary observes the time it takes to encipher and later learns the corresponding plaintext. If, at a later point, the adversary again observes the time it takes to encipher a point then this timing information can reveal whether or not the same point was enciphered without the adversary ever needing to observe any ciphertexts. Depending on the specific scenario and application, this information could be damaging.

Reverse Cycle Walking. We now describe our main result: an alternative to cycle walking with substantially better worst-case running time that does not

[3] Specifically, they show that in a PRP security game the adversary gets no benefit from learning the cycle length in addition to the ciphertext on an encryption query.

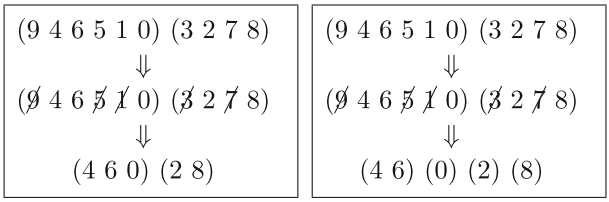

Fig. 1. Example of how one round of reverse 2-cycle walking differs from regular cycle walking. In this example, $\mathcal{T} = \{0, \ldots, 9\}$ and $\mathcal{S}$ are the even numbers in $\mathcal{T}$. **Left:** the effect regular cycle walking has on the cycle structure of the permutation. **Right:** the effect of one round of reverse 2-cycle walking on the cycle structure.

vary based on the input. Towards this, a first attempt might be to try to apply cycle walking, but somehow "cut-off" the algorithm if it is taking too long, since often with an expected-time algorithm one can simply stop the algorithm early and possibly introduce a small error. Unfortunately, it is not clear how to make this approach work. If we are evaluating $\mathsf{CW}_\pi(x)$ and walking through a long sequence of points in $\mathcal{T}\backslash\mathcal{S}$, we cannot just cut off the algorithm because we need to construct a permutation, and thus require a unique point in $\mathcal{S}$ to map x to. Because of this difficulty, we introduce an alternative to cycle walking we call *reverse cycle walking*.

Let $\mathcal{S}$, $\mathcal{T}$, and π be as they are defined above, and suppose again π has a cycle $(t_1\ s_1\ s_2\ s_3\ t_2\ s_4\ s_5\ t_3\ t_4)$. As in traditional cycle walking, under reverse cycle walking s_1 is mapped to s_2 and s_2 is mapped to s_3. Where reverse cycle walking differs from traditional cycle walking is when a point in $\mathcal{S}$ is mapped outside of $\mathcal{S}$; this is the case for s_3, which is mapped under π outside of $\mathcal{S}$ to t_2. To determine where s_3 should be mapped to, reverse cycle walking walks in the reverse direction, to the *left*, until a point outside of $\mathcal{S}$ is encountered; the last point encountered that is in $\mathcal{S}$ will be where s_3 is mapped. So in the case of the current cycle, if we wish to know where s_3 will be mapped, we walk to the left to s_2 and then finally to s_1. Since walking to the left any farther would result in a point outside of $\mathcal{S}$, reverse cycle walking stops here and maps s_3 to s_1. Similarly, s_5 would be mapped to s_4. The cycle structure that results from applying reverse cycle walking is thus $(s_1\ s_2\ s_3)(s_4\ s_5)$.

Notice that reverse cycle walking, as just described, will still have poor worst-case running time and considerably better expected running time, much like traditional cycle walking. The main advantage now, though, is that we can consider variants of reverse cycle walking that "cut-off" the algorithm early and significantly reduce the worst-case running time. Specifically, when reverse t-cycle walking is applied to permutation π, any sequence of at most t points from $\mathcal{S}$ that appear consecutively in a cycle of π sandwiched between points from $\mathcal{T}\backslash\mathcal{S}$ will become a cycle. Any points in $\mathcal{S}$ that do not have this property are simply mapped to themselves.

For the rest of the paper, we focus on perhaps the simplest version of this idea, reverse 2-cycle walking. If π has in some cycle $(\ldots t\ s\ s'\ t' \ldots)$,

meaning two consecutive points from $\mathcal{S}$ sandwiched between points from $\mathcal{T}\backslash\mathcal{S}$, then under reverse 2-cycle walking (denoted RCW_π) $(s\ s')$ becomes a cycle, meaning $\mathsf{RCW}_\pi(s) = s'$ and $\mathsf{RCW}_\pi(s') = s$. When reverse 2-cycle walking is applied to our example above where π has cycle $(t_1\ s_1\ s_2\ s_3\ t_2\ s_4\ s_5\ t_3\ t_4)$, the resulting permutation will have cycles $(s_1)(s_2)(s_3)(s_4\ s_5)$. Notice that because s_1, s_2, and s_3 represented more than two consecutive points from $\mathcal{S}$, they were simply mapped to themselves. On the other hand, s_4 and s_5 were two consecutive points from $\mathcal{S}$ sandwiched between points outside of $\mathcal{S}$, so they are swapped. (For technical reasons, as we will see later in the paper, we will additionally flip a coin to see if these points are actually swapped or not.) The code of the reverse 2-cycle walking transformation can be found in Fig. 2 in Sect. 3. (Note that the transformation is an involution.) Another example illustrating how reverse 2-cycle walking compares to traditional cycle walking can be seen in Fig. 1.

Worst-Case Running Time of Reverse 2-Cycle Walking. In our scenario above, even if π is a random permutation on $\mathcal{T}$, RCW_π will clearly not be close to a random permutation on $\mathcal{S}$; many points are mapped to themselves (i.e., $\mathsf{RCW}_\pi(x) = x$). Thus, with reverse 2-cycle walking, we need to repeat the procedure for multiple rounds with independently chosen permutations π. The question then becomes how many rounds of RCW are needed before the resulting permutation on $\mathcal{S}$ is close to random.

To answer this question, we show that when π is a randomly chosen permutation on $\mathcal{T}$ and when the size of $\mathcal{S}$ is a constant fraction of the size of $\mathcal{T}$, then reverse 2-cycle walking yields a *matching exchange process* (MEP), first defined and analyzed by Czumaj and Kutylowski [9]. A MEP proceeds in rounds to mix N points, where in each round a random matching of some size is chosen and then a coin is flipped for each pair in the matching to decide whether its points should be swapped. Notice that this is exactly how multiple rounds of reverse 2-cycle walking proceed: in any given round, each point in $\mathcal{S}$ is either randomly paired with another point in $\mathcal{S}$, or it is mapped to itself and is not part of the matching for that round.

To analyze MEPs, Czumaj and Kutylowski used *non-Markovian delayed path coupling*, an extension of the well-known path coupling technique [5] in the area of Markov chains, to show that a matching exchange process will mix N points in $O(\log N)$ rounds. Since we show reverse 2-cycle walking yields a MEP, directly applying their result gives us a way to construct an almost-random permutation on an arbitrary set with worst-case running time $\Theta(t(N) \cdot \log N)$, where $t(N)$ is the time it takes to apply permutation π on $\mathcal{T}$. Recall that with traditional cycle walking, we get worst-case running time $\Theta(t(N) \cdot N)$, so our result is a significant improvement.

Since an asymptotic result is of limited practical value in the setting where cycle walking seems most useful, that of small-domain encryption for FPE, we also give concrete bounds relating the number or rounds of reverse 2-cycle walking to the CCA-advantage of an adversary attacking the encryption scheme. Unfortunately, because the Czumaj and Kutylowski paper targeted asymptotic results, their proof does not give explicit constants. To overcome this difficulty,

we give new proofs of two key lemmas from CK's proof in order to minimize the constants for our setting where N is perhaps 2^{30}.

Full Security from Reverse 2-Cycle Walking. Fully secure block ciphers, which are block ciphers that look like random permutations even to an adversary querying all N domain points, have been the target of many recent papers [10,16,18] on small-domain encryption. While all of these recent results are based on a recursive shuffling technique from [8], we instead take a different approach and show that reverse 2-cycle walking can be used to achieve full security. In particular, we show that in certain situations we can take a cipher on a larger set $\mathcal{T}$ that is *not* fully secure and apply reverse 2-cycle walking to get a fully secure cipher on the smaller set $\mathcal{S}$.

To help explain this result in more detail, suppose we wish to construct a fully secure block cipher E_{full} with domain $\{0,\ldots,N-1\}$ and further suppose we have another block cipher E_{part} with a larger domain $\{0,\ldots,2N-1\}$ and which is indistinguishable from a random permutation as long as the adversary only queries half the domain points. (Swap or Not [11] would be an example of such a cipher, which we call partially secure.) Notice that E_{part}, with domain size $2N$, will be secure against N queries, which is the same *quantity* of queries we want E_{full} to be secure against. But how should E_{full} use E_{part} to encipher points in $\{0,\ldots,N-1\}$? To encipher a point $x \in \{0,\ldots,N-1\}$, we could simply apply E_{part} to x. But since E_{part} has a larger domain, $E_{\text{part}}(x)$ might not be in $\{0,\ldots,N-1\}$. Czumaj [6] recently considered something similar and suggested using E_{part} to shuffle all of the points $\{0,\ldots,2N-1\}$ and then "remove" the points outside of $\{0,\ldots,N-1\}$. Unfortunately, it's not clear how to efficiently implement this "remove" step.

Another idea might be to use traditional cycle walking to always make sure we can map a point $x \in \{0,\ldots,N-1\}$ back into the same set. Unfortunately, proving this secure appears difficult, since in a reduction each of the N adversarial queries made while attacking E_{full} could result in many queries to E_{part}. Thus, in the reduction, our adversary attacking E_{part} would likely need to query nearly all points in $\{0,\ldots,2N-1\}$, many more queries than E_{part} is assumed secure against.

Instead, we propose using reverse 2-cycle walking. Using our set names from earlier in the introduction, let $\mathcal{S} = \{0,\ldots,N-1\}$ and let $\mathcal{T} = \{0,\ldots,2N-1\}$. Let E_{part} be a block cipher with domain $\mathcal{T}$. Then reverse 2-cycle walking has the following key property: if we evaluate E_{part} on every point $x \in \mathcal{S}$, then this gives us enough information to determine $\mathsf{RCW}_{E_{\text{part}}}(x)$ for every $x \in \mathcal{S}$. In other words, we never need to evaluate E_{part} on any point outside of $\{0,\ldots,N-1\}$. This property allows the reduction to go through, giving us a fully secure cipher.

2 Preliminaries

Notation. For any set $\mathcal{X}$, let $\mathsf{Perms}(\mathcal{X})$ be the set of all permutations $\pi : \mathcal{X} \to \mathcal{X}$. For sets $\mathcal{X}$ and $\mathcal{Y}$, let $\mathsf{Funs}(\mathcal{X},\mathcal{Y})$ be the set of all functions $f : \mathcal{X} \to \mathcal{Y}$. For set $\mathcal{X}$, let $x \xleftarrow{\$} \mathcal{X}$ denote choosing x uniformly at random from $\mathcal{X}$.

Mixing Time. The time a Markov chain $\mathcal{M}$ takes to converge to its stationary distribution μ is measured in terms of the distance between μ and $\mathcal{P}^t$, the distribution at time t. Let $\mathcal{P}^t(x, y)$ be the t-step transition probability and Ω be the state space. The *mixing time* of $\mathcal{M}$ is $\tau_{\mathcal{M}}(\epsilon) = \min\{t : ||\mathcal{P}^{t'} - \mu|| \leq \epsilon, \forall t' \geq t\}$, where $||\mathcal{P}^t - \mu|| = \max_{x \in \Omega} \frac{1}{2} \sum_{y \in \Omega} |\mathcal{P}^t(x, y) - \mu(y)|$ is the *total variation distance* at time t.

Block Ciphers and Their Security. A block cipher is a family of functions $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$, with $\mathcal{K}$ a finite set called the key space and $\mathcal{M}$ a finite set called the domain or message space. For every $K \in \mathcal{K}$, the function $E_K(\cdot) = E(K, \cdot)$ is a permutation. Let $E^{-1} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ be the inverse block cipher. We will typically let N denote $|\mathcal{M}|$, the number of elements in the domain. Thus, when $\mathcal{M} = \{0, 1\}^n$, $N = 2^n$.

We will consider block cipher security against chosen-ciphertext attack (CCA), often referred to as strong-PRP security. Given block cipher $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ and adversary A, the cca-advantage of A against E is defined to be

$$\mathbf{Adv}^{\mathrm{cca}}_{E}(A) = \mathbf{P}\left(A^{E(K,\cdot),E^{-1}(K,\cdot)} \Rightarrow 1\right) - \mathbf{P}\left(A^{\pi(\cdot),\pi^{-1}(\cdot)} \Rightarrow 1\right),$$

where the first probability is over the choice of K and the coins of A, and the second probability is over the choice of π from $\mathsf{Perms}(\mathcal{M})$ and the coins of A. In words, the adversary A tries to determine which "world" he is in, where he is either in a world where he is given access to the block cipher and its inverse, or in a world where he is given access to a random permutation and its inverse. If an adversary A is given oracle access to an algorithm $\mathcal{O}$ and its inverse $\mathcal{O}^{-1}$, we will sometimes write $A^{\pm\mathcal{O}(\cdot)}$ as shorthand for $A^{\mathcal{O}(\cdot),\mathcal{O}^{-1}(\cdot)}$.

Pseudorandom Functions. Let $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a family of functions with key space $\mathcal{K}$. The prf-advantage of an adversary A against F is defined to be $\mathbf{Adv}^{\mathrm{prf}}_{F}(A) = \mathbf{P}\left(A^{F(K,\cdot)} \Rightarrow 1\right) - \mathbf{P}\left(A^{\rho(\cdot)} \Rightarrow 1\right)$, where the first probability is over the choice of key K and the coins of A, and the second probability is over the choice of ρ from $\mathsf{Funs}(\mathcal{X}, \mathcal{Y})$ and the coins of A. In words, the adversary tries to determine through oracle queries whether it is interacting with the keyed function F or a random function chosen from all functions from $\mathcal{X}$ to $\mathcal{Y}$.

Cycle Walking. This paper focuses on the problem of using permutations on a set $\mathcal{T}$ to build a permutation on a smaller set $\mathcal{S} \subseteq \mathcal{T}$. Specifically, we will be interested in the scenario where $N_S = |\mathcal{S}|$ is a constant fraction of $N_T = |\mathcal{T}|$ (e.g., $2 \cdot N_S = N_T$). Black and Rogaway [3] analyzed a folklore technique for this called *cycle walking*, or *cycling*. Given a permutation π on $\mathcal{T}$, let the cycle walking transformation of π with target set $\mathcal{S}$ be function $\mathsf{CW}_\pi : \mathcal{S} \to \mathcal{S}$ defined as follows

Algorithm $\mathsf{CW}_\pi(x)$:
do
  $x \leftarrow \pi(x)$
while $(x \notin \mathcal{S})$
Return x

In words, cycle walking continues to apply permutation π until it finally gets a point in set $\mathcal{S}$. Cycle walking can also be applied to block ciphers. Notationally, if $E : \mathcal{K} \times \mathcal{T} \rightarrow \mathcal{T}$ is a block-cipher on $\mathcal{T}$, then we will let $\bar{E} : \mathcal{K} \times \mathcal{S} \rightarrow \mathcal{S}$ be the block cipher that, on input K and x, computes $\mathsf{CW}_{E_K}(x)$.

A key fact about cycle walking, argued by Black and Rogaway, is that if π is a random permutation on $\mathcal{T}$, then CW_π will be a random permutation on $\mathcal{S}$. While this is an information-theoretic result, Black and Rogaway also briefly argued it can be used to show the cycle walking transformation preserves cca security as well, which we formalize as follows:

Lemma 1 (Black-Rogaway). *Let $\mathcal{S} \subseteq \mathcal{T}$ be such that $|\mathcal{S}| \geq (1/2)|\mathcal{T}|$ and let let $E : \mathcal{K} \times \mathcal{T} \rightarrow \mathcal{T}$ be a block cipher on $\mathcal{T}$, and $\bar{E} : \mathcal{K} \times \mathcal{S} \rightarrow \mathcal{S}$ the block-cipher resulting from applying cycle walking to E with target set $\mathcal{S}$. Let A be an adversary making q queries, then*

$$\mathbf{Adv}^{\mathrm{cca}}_{\bar{E}}(A) \leq \mathbf{Adv}^{\mathrm{cca}}_{E}(B)$$

where adversary B makes at most an expected $2q$ queries.

As we explained in the introduction, cycle walking has small expected running time, but it has significantly worse worst-case running time. Additionally, the theorem above bounds the advantage of an adversary against $\bar{E}$ by the advantage of an adversary that makes and expected number of oracle queries.

3 Reverse 2-Cycle Walking

We now detail the reverse 2-cycle walking algorithm. Again, for sets $\mathcal{S} \subseteq \mathcal{T}$, let $N_S = |\mathcal{S}|$, $N_T = |\mathcal{T}|$, and assume N_S is a constant fraction of N_T; $c \cdot N_S = N_T$ (e.g., $2 \cdot N_S = N_T$). Suppose we have permutation $\pi : \mathcal{T} \rightarrow \mathcal{T}$ with π^{-1} its inverse. Also suppose we have a function $\mathsf{B} : \mathcal{S} \rightarrow \{0,1\}$. The reverse 2-cycle walking transformation is a function $\mathsf{RCW}_{\pi,\mathsf{B}} : \mathcal{S} \rightarrow \mathcal{S}$ defined in Fig. 2.[4]

To understand the algorithm it is helpful to consider the cycle structure of $\mathsf{RCW}_{\pi,\mathsf{B}}$ (a permutation on $\mathcal{S}$) as compared to π. In $\mathsf{RCW}_{\pi,\mathsf{B}}$, points in $\mathcal{S}$ are mapped to themselves unless they are contained in a cycle (in π) where exactly two points in $\mathcal{S}$ are surrounded by points in $\mathcal{T} - \mathcal{S}$. For example, if π contains the cycle $(s_1 s_2 t_1 t_2 s_3 s_4 t_4 s_5)$ where the s_i's are in $\mathcal{S}$ and the t_i's are in $\mathcal{T} - \mathcal{S}$, then the resulting permutation on $\mathcal{S}$ will contain the cycles $(s_1)(s_2)(s_3 s_4)(s_5)$. Note that for simplicity of analysis, if π contains the cycle $(s_1 s_2)$ then the resulting permutation will contain the cycles $(s_1)(s_2)$, whereas $(s_1 s_2 t_1)$ will result in

[4] It should be noted that the pseudocode in Fig. 2 is written for ease of understanding and, if implemented exactly as written, could leak timing information about the input. An actual implementation would use standard techniques to ensure each path through the code results in the same number of operations. Additionally, if the underlying cipher π has different timings in the forward and backward directions, then both $\pi(y)$ and $\pi^{-1}(z)$ would need to be computed regardless of the input point x.

Algorithm $\mathsf{RCW}_{\pi,\mathsf{B}}(x)$:
$u \leftarrow \perp\ ;\ v \leftarrow \perp$
$y \leftarrow \pi(x)\ ;\ z \leftarrow \pi^{-1}(x)$
if $y \in \mathcal{S}$ and $z \notin \mathcal{S}$ and $\pi(y) \notin \mathcal{S}$:
  $u \leftarrow x\ ;\ v \leftarrow y$
  $b \leftarrow \mathsf{B}(u)$
  if $b = 1$ return v else return u
else if $y \notin \mathcal{S}$ and $z \in \mathcal{S}$ and $\pi^{-1}(z) \notin \mathcal{S}$:
  $u \leftarrow z\ ;\ v \leftarrow x$
  $b \leftarrow \mathsf{B}(u)$
  if $b = 1$ return u else return v
else
  return x

Fig. 2. The reverse 2-Cycle walking algorithm

$(s_1 s_2)$. Additionally, the function B has the effect of only including each 2-cycle in the final permutation on S with probability 1/2. This is currently necessary for our analysis but we believe with further work this function can be removed.

When π and B are clear from context, we will sometimes write just $\mathsf{RCW}(x)$. Note that each point x is either mapped to itself (i.e., $\mathsf{RCW}(x) = x$), or is part of a 2-cycle (i.e., there is a $y \neq x$ s.t. $\mathsf{RCW}(x) = y$ and $\mathsf{RCW}(y) = x$). Notice also that the algorithm is its own inverse (an involution).

Given permutations $\pi_1, \pi_2, \ldots, \pi_k$ all on $\mathcal{T}$, and functions $\mathsf{B}_1, \ldots, \mathsf{B}_k$ from $\mathcal{S}$ to $\{0,1\}$, we denote by $\mathsf{RCW}^k_{(\pi_1,\ldots,\pi_k),(\mathsf{B}_1,\ldots,\mathsf{B}_k)}$ the composition $\mathsf{RCW}_{\pi_1,\mathsf{B}_1} \circ \ldots \circ \mathsf{RCW}_{\pi_k,\mathsf{B}_k}$. When the permutations π_i and functions B_i are clear from the context, we will often write RCW^k. The inverse of RCW^k will simply apply the rounds in reverse order, since the RCW algorithm above is its own inverse.

The rest of the paper focuses on the security of the reverse 2-cycle walking transformation. The next section gives an information theoretic result, bounding the mixing time of the Markov chain that results from applying a number of rounds of reverse 2-cycle walking where in each round we use a randomly chosen underlying permutation π_i and function B_i. In Sect. 5, we analyze the cca security of reverse 2-cycle walking when the underlying permutations on $\mathcal{T}$ are cca-secure and the round functions are implemented with a pseudorandom function. Finally, in Sect. 6, we show that reverse 2-cycle walking can be used to build fully secure block ciphers.

4 Bounding the Mixing Time

Here, we focus on how many rounds of reverse 2-cycle walking are needed before the resulting permutation on $\mathcal{S}$ is "close"to random. We consider the ideal case where at each round the underlying permutation π_i and function B_i are chosen uniformly at random and bound the mixing time of the underlying Markov

chain. To do this, we use a technique called delayed path coupling, introduced by Czumaj and Kutylowski [9] to analyze what they call a matching exchange protocol. They are interested in studying a class of Markov chains for sampling permutations of N points where at each step a number κ is chosen according to some distribution, then a matching of size κ is chosen uniformly from all matchings of size κ, and finally the points corresponding to each pair in the matching are each independently swapped (or not) with probability 1/2. Assuming the expected size of the matching at each step is $\Theta(N)$ they show that after $\Theta(\log(N))$ steps the variation distance is $O(1/N)$. If you consider the effect of the RCW algorithm on all elements in N_S, a single step of the algorithm is equivalent to selecting a matching M_i on N_S (since we only consider 2-cycles) according to some distribution and then swapping each pair in the matching with probability 1/2. Claim 1, which we prove below, implies that RCW is a matching exchange protocol with $\mathbf{E}[\kappa] \leq \frac{(c-1)^2 N_S}{c^3}$, where $c = N_T/N_S$. Given this, we can apply Czumaj and Kutylowski's results directly to bound the variation distance. Specifically, their result implies that there exist constants k_1, k_2 such that for $k = k_1 \log(N_S)$, $||\nu_{\text{rcw}^k} - \mu_s|| \leq \frac{k_2}{N_S}$ where ν_{rcw^k} is the distribution after k steps of the RCW algorithm and μ_s is the uniform distribution on permutations of the elements in N_S. However, their result does not explicitly compute the constants. Although we use many of the general ideas from their proof we not only give explicit constants but we provide new proofs of two key lemmas in order to provide a bound that is reasonable in our context and customized for the RCW algorithm. Despite these changes, we believe this is just a starting point and a further reduction of the constants is possible. We begin by providing an overview of the approach and then give a detailed proof focusing on our modifications. For additional information on the Markov chain analysis techniques used in this section please see [13,19].

We will first show that the Markov chain that results from repeatedly applying RCW is ergodic and it's stationary distribution is the uniform distribution. In a single step of the RCW algorithm there is a non-zero probability that we select any single transposition (i.e., $(s_i, s_j)(s_1)(s_2)\ldots$). It is well known that transpositions (swapping any two elements) connect the set of all permutations (see e.g., [13]) and thus RCW connects $\mathsf{Perms}(\mathcal{S})$ (the set of all permutations on $\mathcal{S}$). Additionally, RCW is aperiodic since there is a non-zero probability that no changes are made and thus ergodic. It is also relatively straightforward to see that RCW is symmetric (i.e., for all pairs of permutations (x, y), $\mathcal{P}(x, y) = \mathcal{P}(y, x)$, where $\mathcal{P}(x, y)$ is the probability of moving from x to y in one step of RCW). Combining these implies that the stationary distribution of RCW is the uniform distribution as desired (see e.g., [13]).

In order to bound the mixing time of the matching exchange process, Czumaj and Kutylowski use a technique they call *delayed path coupling* which is an extension of coupling and path coupling, both well-known techniques in the Markov chain community. A *coupling* of a Markov chain $\mathcal{M}$ with state space Ω is a joint Markov process on $\Omega \times \Omega$ such that the marginals each agree with $\mathcal{M}$ and, once the two coordinates coalesce, they move in unison. The coupling

time (or expected time until the two coordinates coalesce) can be used to upper bound the mixing time. *Path coupling*, introduced by Bubley and Dyer, simplifies this approach by considering only a subset U of the joint state space $\Omega \times \Omega$ of a coupling [5]. By considering an appropriate metric Δ on Ω, proving that the two marginal chains, if in a joint configuration in subset U, get no farther away in expectation after one iteration is sufficient to give a polynomial bound on the mixing time. For our argument we will define the distance between two configurations $\Delta(X, Y)$ as the minimum number of transpositions (swapping two points) needed to go from X to Y and U as the set of all pairs of configurations that differ by a single transposition $\Delta(X, Y) = 1$. Using this definition of U it is relatively straightforward to use path coupling to show that the mixing time is $O(N_S \log N_S)$. However for our application this bound is not sufficient and we require more complex techniques.

In delayed path coupling we consider the change in distance between two processes over more than just a single step. We bound the change in distance over $t = \Theta(\log(N_S))$ steps and use a non-Markovian coupling, allowing us to delay the coupling decisions based on future events. We will use the following delayed path coupling theorem due to Czumaj, Kanarek, Kutylowski and Lorys [7]. Let $\mathcal{M}$ be an ergodic Markov chain with statespace Ω (not necessarily $\mathsf{Perms}(\mathcal{S})$) and mixing time $\tau_{\mathcal{M}}(\epsilon)$ as defined in Sect. 2.

Theorem 1 (Czumaj et al.). *Let Δ be a metric defined on $\Omega \times \Omega$ which takes values in $\{0, \ldots, D\}$, let $U = \{(X, Y) \in \Omega \times \Omega : \Delta(X, Y) = 1\}$ and let δ be a positive integer. Let $(X_t, Y_t)_{t \in \mathbb{N}}$ be a coupling for $\mathcal{M}$, such that for every $(X_{t\delta}, Y_{t\delta}) \in U$ it holds that $\mathbf{E}\left[\, \Delta(X_{(t+1)\delta}, Y_{(t+1)\delta}) \,\right] \leq \beta$ for some real $\beta < 1$. Then,*

$$\tau_{\mathcal{M}}(\epsilon) \leq \delta \cdot \left\lceil \frac{\ln(D * \epsilon^{-1})}{\ln \beta^{-1}} \right\rceil .$$

Czumaj and Kutylowski's use the distance metric Δ defined above (the minimum number of transpositions) and define a coupling $(X_t, Y_t)_{t=0}^{T}$ where $\Delta(X_0, Y_0) = 1$ (i.e., X_0 and Y_0 differ by a single transposition). They show that using their coupling, $\mathbf{E}\left[\, \Delta(X_T, Y_T) \,\right] \leq 1/N$, for $T = \Theta(\log N)$ which is sufficient to show the mixing time is $O(\log(N))$. We will use the same coupling to analyze the RCW algorithm and provide a brief overview here for completeness. Full details can be found in their paper [9]. Note that for ease of explanation, the matchings described here are the matchings actually applied at each step (i.e., the B_i's are already incorporated into the description of the matchings). Let $M_1, M_2, \ldots M_T$ be the matchings defined by the coupling for the process X and $N_1, N_2, \ldots N_T$ be the matchings for Y, so that applying these matchings at each step results in the coupling $(X_t, Y_t)_{t=0}^{T}$. We begin by choosing the permutations and corresponding matchings for X at each step, $M_1, M_2, \ldots M_T$, according to the distribution given by the RCW algorithm thus ensuring that the marginals of X agree with the RCW algorithm. Next using the matchings chosen for X we will carefully select the matchings for Y, $N_1, N_2, \ldots N_T$ to ensure that by the end of T steps the two processes will have coupled with probability $1 - 1/N_S$. Without loss

of generality, assume that X_0 and Y_0 differ only by a transposition of points x and y (recall that $\Delta(X_0, Y_0) = 1$). If the matching M_1 contains the pair (or edge) (x, y) then if we apply the same matching minus this pair to Y_0 then after one step, the process has coupled (e.g. $\Delta(X_1, Y_1) = 0$). However the probability that a matching contains this pair is only $\Theta(1/N_S)$ and thus not sufficient to obtain the bound we desire. In order to overcome this Czumaj and Kutylowski observe that if (x, w) and (y, z) are pairs in the matching M_1 then if we let $N_1 = M_1 - (x, w) - (y, z) + (x, z) + (y, w)$ then X_1 and Y_1 differ by a (x, y) transposition. Conversely if we let $N_1 = M_1$ then X_1 and Y_1 differ by a (w, z) transposition. Given this, if M_2 contains either (x, y) or (w, z) then we can choose N_1, N_2 so that $N_3 = M_3$ and the process has coupled. As Czumaj and Kutylowski do, we will call (x, y) and (w, z) good pairs and let GP_t denote the set of good pairs at step t. The general idea behind the argument is to show that at every step the number of good pairs increases by a constant factor and thus after $\Theta(\log N_S)$ steps the number of good pairs is $\Omega(N_S)$. Given this, with high probability in another $\Theta(\log N_S)$ steps one of the matchings M_t will contain a good pair and thus we can define corresponding matchings for Y so that the process couples. We formally define a good pair as follows.

Definition 1 (Czumaj, Kutylowski). *Without loss of generality, assume X_0 and Y_0 differ by a (x, y) transposition and let $GP_0 = \{(x, y)\}$. For each $(x, y) \in GP_{t-1}$:*

1. *If neither x or y is part of the matching M_t then $(x, y) \in GP_t$.*
2. *If $(x, w) \in M_t$ and y is not part of M_t then $(w, y) \in GP_t$.*
3. *If $(y, w) \in M_t$ and x is not part of M_t then $(w, x) \in GP_t$.*
4. *If (x, w), $(y, z) \in M_t$ then if neither w or z are part of pairs in GP_t then $(w, z) \in GP_t$ and $(x, y) \in GP_t$. Otherwise $(w, z) \in GP_t$.*

Using this strategy, Czumaj and Kutylowski formally give a coupling so that if a pair (x, y) is a good pair at time t and M_t contains (x, y) then $X_T = Y_T$. We use this coupling exactly and rely on their proof to show that it is indeed a valid coupling and the marginal distributions of X and Y agree with those given by RCW. Given this coupling, it remains to show that after a time t_1 the number of good pairs is large enough so that in the next t_2 steps one of the t_2 matchings will contain a good pair. We deviate from Czumaj and Kutylowski's approach in this analysis.

We begin by showing that after $t_1 = \Theta(\log N_S)$ steps the probability that there are less than $N_S/9$ good pairs is at most $.5N_S^{-2}$. Next, we show that after an additional $t_2 = \Theta(\log N_S)$ steps the probability that none of the matchings during those additional t_2 steps includes a good pair is at most $.5N_S^{-2}$. Combining these shows that using the given coupling, after $t_1 + t_2$ steps, with probability at most N_S^{-2}, the two processes remain at distance 1 and otherwise they are at distance 0. Thus, $\mathbf{E}\left[\Delta(X_{(t+1)\delta}, Y_{(t+1)\delta})\right] \leq N_S^{-2}$ for $\delta = t_1 + t_2$. Given this we can now apply the delayed path coupling theorem. Since Δ is the minimum number of transpositions to move from one configuration to another, D (the maximum distance between two configurations) is at most N_S. This is due to

the fact that by using a single transposition per point we can put each point in it's new location. Combining these and the delayed path coupling theorem gives the following bound on the mixing time.

Theorem 2. *For* $T \geq \max\left(40\ln(2N_S^2), \frac{10\ln(N_S/9)}{\ln(1+.3(c-1)^4/c^6)}\right) + \frac{36c^3\ln(2N_S^2)}{(c-1)^2}$ *and* $N_S \geq 2^{10}$*, the mixing time* τ *of the* RCW *algorithm satisfies*

$$\tau(\epsilon) \leq T \cdot \left\lceil \frac{\ln(N_S/\epsilon)}{\ln N_S^2} \right\rceil .$$

When $\epsilon = 1/N_S$ *the bound simplifies to* $\tau(1/N_S) \leq T = \Theta(\ln(N_S))$

A straightforward manipulation of the bound on the mixing time gives us the following bound on the variation distance that will be useful in the remainder of the paper. Notice again that as long as the number of rounds of the RCW algorithm is at least $T = \Theta(\ln(N_S))$, the variation distance is less than $1/N_S$.

Corollary 1. *Let* $T = \max\left(40\ln(2N_S^2), \frac{10\ln(N_S/9)}{\ln(1+.3(c-1)^4/c^6)}\right) + \frac{36c^3\ln(2N_S^2)}{(c-1)^2}$ *and* $N_S \geq 2^{10}$*, then*

$$||\nu_{rcw^r} - \mu_s|| \leq N_S^{1-2r/T},$$

where ν_{rcw^r} *is the distribution after* r *rounds of the* RCW *algorithm and* μ_s *is the uniform distribution on permutations of the elements in* $\mathcal{S}$*.*

Our theorem does not explicitly condition on $E[\kappa] = \Theta(N_S)$ as in Czumaj and Kutylowski [9]. Instead our theorem applies only to the RCW algorithm and relies on more specific statements about the chain. For example, a key step in our analysis is to show that at each step of the RCW algorithm a particular point is part of a 2-cycle with constant probability (which implies that $E[\kappa] = \Theta(N_S)$). Let c_x be the probability that point x is part of a 2-cycle. We prove the following claim.

Claim 1.

$$c_x = \frac{(N_S - 1)\cdot(N_T - N_S)^2}{N_T \cdot (N_T - 1)\cdot(N_T - 2)} \geq \frac{(c-1)^2}{c^3} .$$

where the probability is over the choice of π *and* $c = N_T/N_S$*.*

The point x is part of a potential 2-cycle when the algorithm RCW is applied to x and u and v are set; this happens in either the "if" of "else if" blocks of the RCW algorithm given in Sect. 3. The bit b then determines whether or not x and the point it gets paired with actually become part of a 2-cycle. To prove the claim, we need to consider how u and v can be set in the algorithm. There are two cases, corresponding to the "if" and "else if" blocks of the algorithm. First consider the "if" case. We need to determine the probability that $\pi(x) \in \mathcal{S} \wedge \pi^{-1}(x) \notin \mathcal{S} \wedge \pi(\pi(x)) \notin \mathcal{S}$ and $\mathsf{B}(x) = 1$ for a randomly chosen permutation π on $\mathcal{T}$ and B from $\mathcal{S}$ to $\{0,1\}$. There are $N_S - 1$ choices for $\pi(x)$ (the minus one is since we don't want x mapped to itself), $N_T - N_S$ choices for $\pi^{-1}(x)$,

and $N_T - N_S$ choices for $\pi(\pi(x))$. This fixes three mappings, so there are then $(N_T - 3)!$ choices for how to map the remaining points. Thus, the probability we end up in the "if" case is

$$\frac{.5(N_S - 1) \cdot (N_T - N_S)^2 (N_T - 3)!}{N_T!} = \frac{(N_S - 1) \cdot (N_T - N_S)^2}{N_T \cdot (N_T - 1) \cdot (N_T - 2)}.$$

The argument for the "else if" case is almost identical, and gives the same probability. We lower bound the probability as follows.

$$\mathbf{P}\left(X_i = 1\right) = \frac{(N_S - 1) \cdot (N_T - N_S) \cdot (N_T - N_S)}{N_T \cdot (N_T - 1) \cdot (N_T - 2)} \geq \frac{(c-1)^2}{c^3},$$

where $N_T = cN_S$. Note that using linearity of expectations over all points in N_S, this claim implies that the expected number of 2-cycles is $((c-1)^2/c^3)N_S$.

Next, we prove the following lemma which shows that after t_1 steps there are linear number of good pairs.

Lemma 2. *Let $|GP_t|$ be the number of good pairs at step t, $N_S \geq 2^{10}$ and $t_1 = \max(40\ln(2N_S^2), 10\ln(N_S/9)/\ln(1 + .3(c-1)^4/c^6)$ then*

$$\mathbf{P}\left(|GP_{t_1}| < N_S/9\right) \leq .5N_S^{-2}.$$

Proof. We start with one good pair at $t = 1$ and then at each step of the algorithm we say that a good pair (x, y) *splits* if it creates a second good pair (this corresponds to the last case of Definition 1). We begin with bounding the probability that a good pair splits in the RCW algorithm. First, we assume that there are less than $N_S/9$ good pairs (if there are more than we're done). Since we have assumed that there are less than $N_S/9$ good pairs, there are at most $2N_S/9$ points in good pairs and at least $N_S - 2N_S/9 = (7/9)N_S$ points not in good pairs. Good pair (x, y) splits when x and y are both matched to points that are not already in good pairs of which there are at least $(7/9)N_S$. Using this we can now extend the proof of Claim 1 to show the following where c_p is the probability that a particular good pair splits:

$$c_p \geq \frac{(\frac{7}{9})^2 (c-1)^4 (1 - 2^{-8})}{c^6}.$$

Let (x, y) be a good pair. We want to lower bound that probability that point x and y are both part of potential 2-cycles (x, w) and (y, z) where w and z are not in good pairs. Since we are now interested in two points being part of potential 2-cycles, there are 4 different cases; the first case corresponds to u and v begin set for both x and y in the "if" block of the RCW algorithm. We need to determine the probability that $\pi(x) \in (\mathcal{S} - GP) \wedge \pi^{-1}(x) \notin \mathcal{S} \wedge \pi(\pi(x)) \notin \mathcal{S}$ and that $\pi(y) \in (\mathcal{S} - GP) \wedge \pi^{-1}(y) \notin \mathcal{S} \wedge \pi(\pi(y)) \notin \mathcal{S}$ and $\mathsf{B}(x) = \mathsf{B}(y) = 1$ for a randomly chosen permutation π on $\mathcal{T}$ and B from $\mathcal{S}$ to $\{0, 1\}$. Since there are at least $(7/9)N_S$ points not in good pairs, there are $(7/9)N_S$ choices for $\pi(x)$, $N_T - N_S$ choices for $\pi^{-1}(x)$, and $N_T - N_S$ choices for $\pi(\pi(x))$. Given these

mappings, there are $(7/9)N_S - 1$ choices for $\pi(y)$ (the minus one accounts for $\pi(x)$ which is already mapped to a point in $\mathcal{S} - GP$), $N_T - N_S - 1$ choices for $\pi^{-1}(y)$, and $N_T - N_S - 1$ choices for $\pi(\pi(y))$. This fixes six mappings, so there are then $(N_T - 6)!$ choices for how to map the remaining points. Thus, the probability x and y are both mapped to points in $\mathcal{S}$ that are not in good pairs in the "if" block of the algorithm is

$$.25\frac{(7/9)N_S \cdot (N_T - N_S)^2 \cdot ((7/9)N_S - 1) \cdot (N_T - N_S - 1)^2}{N_T \cdot (N_T - 1) \cdot (N_T - 2) \cdot (N_T - 3) \cdot (N_T - 4) \cdot (N_T - 5)}.$$

As in Claim 1, the argument for the other three cases is almost identical, and gives the same probability. We lower bound the probability as follows.

$$c_p \geq \frac{(\frac{7}{9}N_S)(\frac{7}{9}N_S - 1)(N_T - N_S)^2(N_T - N_S - 1)^2}{N_T(N_T - 1)(N_T - 2)(N_T - 3)(N_T - 4)(N_T - 5)} \geq \frac{(\frac{7}{9})^2(c-1)^4(1 - 2^{-8})}{c^6},$$

where $N_T = cN_S$ and $N_S \geq 2^{10}$. Note that the restriction $N_S \geq 2^{10}$ could easily be loosened at the expense of a small constant factor in the bound.

By linearity of expectations, if we have $|GP_t|$ good pairs at step t, then the expected number of good pairs at step $t+1$ is $\mathbf{E}[\,|GP_{t+1}|\,] = |GP_t| + c_p|GP_t|$. Let $G_t = (|GP_{t+1}| - |GP_t|)/|GP_t|$ be the fraction of good pairs that split between time t and $t+1$ (the growth rate). Thus, we have that $\mathbf{E}[\,G_t\,] = c_p$. Next, define an indicator random variable Z_t that is 1 if $G_t \geq \mathbf{E}[\,G_t\,]/2 = c_p/2$ and 0 otherwise. Thus if $\sum_{t=0}^{t_1} Z_t \geq \frac{\ln n/9}{\ln(1+c_p/2)}$ then $|GP_{t_1}|$ is at least $(1 + c_p/2)^{(\ln N_S/9)/\ln(1+c_p/2)} = N_S/9$. This is due to the fact that each times Z_t is one $|GP_t|$ increases at least by a factor of $1 + c_p/2$.

Next, we will show that for $t_1 = \max(40\ln(2N_S^2), 10\frac{\ln N_S/9}{\ln(1+c_p/2)})$,

$$\mathbf{P}\left(\sum_{t=0}^{t_1} Z_t < \frac{\ln N_S/9}{\ln(1 + c_p/2)}\right) < .5N_S^{-2}$$

which implies $\mathbf{P}\,(|GP_{t_1}| < N_S/9) \leq .5N_S^{-2}$. First, using Markov's inequality we will show that $\mathbf{P}\,(Z_t = 0) = \mathbf{P}\,(G_t \leq \mathbf{E}[\,G_t\,]/2) \leq 4/5$. Let $A = 3\mathbf{E}[\,G_t\,] - G_t$, then $\mathbf{P}\,(G_t \leq \mathbf{E}[\,G_t\,]/2) = \mathbf{P}\,(A \geq (3 - 1/2)\mathbf{E}[\,G_t\,] = 2.5\mathbf{E}[\,G_t\,])$. By linearity of expectations, $\mathbf{E}[\,A\,] = \mathbf{E}[\,3\mathbf{E}[\,G_t\,] - G_t\,] = 2\mathbf{E}[\,G_t\,]$. Thus $\mathbf{P}\,(Z_t = 0) = \mathbf{P}\,(A \geq 2.5\mathbf{E}[\,G_t\,]) \leq \mathbf{E}[\,A\,]/2.5\mathbf{E}[\,G_t\,] = 4/5$. Next, we note that the Z_i's are not independent since the probability Z_i is 1 is determined by the number of good pairs. However, since we are assuming there are always at most $N_S/9$ good pairs, this process is stochastically lower bounded by a process with independent variables $X_1, \ldots X_{t_1}$ where each variable X_i is 1 with probability $1/5$ and 0 with probability $4/5$. In the actual process, especially toward the beginning the Z_i's are much more likely to be 1 because there are substantially fewer than $N_S/9$ good pairs. However throughout the process the probability is always at least $1/5$. Next, we will apply the Chernoff bound $\mathbf{P}\,(X < \mathbf{E}[\,X\,]/2) < \exp(-\mathbf{E}[\,X\,]/8)$

with $X = \sum_{t=0}^{t_1} X_t$ and $t_1 = \max(40\ln(2N_S^2), 10(\ln N_S/9)/\ln(1+c_p/2))$. Our choice of t_1 implies that $\mathbf{E}[X] \geq (1/5)40\ln(2N_S^2) = 8\ln(2N_S^2)$. Therefore,

$$\mathbf{P}(X < \mathbf{E}[X]/2) < \exp(-\mathbf{E}[X]/8) <= \exp(-8\ln(2N_S^2)/8) = .5N_S^{-2}.$$

Again due to our choice of t_1, $\mathbf{E}[X] \geq (1/5)10\frac{\ln N_S/9}{\ln(1+c_p/2)} = 2\frac{\ln N_S/9}{\ln(1+c_p/2)}$. Combining these gives the desired result,

$$\mathbf{P}\left(X < \frac{\ln N_S/9}{\ln(1+c_p/2)}\right) < \mathbf{P}(X < \mathbf{E}[X]/2) < .5N_S^{-2}. \qquad \square$$

Finally, we consider the matchings during the next t_2 steps and show the probability that none of them includes a good pair is at most $.5N_S^{-2}$. We say that a pair (x, y) is part of a *potential* matching if the RCW algorithm maps x to y regardless of the value of $B(x)$. Specifically, we prove the following lemma.

Lemma 3. *Let $t_2 = 36c^3\ln(2N_S^2)/(c-1)^2$ then conditioned on $|GP_{t_1}| \geq n/9$, the probability that the next t_2 potential matchings contain no edges from GP_{t_1} is at most $.5N_S^{-2}$.*

Proof. First, consider a good pair (x, y). We claim that the probability that x is mapped to y in one step of the RCW algorithm is $2\cdot(c-1)^2/(c^3N_S)$. Again, there are two cases corresponding to the "if" and "else if" blocks of the algorithm. Consider the "if" case, we need to determine the probability that $\pi(x) = y \wedge \pi^{-1}(x) \notin \mathcal{S} \wedge \pi(\pi(x)) \notin \mathcal{S}$. There is one choice for $\pi(x)$, $N_T - N_S$ choices for $\pi^{-1}(x)$, and $N_T - N_S$ choices for $\pi(\pi(x))$. This fixes three mappings, resulting in $(N_T-3)!$ choices for the remaining points. Thus, the probability x is mapped to y in the "if" case is

$$\frac{(N_T-N_S)^2\cdot(N_T-3)!}{N_T!} = \frac{(N_T-N_S)^2}{N_T\cdot(N_T-1)\cdot(N_T-2)} \leq \frac{(c-1)^2}{c^3N_S}.$$

Again, the argument for the "else if" case is almost identical giving a factor of two in the probability that x is mapped to y.

Let H_t be the number of edges in the potential matching at time t that correspond to good pairs. There are at least $N_S/9$ good pairs at time t_1 and each is in the potential matching with probability at least $2(c-1)^2/(c^3N_S)$. Thus, by linearity of expectations, for $t > t_1$ we have that

$$\mathbf{E}[H_t] \geq (N_S/9)(2(c-1)^2/(c^3N_S)) = 2(c-1)^2/(9c^3).$$

Since the potential matchings generated at each time step are independent we can now use a Chernoff bound to show that $\mathbf{P}\left(\sum_{t=t_1}^{t_1+t_2} H_t < 1\right) < .5N_S^{-2}$. Again, we will use the following form of the Chernoff bound; $\mathbf{P}(X < \mathbf{E}[X]/2) < \exp(-\mathbf{E}[X]/8)$. If we let $X = \sum_{t=t_1}^{t_1+t_2} H_t$ where $t_2 = 36c^3\ln(2N_S^2)/(c-1)^2$

then by linearity of expectations $\mathbf{E}[X] = \sum_{t=t_1}^{t_1+t_2} \mathbf{E}[H_t] \geq t_2 \cdot 2(c-1)^2/(9c^3) = 8\ln(2N_S^2)$. Applying the above Chernoff bound gives the following,

$$\mathbf{P}\left(\sum_{t=t_1}^{t_1+t_2} H_t < 4\ln(2N_S^2)\right) < \exp(-8\ln(2N_S^2)/8) = .5N_S^{-2}$$

Thus, since $\mathbf{P}\left(\sum_{t=t_1}^{t_1+t_2} H_t < 1\right) < \mathbf{P}\left(\sum_{t=t_1}^{t_1+t_2} H_t < 4\ln(2N_S^2)\right)$, we have that $\mathbf{P}\left(\sum_{t=t_1}^{t_1+t_2} H_t < 1\right) < .5N_S^{-2}$, as desired. □

5 CCA Security

Let $\mathcal{S} \subseteq \mathcal{T}$ with N_S and N_T their sizes, respectively. Let r be a positive integer called the repetition number. Let $E : \mathcal{K}_E \times \mathcal{T} \to \mathcal{T}$ be a block cipher with domain $\mathcal{T}$. Let $F : \mathcal{K}_F \times \{1, \ldots, r\} \times \mathcal{S} \to \{0,1\}$ be a pseudorandom function family.

We use reverse 2-cycle walking to define a new block cipher $\tilde{E} : \mathcal{K} \times \mathcal{S} \to \mathcal{S}$ as follows. The key space $\mathcal{K}$ is $\mathcal{K}_E^r \times \mathcal{K}_F$. Let $F_{i,K}(\cdot) = F(K, i, \cdot)$. Then $\tilde{E}$, on input key K and point x, parses its key K as r block cipher keys $K_1, \ldots, K_r$ and a PRF key K' and then computes

$$\mathsf{RCW}^r_{(E_{K_1},\ldots,E_{K_r}),(F_{1,K'},\ldots,F_{r,K'})}(x) \ .$$

The following theorem establishes the CCA security of block cipher $\tilde{E}$.

Theorem 3. *Let E, F, and $\tilde{E}$ be defined as above. Let A be an adversary attacking $\tilde{E}$ and making q queries. Then,*

$$\mathbf{Adv}^{\mathrm{cca}}_{\tilde{E}}(A) \leq r \cdot \mathbf{Adv}^{\mathrm{cca}}_{E}(B) + \mathbf{Adv}^{\mathrm{prf}}_{F}(C) + \Gamma \ ,$$

with B making $3 \cdot q$ queries, C making $r \cdot q$ queries, and Γ being the bound on variation distance from Corollary 1 that depends on r.

Proof. We wish to bound the cca-advantage of an adversary A attacking $\tilde{E}$ and making at most q oracle queries. Thus we wish to bound

$$\mathbf{Adv}^{\mathrm{cca}}_{\tilde{E}}(A) = \mathbf{P}\left(A^{\pm\tilde{E}(K,\cdot)} \Rightarrow 1\right) - \mathbf{P}\left(A^{\pm\pi(\cdot)} \Rightarrow 1\right) \ .$$

We will start with the left term above, where A is given access to oracles for $\tilde{E}$ and $\tilde{E}^{-1}$, and gradually change the oracles until they are simply random permutations on $\mathcal{S}$, bounding each oracle change accordingly.

Recall that $\pm\tilde{E}(K, \cdot)$ is really just a more compact way of writing

$$\pm\mathcal{O}_1(\cdot) = \pm\mathsf{RCW}^r_{(E_{K_1},\ldots,E_{K_r}),(F_{1,K'},\ldots,F_{r,K'})}(\cdot).$$

Our first oracle transition, from $\mathcal{O}_1$ to $\mathcal{O}_2$, replaces all of the block ciphers with random permutations on the same domain $\mathcal{T}$, turning the oracle into

$$\pm\mathcal{O}_2(\cdot) = \pm\mathsf{RCW}^r_{(\pi_1,\ldots,\pi_r),(F_{1,K'},\ldots,F_{r,K'})}(\cdot) \ ,$$

where each π_i is a random permutation on $\mathcal{T}$. We can bound the difference using a hybrid argument and an adversary B attacking the cca security of E.

The adversary B is given an encryption algorithm and its inverse, which we denote by $\mathcal{O}_B$ and $\mathcal{O}_B^{-1}$. Adversary B first chooses a random index $i \in \{1, \ldots, r\}$. Next, B chooses $i-1$ keys $K_1, \ldots, K_{i-1}$ for block cipher E, and a PRF key K'. It then runs adversary A, simulating A's oracle queries as follows.

On encryption query x from A, B first computes the mapping

$$x' = \mathsf{RCW}^{i-1}_{(E_{K_1},\ldots,E_{K_{i-1}}),(F_{1,K'},\ldots,F_{r,K'})}(x).$$

In words, B applies $i-1$ rounds of the RCW algorithm with the keys B chose earlier; let the result be x'. B then uses its oracles to determine how x' should be mapped in the ith step of RCW. Looking at the RCW algorithm in Sect. 3, we see that to determine this B will need to query $\mathcal{O}(x')$, $\mathcal{O}^{-1}(x')$, and one of $\mathcal{O}(\mathcal{O}(x'))$ and $\mathcal{O}^{-1}(\mathcal{O}^{-1}(x'))$ depending on the results of the first two queries. Thus, for each encryption query A makes, B queries its own oracles three times, making either two forward and one inverse or one forward and two inverse queries. After B determines how x' should be mapped at the ith step (call the result x''), it computes how x'' should be mapped by steps $i+1$ through r using the RCW algorithm with random permutations. To simulate these permutations, B simply uses tables.

B handles inverse queries from A similarly, except it computes RCW^r in the reverse direction, using the same tables for random permutations in steps $i+1$ through r, using its own oracles at step i, and using the keys it chose for steps 1 through $i-1$.

From the description of B, we can see that if B is given as oracles a real block cipher and its inverse under some key, then B simulates for A the oracles $\pm\mathsf{RCW}^r_{(E_{K_1},\ldots,E_{K_{i-1}},E_{K_i},\pi_{i+1},\ldots,\pi_r)}$, while if B is given as oracles a random permutation and its inverse, it simulates for A oracles $\pm\mathsf{RCW}^r_{(E_{K_1},\ldots,E_{K_{i-1}},\pi_i,\ldots,\pi_r)}$.

Thus, it follows that

$$\mathbf{P}\left(A^{\pm\mathcal{O}_1(\cdot)} \Rightarrow 1\right) - \mathbf{P}\left(A^{\pm\mathcal{O}_2(\cdot)} \Rightarrow 1\right) \leq r \cdot \mathbf{Adv}^{\mathrm{cca}}_E(B) \tag{1}$$

where B makes at most $3q$ oracle queries.

For our next oracle transition, from $\mathcal{O}_2$ to $\mathcal{O}_3$, we replace the PRF $F : \mathcal{K}_F \times \{1, \ldots, r\} \times \mathcal{S} \to \{0,1\}$ with a truly random function $\rho : \{1, \ldots, r\} \times \mathcal{S} \to \{0,1\}$. Similar to how we defined $F_{i,K}(\cdot) = F(K, i, \cdot)$, we let $\rho_i(\cdot) = \rho(i, \cdot)$. Thus, our oracle $\mathcal{O}_3$ becomes

$$\pm\mathcal{O}_3(\cdot) = \pm\mathsf{RCW}^r_{(\pi_1,\ldots,\pi_r),(\rho_1,\ldots,\rho_r)}(\cdot).$$

We can bound the change in advantage by the prf-advantage of an adversary C. The adversary C, given access to an oracle that is either the real pseudorandom function or a truly random function, runs A and simulates its oracles by computing RCW^r using tables and random sampling to simulate the random permutations used by RCW, and using its own oracle to compute the bit b used

in each round. Since there are r rounds of RCW and A makes q queries, C will make rq queries to its own oracle. Clearly, if C's oracle is a real pseudorandom function, it simulates $\mathcal{O}_2$ for A, while if C's oracle is a truly random function it simulates $\mathcal{O}_3$ for A. Thus,

$$\mathbf{P}\left(A^{\pm\mathcal{O}_2(\cdot)} \Rightarrow 1\right) - \mathbf{P}\left(A^{\pm\mathcal{O}_3(\cdot)} \Rightarrow 1\right) \leq \mathbf{Adv}_F^{\mathrm{prf}}(C) \tag{2}$$

where C makes at most $r \cdot q$ oracle queries.

At this point, we have r rounds of RCW using only ideal components. For our last oracle transition, $\mathcal{O}_3$ to $\mathcal{O}_4$, we replace RCW entirely with a random permutation on $\mathcal{S}$. Thus,

$$\pm\mathcal{O}_4(\cdot) = \pm\pi(\cdot).$$

We are now in the information theoretic setting, and the maximum advantage of any adversary in distinguishing between RCW^r with ideal components and a random permutation on $\mathcal{S}$ is bounded in Corollary 1 in the previous section. Thus,

$$\mathbf{P}\left(A^{\pm\mathcal{O}_3(\cdot)} \Rightarrow 1\right) - \mathbf{P}\left(A^{\pm\mathcal{O}_4(\cdot)} \Rightarrow 1\right) \leq \Gamma \tag{3}$$

where Γ is the value the variation distance is bounded by in the Corollary.

We can thus bound the cca-advantage of A as follows:

$$\begin{aligned}\mathbf{Adv}_{\tilde{E}}^{\mathrm{cca}}(A) \leq &\left(\mathbf{P}\left(A^{\pm\mathcal{O}_1(\cdot)} \Rightarrow 1\right) - \mathbf{P}\left(A^{\pm\mathcal{O}_2(\cdot)} \Rightarrow 1\right)\right)\\ &+\left(\mathbf{P}\left(A^{\pm\mathcal{O}_2(\cdot)} \Rightarrow 1\right) - \mathbf{P}\left(A^{\pm\mathcal{O}_3(\cdot)} \Rightarrow 1\right)\right)\\ &+\left(\mathbf{P}\left(A^{\pm\mathcal{O}_3(\cdot)} \Rightarrow 1\right) - \mathbf{P}\left(A^{\pm\mathcal{O}_4(\cdot)} \Rightarrow 1\right)\right)\end{aligned}$$

where recall that

$$\begin{aligned}\mathcal{O}_1 &= \mathsf{RCW}^r_{(E_{K_1},\ldots,E_{K_r}),(F_{1,K'},\ldots,F_{r,K'})}\\ \mathcal{O}_2 &= \mathsf{RCW}^r_{(\pi_1,\ldots,\pi_r),(F_{1,K'},\ldots,F_{r,K'})}\\ \mathcal{O}_3 &= \mathsf{RCW}^r_{(\pi_1,\ldots,\pi_r),(\rho_1,\ldots,\rho_r)}\\ \mathcal{O}_4 &= \pi\end{aligned}$$

Substituting in Eqs. (1), (2), and (3) gives the bound from the theorem statement. □

6 Full Security via Reverse Cycle Walking

Let $\mathcal{S} = \{0,\ldots,N-1\}$ and $\mathcal{T} = \{0,\ldots,2N-1\}$. Thus, $N_T = |\mathcal{T}| = 2N$ and $N_S = |\mathcal{S}| = N$. Let r be a positive integer called the repetition number.

Let $E : \mathcal{K} \times \mathcal{T} \to \mathcal{T}$ be a block cipher on $\mathcal{T}$ and let E^{-1} be its inverse. Let $F : \mathcal{K}_F \times \{1, \ldots, r\} \times \mathcal{S} \to \mathcal{S}$ be a pseudorandom function family. Let $\tilde{E}$ be defined as it was in the previous section, using r rounds of RCW with E and F. The following theorem states that if E is secure against cca adversaries making $N = (1/2)N_T$ queries, then $\tilde{E}$ is fully secure (i.e., secure against adversaries making $N = N_S$ queries). In other words, the RCW construction allows us to build a fully secure cipher out of a partially secure cipher on a larger set.

Theorem 4. *Let $\mathcal{S}$, $\mathcal{T}$, E, F, and $\tilde{E}$ be defined as above. Let A be a cca adversary attacking $\tilde{E}$ and making $N = N_S$ queries. Then*

$$\mathbf{Adv}^{\mathrm{cca}}_{\tilde{E}}(A) \leq r \cdot \mathbf{Adv}^{\mathrm{cca}}_{E}(B) + \mathbf{Adv}^{\mathrm{prf}}_{F}(C) + \Gamma$$

where B makes $N = (1/2)N_T$ queries to its encryption oracle, C makes $N \cdot r$ oracle queries, and Γ is the bound on variation distance from Corollary 1 that depends on r.

Proof. The proof is identical to the proof of Theorem 3 except for how adversary B answers oracle queries from A in the hybrid argument. As in the proof of Theorem 3, we let

$$\pm\mathcal{O}_1(\cdot) = \pm\mathsf{RCW}^r_{(E_{K_1},\ldots,E_{K_r}),(F_{1,K'},\ldots,F_{r,K'})}(\cdot),$$

and

$$\pm\mathcal{O}_2(\cdot) = \pm\mathsf{RCW}^r_{(\pi_1,\ldots,\pi_r),(F_{1,K'},\ldots,F_{r,K'})}(\cdot).$$

We then use adversary B attacking E to argue that replacing A's $\pm\mathcal{O}_1$ oracles with $\pm\mathcal{O}_2$ has little effect.

Let B's oracles be denoted by $\mathcal{O}_B$ and $\mathcal{O}_B^{-1}$. As in the previous proof, B begins by choosing a random index $i \in \{1, \ldots, r\}$, then $i - 1$ keys $K_1, \ldots, K_{i-1}$ for E and a PRF key K' for F.

This is where we encounter the major change from the adversary in the previous proof. At this point, adversary B should query its own oracle $\mathcal{O}_B$ on all points $x \in \{0, \ldots, N-1\}$, recording the answers in a table. Specifically, B sets $\mathtt{T}[x] = \mathcal{O}_B(x)$. B then runs A, answer its oracle queries as follows.

On encryption query x from A, B computes

$$x' = \mathsf{RCW}^{i-1}_{(E_{K_1},\ldots,E_{K_{i-1}}),(F_{1,K'},\ldots,F_{r,K'})}(x).$$

using the block cipher keys and the PRF key it chose earlier. To determine how x' should be mapped with the ith step of RCW, B now has to use the answers it received from its oracle and stored in table T. Notice that B can evaluate the boolean conditions in the "if" case of the RCW algorithm as follows: if $\mathtt{T}[x] \in \mathcal{S}$ and there is no $z \in \mathcal{S}$ s.t. $\mathtt{T}[z] = x$ and $\mathtt{T}[\mathtt{T}[x]] \notin \mathcal{S}$. Similarly, B can evaluate the "else if" as follows: if $\mathtt{T}[x] \notin \mathcal{S}$ and there does exist $z \in \mathcal{S}$ s.t. $\mathtt{T}[z] = x$ and there does not exist $w \in \mathcal{S}$ s.t. $\mathtt{T}[w] = z$. In other words, the table T contains enough information to evaluate the RCW algorithm on any point in $\mathcal{S}$.

After B computes how x' is mapped in the ith RCW round, it uses tables to simulate random permutations for rounds $i+1$ through r, just as it did in the proof of Theorem 3. Inverse queries from A are handled similarly to in that proof, just with the ith round of RCW computed as above with table T.

The rest of the proof (i.e., bounding the change from using F to using a truly random function) follows the exact steps as the proof in the previous section. □

7 Open Questions

There are a number of interesting open questions surrounding reverse cycle walking. We analyzed the security of reverse 2-cycle walking, but we explained in the introduction that the algorithm can be generalized to longer cycles. An interesting question is whether reverse t-cycle walking, for $t > 2$, leads to better bounds than we were able to prove here. Another interesting question is what is the optimal worst-case running time for strong pseudorandom permutations on general sets where only efficient membership testing is assumed. We were able to show a worst case running time of $\Theta(t(N) \log N)$, where $t(N)$ is the time to encipher a point in the larger set $\mathcal{T}$. We conjecture that this is in fact optimal.

Acknowledgements. We thank Tom Ristenpart for his very helpful comments on an earlier draft of this paper. We also thank the anonymous Asiacrypt reviewers for their detailed feedback.

References

1. Bellare, M., Ristenpart, T., Rogaway, P., Stegers, T.: Format-preserving encryption. In: Jacobson, M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 295–312. Springer, Heidelberg (2009). doi:10.1007/978-3-642-05445-7_19
2. Bellare, M., Rogaway, P., Spies, T.: The FFX mode of operation for format-preserving encryption. Submission to NIST, February 2010
3. Black, J., Rogaway, P.: Ciphers with arbitrary finite domains. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 114–130. Springer, Heidelberg (2002). doi:10.1007/3-540-45760-7_9
4. Brightwell, M., Smith, H.: Using datatype-preserving encryption to enhance data warehouse security. In: National Information Systems Security Conference (NISSC) (1997)
5. Bubley, R., Dyer, M.E.: Faster random generation of linear extensions. In: Karloff, H.J. (ed.) 9th SODA, January 1998, pp. 350–354. ACM-SIAM (1998)
6. Czumaj, A.: Random permutations using switching networks. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC, June 2015, pp. 703–712. ACM Press (2015)
7. Czumaj, A., Kanarek, P., Kutylowski, M., Lorys, K.: Delayed path coupling and generating random permutations via distributed stochastic processes. In: Tarjan, R.E., Warnow, T. (eds.) 10th SODA, January 1999, pp. 271–280. ACM-SIAM (1999)
8. Czumaj, A., Kanarek, P., Kutylowski, M., Lorys, K.: Fast generation of random permutations via networks simulation. In: European Symposium on Algorithms, pp. 246–260 (1996)

9. Czumaj, A., Kutylowski, M.: Delayed path coupling and generating random permutations. Random Struct. Algorithms **17**, 238–259 (2000)
10. Granboulan, L., Pornin, T.: Perfect block ciphers with small blocks. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 452–465. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74619-5_28
11. Hoang, V.T., Morris, B., Rogaway, P.: An enciphering scheme based on a card shuffle. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 1–13. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_1
12. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996). doi:10.1007/3-540-68697-5_9
13. Levin, D.A., Peres, Y., Wilmer, E.L.: Markov Chains and Mixing Times. American Mathematical Society (2006)
14. Luchaup, D., Dyer, K.P., Jha, S., Ristenpart, T., Shrimpton, T.: LibFTE: a toolkit for constructing practical, format-abiding encryption schemes. In: Proceedings of the 23rd USENIX Security Symposium, pp. 877–891 (2014)
15. Luchaup, D., Shrimpton, T., Ristenpart, T., Jha, S.: Formatted encryption beyond regular languages. In: Ahn, G.J., Yung, M., Li, N. (eds.) ACM CCS 14, November 2014, pp. 1292–1303. ACM Press (2014)
16. Morris, B., Rogaway, P.: Sometimes-recurse shuffle. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 311–326. Springer, Heidelberg (2014). doi:10.1007/978-3-642-55220-5_18
17. Morris, B., Rogaway, P., Stegers, T.: How to encipher messages on a small domain. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 286–302. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03356-8_17
18. Ristenpart, T., Yilek, S.: The mix-and-cut shuffle: small-domain encryption secure against N queries. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 392–409. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40041-4_22
19. Sinclair, A.: Algorithms for Random Generation and Counting. Progress in Theoretical Computer Science. Birkhäuser, Boston (1993)