Lightweight Leakage-Resilient PRNG from TBCs using Superposition

Mustafa Khairallah[1], Srinivasan Yadhunathan[1], and Shivam Bhasin[2]

[1] Seagate Research Group, Singapore, Singapore, firstname.lastname@seagate.com
[2] Nanyang Technological University, Singapore, Singapore, sbhasin@ntu.edu.sg

Abstract. In this paper, we propose a leakage-resilient pseudo-random number generator (PRNG) design that leverages the rekeying techniques of the PSV-Enc encryption scheme and the superposition property of the Superposition-Tweak-Key (STK) framework. The random seed of the PRNG is divided into two parts; one part is used as an ephemeral key that changes every two calls to a tweakable block cipher (TBC), and the other part is used as a static long-term key. Using the superposition property, we show that it is possible to eliminate observable leakage by only masking the static key. Thus, our proposal itself can be seen as a superposition of masking and rekeying. We show that our observations can be used to design an unpredictable-with-leakage PRNG as long as the static key is protected, and the ephemeral key cannot be attacked with 2 traces. Our construction enjoys better theoretical security arguments than PSV-Enc; better Time-Data trade-off and leakage assumptions, using the recently popularized unpredictability with leakage. We verify our proposal by performing Test Vector Leakage Assessment (TVLA) on an STK-based TBC (Deoxys-TBC) operated with a fixed key and a dynamic random tweak. Our results show that while the protection of the static key is non-trivial, it only requires $\approx 10\%$ overhead for first-order protection in the most conservative setting, unlike traditional masking that may require significant overheads of 300% or more.

Keywords: Leakage Resilience · PRNG · TBC · Levelled Implementations · Unpredictability · TVLA · STK · PSV-Enc

1 Introduction

For more than 25 years, side-channel analysis has been at the forefront of cryptology in general, and symmetric key cryptography in particular. While in classical cryptography the adversary is assumed to interact with the cryptographic scheme in a black-box way, respecting a set of pre-imposed rules and limitations, side-channel analysis is concerned with adversaries that can observe the device's behaviour in action (also known as grey-box model) and make inferences based on physical measurements, such as timing, power or electromagnetic measurements. One of the most famous examples of this paradigm is the

Differential Power Analysis (DPA) proposed in [KJJ99]. Protecting implementations against side channel attacks has been a goal for designers for more than 25 years. The most widespread countermeasure against DPA-like attacks is circuit masking [ISW03], where all the secret and sensitive variables are encoded using a secret-sharing scheme, and the circuit/software is adjusted to compute the function over the new representation. This task is far from trivial, as the designers must make sure that the secret is never unveiled in plain during any step of the computation. Several models having been proposed over the years to capture this goal, most notably the probing model [ISW03], which represents the implementation as a circuit and makes sure that for any adversary that can observe d wires during any one execution, cannot deduce any sensitive variables, d is known as the security order of the implementation. It was later shown that the probing model does not capture the full reality, as combining shares of the secret may happen in other ways than unveiling the secret in temporary variables (represented by wires). Different types of *glitches* have been shown to leak the secret even if the scheme is secure in the probing model [NRR06], which led to new security models and many masking schemes attempting to address these issues [CS20]. These countermeasures may increase the cost of computations by several orders of magnitude, due to multiple reasons. The cost of masking non-linear functions grows quadratically with d. Besides, protecting against glitches requires synchronizing temporary variables in sophisticated ways that slow down the operation and may require large amounts of random bits.

For these reasons, a new design paradigm emerged known as leakage-resilient cryptography [DP08,BBC+20]. In this approach, a cryptographic mode of operation is designed such that it may be costly as a black-box design, compared to classical methods, but is easier to protect against side-channel analysis. The main approach towards this goal is the so-called *levelled implementations*, where the algorithm is divided into two parts: one part uses a long-term secret key, is heavily protected against side-channel analysis, and the other part uses only temporary secrets that acts as moving targets for the adversary. This makes DPA significantly harder as it requires many traces with the same secret. If the heavily protected part is protected against DPA, the adversary can only target the other part, and since it does not expose long-term secrets, DPA is not possible. This leaves the adversary with Simple Power Analysis (SPA). SPA refers to attacks that use low trace counts, typically single trace. Besides, SPA is sometimes used as an umbrella-term that includes all attacks that require a small number of traces with the same key, but can have extensive modelling/profiling phases, where the adversary can collect traces from the implementation with their own key. Template attacks with small number of traces fall into this category. These (profiled/non-profiled single trace) attacks are much harder to mount, and are easier to protect against compared to DPA. For software, a cheaper countermeasure such as shuffling can be used [VCMKS12], while in hardware if the implementation includes many parallel functionalities, such attacks maybe unfeasible due to inherent noise (or cost effective noise sources could be easily deployed).

Another related countermeasure is fresh rekeying, initially proposed by Medwed *et al.* [MSGR10a] and improved through various methods (see Section 2). This type of countermeasure assumes the existence of a easier to protect rekeying function that takes the secret key and a public rekeying parameter and uses it to generate temporary keys that are only used a small number of times. This rekeying function needs to be easier to protect against DPA than a classical cryptographic mode targeting the same security goal. It can be a leakage-resilient PRF, such as the GGM scheme [GGM85] or more recently the LR4 scheme [UHIM23]. It has also been shown that it can be a weaker function such as Galois Field Multiplication [MSGR10a]. Mennink [Men20] formalized the requirement using Universal Hash Functions (UHFs), where depending on the rekeying scheme, the rekeying function needs to satisfy a combinatorial security goal.

This work: One of the early symmetric key encryption modes that target levelled implementations is the PSV-Enc scheme proposed in [PSV15]. The scheme requires a heavily protected key-derivation function that takes as input a key and a nonce (or random IV) and generates an initial subkey K_0 that is used in the scheme depicted in Figure 1. The PSV-Enc scheme is based on the 2-PRG construction [SPY13] and in this paper, we will focus on using it as a Pseudo-Random Number Generator (PRNG), where we assume that a randomness source exists that generates a uniformly random initial key K_0, and the scheme in Figure 1 is used as a PRNG. Note that is equivalent to assessing the outcome of a single query of the original PSV-Enc, with query length q_e. Since each key is used only twice (unless a collision on the subkeys occurs), the scheme is leakage resilient as long as the underlying Block Cipher (BC) cannot be attacked with a 2-trace attack. On the other hand, PSV-Enc as a PRNG (the 2-PRG construction) experiences the following limitations:

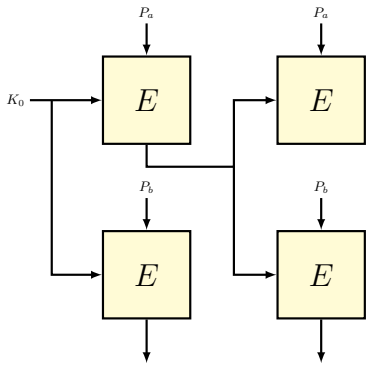

Fig. 1. Two blocks of the PRNG used in PSV-Enc.

- As a PRNG, PSV-Enc can only achieve birthday bound security, even in the black box setting. This is due to the fact that if any two subkeys collide, the

output string will have a short periodic cycle, and can be easily distinguished from a random stream of bits. Besides, the PRNG suffers from a Time-Data trade-off, where the attacker can guess any of the subkeys, and their advantage is close to one when they make q_p offline guesses and observed q_e blocks, such that $q_e q_p \approx 2^n$. Both these limitations maybe addressed using a Tweakable Block Cipher (TBC), rather than a BC, by including a counter as a tweak in the TBC calls.

- A related limitation is that the BC calls in PSV-Enc do not (and should not) share a common secret key, making its analysis in the black-box setting rely on easier ideal cipher model or multi-key analysis of the BC. In terms of leakage resilience, this makes it harder to use the unpredictability with leakage assumption, recently introduced in [BGPS21]. This limitation, unlike the previous one, cannot be addressed using a TBC.
- PSV-Enc is designed such that the subkey size is equal to the block size n. This means that if we want higher security, or larger input to the PRNG, we need to not only use a (T)BC with a larger key size, but also with a larger block size. Another possibility is to use $b+1$ calls per iteration to generate a (bn)-bit subkey. However, this increases the cost by a factor of $(b+1)/2$ and requires a stronger assumption that the BC is secure against $(b+1)$-trace attacks.

This presents us with a set of research goals to design a new leakage-resilient PRNG:

- The key size should be decoupled from the block size.
- It should be possible to use the unpredictability with leakage assumption on the underlying (T)BC to derive the unpredictability with leakage of the PRNG.
- The design should be realizable using a practical TBC with a very small performance penalty compared to if it is used in a PSV-Enc-like design.
- The design (in the black-box setting) should only permit much safer Time-Data trade-offs.

To address these goals, we present the PRNG design in Figure 2. It is similar to PSV-Enc except that initial key consists of two parts: an n-bit part K_0, which is similar to K_0 in PSV-Enc, and a k-bit part K_m, which is used as a static key to an underlying TBC. If we set K_m to a known constant, the two designs are similar. However, by having this static key, we are able to model the TBC as a standard Tweakable Pseudo-Random Permutation (TPRP) in the black-box model and an unpredictable-with-leakage TBC. Besides, we are able to improve the security against Time-Data trade-offs and decouple the size of the input from the block-size of the TBC, allowing using TBCs with smaller block sizes.

On the other hand, when the reader sees Figure 2, alarm bells should be ringing: *how can this design be implemented without using a heavily protected TBC?* The answer is *superposition*. Superposition is a concept introduced in the Superposition-Tweak-Key (STK) framework in [JNP14], where the tweak and

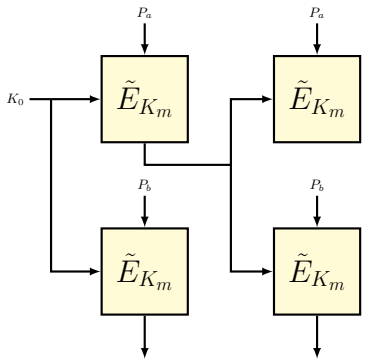

Fig. 2. The proposed PRNG with STK-based TBCs.

the key are seen as a (cn)-bit string called the tweakey for a small constant c (typically 2 or 3), and c linear transformations are applied in parallel on each c-bit section of the tweakey to generate c n-bit sub-tweakeys for each round of the TBC. Then, these c values are bitwise-XORed to generate the round subkey. Consider a case where one n-bit section of the tweakey is uniformly random (with n-bit entropy), then the round keys are also randomly selected with n-bit entropy. Thus, when a static secret key is added to the random tweak, the information from the secret key is perfectly hidden. This means that we do not need heavy protection of the bulk of TBC, specially the costly SBox operation. What remains is to protect the part of the key schedule that operates on the static key against DPA. This is the main goal of the experiments performed in Section 4. Several TBCs follow the STK framework, such as Deoxys-TBC [JNPS21] and Skinny [BJK+16]. In the rest of the paper, we will focus on Deoxys-TBC as a proof-of-concept.

It worth noting that this is not the first design that uses a static key in a function that is not heavily protected and claims leakage resilience. ISAP [DEM+17] uses a similar concept for its rekeying function, which uses a static secret key but relies on the properties of the underlying Sponge function with a very small (1-bit) rate to ensure it is hard to perform DPA. Our proposal can be seen in the same category, where we rely on the superposition property of the underlying TBC to make a similar claim.

2 Preliminaries

Pseudo-Random Number Generators (PRNGs) A PRNG is a function $G : \{0,1\}^\lambda \times \mathbb{N} \to \{0,1\}^{\lambda+l}$ that takes as input a short λ-bit input and a natural number l and returns a $\lambda + l$-bit output. Its security ensures that if the input selected uniformly at random from $\{0,1\}^\lambda$, its output is indistinguishable from a string selected uniformly at random from $\{0,1\}^{\lambda+l}$. In other words, let $\mathbf{A}$ be an adversary that requests a $(\lambda+l)$-bit string and outputs 0 or 1. Let \$ be an oracle that

when queried with any input returns a string sampled uniformly from $\{0,1\}^{\lambda+l}$. Then, we say G is a secure PRNG if for any adversary,

$$\mathbf{Adv}^{\mathtt{prng}}_{G}(\mathbf{A}) \stackrel{\text{def}}{=} |\Pr[r \xleftarrow{\$} \{0,1\}^{\lambda} : \mathbf{A}^{G(r,l)} \Rightarrow 1] - \Pr[\mathbf{A}^{\$(l)} \Rightarrow 1]| \leq \epsilon_{\mathtt{prng}}$$

where $\epsilon_{\mathtt{prng}}$ is negligible.

Tweakable Block Ciphers (TBC) and their Security Models A function $\tilde{E} : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \rightarrow \{0,1\}^n$ is called a TBC is for each pair $(K,T) \in \mathcal{K} \times \mathcal{T}$, $\tilde{E}$ acts a permutation over $\{0,1\}^n$. If $\mathcal{T} = \Phi$ (the empty set), then $\tilde{E}$ is simply a block cipher. The most widely used security notion of TBCs, commonly referred to as the standard model, is the indistinguishability from tweakable random permutations; given a key K selected uniformly at random, then for each $T \in \mathcal{T}$, $\tilde{E}$ is indistinguishable from a random permutation. In this case, we say $\tilde{E}$ behaves as a TPRP. Let $\tilde{\Pi}$ be a family of $|\mathcal{T}|$ permutations selected uniformly at random and indexed by $T \in \mathcal{T}$, then for an adversary $\mathbf{A}$,

$$\mathbf{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathbf{A}) \stackrel{\text{def}}{=} |\Pr[K \xleftarrow{\$} \mathcal{K} : \mathbf{A}^{\tilde{E}(K,\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathbf{A}^{\tilde{\Pi}} \Rightarrow 1]|.$$

and for all adversaries that can make q queries and run in time at most t,

$$\mathbf{Adv}^{\mathtt{tprp}}_{\tilde{E}}(q,t) \stackrel{\text{def}}{=} \max_{\mathbf{A}} \mathbf{Adv}^{\mathtt{tprp}}_{\tilde{E}}(\mathbf{A}).$$

Another security notion of TBCs is the ideal cipher model (ICM). The ICM states that a TBC $\tilde{E}$ behaves as a random permutation for all possible key-tweak pairs.

Rekeying: A rekeying scheme (in our context, inspired by [Men20]) is a scheme that uses a (T)BC and targets the protection of the secret key against side-channel attacks by separating encryption into two functionalities: a subkey generation function that takes as input the master key and a public parameter R to generates a subkey S. A block encryption function takes the subkey S and encrypts the message. The subkey generation function does not need to be cryptographically strong but needs to be protected against side-channel analysis like differential power analysis (DPA). The block encryption needs to provide TPRP security but only needs to be protected against simple power analysis (SPA). This concept was introduced in the context of block ciphers by Medwed *et al.* [MSGR10b] and has since been one of the bedrocks of leakage-resilient symmetric key cryptography [DEMM14,DKM+16,BKP+18,BGP+19,GIK+22]. It has been adopted in some industrial applications [nxp]. A few years ago, Mennink [Men20] showed that in terms of its syntax and security goal, a rekeying scheme is not different from a TBC. What matters is how it is designed and how efficient it is to implement it in a secure fashion against side-channel attacks.

The Superposition Tweakay (STK) Construction: Jean *et al.* [JNP14] introduced the tweakey framework for designing adhoc TBC and its variant, the STK construction, specifically for AES-like designs. It was used to design the Deoxys-TBC [JNPS21]. Let $\tilde{E} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a TBC, such

that $(k+t)$ is a multiple of n, i.e. $(k+t) = cn$ where c is a small constant integer. A tweakey is constructed by concatenating the key and tweak, i.e. $T_K = K \| T$. T_K is divided into c blocks of n bits, each. During each round of the cipher, the c blocks are XORed together to generate the round key, then the round function is applied. Simultaneously, d parallel linear transforms are applied in parallel to the c blocks of the key. A single round of a TBC built using the STK framework is described in Algorithm 5.

3 An Unpredictable PRNG from Unpredictable TBCs

Consider a rekeying scheme $\tilde{R} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$, where $k = \lambda$ and $q_e = \lceil l/n \rceil$, $t \geq log_2(q_e+1)$. Then, we can define a PRNG as in Algorithm 1. It is easy to see that if $\tilde{R}$ is a secure TPRP, each loop calls the function with a unique tweak. Hence, for any adversary $\mathbf{A}$ against the PRNG that runs in time at most t, there is a TPRP adversary $\mathbf{B}$ that makes q_e queries and runs in time $O(t + q_e)$, such that

$$\epsilon_{\mathtt{prng}} \leq \mathbf{Adv}^{\mathtt{tprp}}_{\tilde{R}}(\mathbf{B}).$$

By observing that the rekeying scheme is used in counter-in-tweak mode (the tweak is a counter), this opens up different possibilities for building efficient designs. A popular design paradigm has started with the PSV-Enc proposed in [PSV15]. The design is depicted in Figure 1. The design makes calls to an ideal cipher. During each iteration, a call is made to the ideal cipher to *"increment"* the ephemeral key, while a second call is made (with a different constant TBC plaintext) to generate a block of the output. In practice, the initial key is generated from a master secret key using a key generation function. However, this is beyond the scope of the PRNG itself. This design has inspired many leakage-resilient designs, such as TEDT [BGP+19] and Romulus-T [GIK+22]. On the other hand, it suffers from two limitations:

1. It forces $\lambda \leq n$, where n is the block size of the TBC.
2. It can only achieve birthday bound security, due to collisions between the TBC keys and the possibility of the adversary to guess one of these keys. While this can be easily captured from an analysis of PSV-Enc as a PRNG. It is also captured in the analysis in [PSV15, Lemma (4)], where the security of PSV-Enc used for l-block encryption is $l \times \epsilon_1$ where ϵ_1 is the advantage of performing l single block encryptions. The single-block encryption security is related to a term denoted as ϵ_{2-sim} in [PSV15, Lemma (2)] which is (in simple terms) the security of the BC against adversaries that can observe 2 traces per key. This is capped (from ideal cipher security with random keys) to
$$\frac{\sigma^2}{2^n} + \frac{\sigma q_p}{2^n},$$
where σ is the number of keys used and q_p is the number of offline queries made to the ideal cipher. In the PRNG setting we are concerned with, we generate one message with q blocks. Thus, applying the bound in [PSV15]

would only lead to less than birthday bound security. However, in the black-box setting, a dedicated analysis may reclaim birthday bound security but not more.

The designs of TEDT and Romulus-T aim to solve the second limitation by achieving beyond birthday bound security, using extra counters and nonce. However, this requires extra assumptions, such as nonce-respecting adversaries or limited nonce-misuse, and is outside the scope and syntax of PRNGs. On the other hand, the first limitation is inherent in the design strategy. We can, conceptually, use $\lambda > n$ if the TBC supports $k > n$. However, this would require generating $k+n$ bits in each iteration, which would increase the number of calls to the TBC per iteration to at least 3 calls, which would also mean a stronger security model.

Another difference between our set-up and the set-up from [PSV15] and follow-up works is that in [PSV15] a full query either leaks or does not leak, but in our set-up, there is only one long query, and we are looking at the unpredictability of the next block, given the leakage from all the previous blocks have been observed.

This leads to an interesting research question:

Can we design a PSV-like PRNG with only two calls per block that takes more than n-bit input and does not suffer from a significant Time-Data trade-off security degradation like PSV-Enc?

In Figure 2, we propose a variant of PSV-Enc, which instead of using a cipher with n-bit key, it uses a TBC with n-bit tweak and k-bit key. The construction is a natural extension of PSV-Enc, where the ideal cipher is replaced by a TPRP. However, the interesting challenge is how to implement it without heavily protecting it, while maintaining some form of leakage-resilience. In this case, we can view the PRNG construction as having $\lambda = (n + k)$, where a k-bit part is fixed for all blocks, while an n-bit part is ephemeral in the same manner as PSV-Enc. Of course, this is not leakage-resilient in general except when the TBC is heavily protected, since K_m is fixed, which goes against the motivation of PSV-Enc. However, we will show that the synergy between Figure 2 and the STK design paradigm allows for a very efficient solution that does not require heavy protection of the TBC.

In the remainder of this section, we will focus on the black-box security of the proposed PRNG.

Algorithm 1 A framework for building PRNGs from rekeying schemes.

$K \xleftarrow{\$} \{0,1\}^{\lambda}$
$X \leftarrow \varepsilon$
for $i \in \{1, \ldots q\}$ **do**
    $C \leftarrow \tilde{R}(K, i, 0^n)$
    $X \leftarrow X \| C$
end for
return X

Algorithm 2 Our proposed PRNG from TBCs.

$(K_m, K_0) \xleftarrow{\$} \{0,1\}^k \times \{0,1\}^n$
$X \leftarrow \varepsilon$
for $i \in \{1, \ldots q\}$ **do**
    $C \leftarrow \tilde{E}_{K_m}^{K_{i-1}}(P_b)$
    $K_i \leftarrow \tilde{E}_{K_m}^{K_{i-1}}(P_a)$
    $X \leftarrow X \| C$
end for
return X

Theorem 1. *Let $\tilde{E} : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a TBC and let $G : \{0,1\}^{k+n} \to (\{0,1\}^n)^*$ be the PRNG given in Algorithm 2. Then, for any adversary $\mathbf{A}$ against G that generates q blocks and runs in time t, there exists an adversary $\mathbf{B}$ against $\tilde{E}$, such that,*

$$\mathbf{Adv}_{G}^{\mathtt{prng}}(\mathbf{A}) \leq \mathbf{Adv}_{\tilde{E}}^{\mathtt{tprp}}(\mathbf{B}) + \frac{q^2 + q}{2^n},$$

where $\mathbf{B}$ makes $2q$ TBC calls and runs in time $t^{'} = O(t + q)$.

Proof. In the first step of the proof, we replace $\tilde{E}$ with an ideal TBC $\tilde{\Pi}$, we call this Game 1, where E_1 is the event that the adversary wins according to the PRNG game definition in Section 2. Then, we define Game 2 as the game that terminates if for an index $i > 0$, with tweak K_i, there exists an index $0 \leq j < i$, such that $K_i = K_j$. E_2 that the adversary wins according to the PRNG game definition in Section 2. Using the hybrid argument,

$$|\Pr[E_2] - \Pr[E_1]| \leq \sum_{i=1}^{q} \frac{i}{2^n} \leq \frac{q^2}{2^n}.$$

Next, we define Game 3 where we replace $\tilde{\Pi}$ with a random function $\tilde{F}$ with the same domain and range. In this case, we can see that each tweak K_i is used at most twice, and Games 2 and 3 can only be distinguished if for any index i, $\tilde{F}(K_i, P_a) = \tilde{F}(K_i, P_b)$, which can happen with probability $1/2^n$. Thus,

$$|\Pr[E_3] - \Pr[E_2]| \leq \sum_{i=1}^{q} \frac{1}{2^n} = \frac{q}{2^n}.$$

Finally, Game 3 is indistinguishable from an ideal PRNG. Thus,

$$\Pr[E_3] = 0.$$

The bound follows from adding the transition probabilities.

3.1 Unpredictability of the Next Block: A leakage resiliency target for PRNGs

Unpredictability as a leakage resiliency target for TBCs was discussed in [BGPS21]. The adversary observes a certain number of evaluations of the TBC with chosen plaintexts with their corresponding leakages[3], then wins if they can predict the outcome of a new evaluation of the TBC that has not been observed before. Unpredictability as a target for leakage-resilient symmetric-key cryptography is a well-established topic that dates earlier than [BGPS21], and has been discussed in multiple works [DS09,DJS19]. We follow the definition of [BGPS21] since it is the most mature for TBCs. In this section, we recall the definition of unpredictability with leakage given in [BGPS21], and propose the unpredictability of the next block as a security goal for PRNGs. Then, we show that if $\tilde{E}$ is unpredictable with leakage, then our PRNG achieves unpredictability of the next block. The definition below is adapted from [BGPS21], removing the inverse function and adapting notation.

Definition 1. *A TBC* $\tilde{E} : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *with a leakage function* $\mathsf{L}_{\mathsf{Eval}}$ *is* $(q_l, q_c, t, \epsilon_{\mathtt{upl1}})$*-unpredictable with forward leakage if for any* (q_l, q_c, t)*-adversary* $\mathbf{A}$*, we have*

$$\mathbf{Adv}^{\mathtt{upl1}}_{\tilde{E}}(\mathbf{A}) \leq \epsilon_{\mathtt{upl1}},$$

where the UPL1 game is defined in Algorithm 3, and $\mathbf{A}$ *makes at most* q_c *construction queries with leakage, such that each tweak is repeated at most twice, and* q_l *modelling queries with leakage to the modelling oracle* L*.*

Berti *et al.* [BGPS21] define modelling queries as queries made to the TBC with chosen plaintext and chosen key to model the leakage function. They are meant to capture attacks such as template attacks. They are similar to primitive queries in the ideal cipher model.

Algorithm 3 TBC Unpredictability with Leakage Game

```
Initialize :
K ←$ {0,1}^k
L ← φ
for X ∈ {0,1}^n do
    T[X] ← 0
end for
Enc(M, T) :
if T[T] = 2 then
    return ⊥
end if
T[T] ← T[T] + 1
C ← Ẽ_K^T(M)
l_e ← L_Eval(M, T; K)
L ← L ∪ {(M, T, C)}
return (C, l_e)
Finalize :
(M, T, C) ← A^{L,Enc}
if (M, T, C) ∈ L then
    return 0
else if C = Ẽ_K^T(M) then
    return 1
else
    return 0
end if
```

As discussed earlier and studied in [Men20], a rekeying scheme is nothing but a TBC with special implementation properties. Thus, consider a rekeying

[3] [BGPS21] discusses strong unpredictability, which also allows chosen ciphertext queries. However, this is not needed for our construction.

scheme that is $(q_l, q_e, t, \epsilon_{\texttt{upl1}})$-unpredictable as the building block of Algorithm 1. Algorithm 4 describes a security game where an adversary $\mathbf{A}$ makes a single query to the PRNG in Algorithm 1 with output $k + ln$, observing the output string and the associated leakage, and tries to guess the next n bits. It is easy to see that if the rekeying scheme is $(q_l, q_e, t^{'}, \epsilon_{\texttt{upl1}})$-unpredictable, then the PRNG is $(q_l, q, t, \epsilon_{\texttt{upl1}})$-next-block-unpredictable, where $t^{'} = O(t + q)$.

Algorithm 4 PRNG Unpredictability of the Next Block with Leakage Game

```
1: Initialize :
2: K ←$ {0,1}^k
3: PRNG(q) :
4: X ← ε
5: l_p ← φ
6: for i ∈ {1, ..., q} do
7:     X ← X‖R̃(K, i, 0^n)
8:     l_p ← l_p ∪ {L_R(K, i, 0^n)}
9: end for
10: return (X, l_p)
11: Finalize :
12: C ← A^{L,PRNG}
13: if C = R̃(K, q + 1, 0^n)
    then
14:     return 1
15: else
16:     return 0
17: end if
```

Definition 2. *We say the PRNG G in Algorithm 1 with associated leakage function is $(q_l, q, t, \epsilon_{\texttt{prng-upl1}})$-unpredictable if for any (q_l, q_e, t)-adversary $\mathbf{A}$, we have*

$$\mathbf{Adv}_G^{\texttt{prng-upl1}}(\mathbf{A}) \leq \epsilon_{\texttt{prng-upl1}},$$

where $\mathbf{Adv}_G^{\texttt{prng-upl1}}(\mathbf{A})$ is the advantage that for any $0 \leq i \leq q$, $\mathbf{A}$ observes the first in bits and predicts the next n bits.

Theorem 2. *Given a rekeying scheme $\tilde{R}$ that is $(q_l, q_e, t^{'}, \epsilon_{\texttt{upl1}})$-unpredictable, a PRNG that is defined according to Algorithm 1 is $(q_l, q_e, t, q_e\epsilon_{\texttt{upl1}})$-unpredictable where $t^{'} = O(t + q_e)$.*

Proof. The proof follows from simple hybrid argument where the adversary calls Finalize in Algorithm 4 after observing each block and the associated leakage, and wins if the function returns 1 at any call. Since the rekeying scheme is $(q_l, q_e, t^{'}, \epsilon_{\texttt{upl1}})$-unpredictable, then at any time Finalize is called, the advantage is at most $\epsilon_{\texttt{upl1}}$, the function is called at most q_e times. $t^{'} = O(t + q_e)$ since the time taken an adversary against the rekeying scheme is bounded by at most the time needed by the adversary against the PRNG in addition to a constant overhead per query.

Theorem 3. *Let $\tilde{E} : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a TBC and let $G : \{0,1\}^{k+n} \rightarrow (\{0,1\}^n)^*$ be the PRNG given in Algorithm 2. If $\tilde{E}$ is $(q_l, 2q_e, t^{'}, \epsilon_{\texttt{upl1}})$-unpredictable, G is $(q_l, q_e, t, \epsilon_{\texttt{prng-upl1}})$-unpredictable, where*

$$\epsilon_{\texttt{prng-upl1}} \leq q_e\epsilon_{\texttt{upl1}} + \frac{q_e^2}{2^n}$$

and $t^{'} = O(t + q_e)$.

Proof. First, we define a hybrid game that terminates if for an index $i > 0$, with tweak K_i, there exists an index $0 \leq j < i$, such that $K_i = K_j$. This is analyzed in Game 2 of the proof of Theorem 1. Otherwise, we notice that the TBC is never called with the same tweak twice. Using the hybrid argument in Theorem 2, we get the full bound.

Theorem 3 can be understood as follows: if $\tilde{E}$ is secure against key recovery, with K_m as the key, and secure against 2-trace attacks trying to leak any K_i, then the PRNG remains unpredictable (up to birthday bound). The rest of the paper is dedicated to reaching a lightweight realization of Algorithm 2 when $q_l = 0$, *i.e.* for non-profiled attacks. This is done as a proof of concept, since the protection against profiled attacks with low number of traces, such as SPA and template attacks, are usually cheaper than non-profiled attacks. For instance, Simple Power Analysis (SPA) and template attacks may be made harder using cheaper countermeasures such as hiding and shuffling. That being said, we note that both our proposed design and PSV-Enc equally require protection against template and SPA-like attacks.

Comparison to PSV-Enc: Algorithm 2 can clearly be more costly than PSV-Enc. For instance, it has a static key that needs to be protected. For a general TBC, this means heavy protection of the full TBC. Luckily, as we shall see in Section 4, the cost of protecting the static key in STK TBCs is minimal. On the other hand, PSV-Enc suffers from the security bound $q_e q_p / 2^n$, since its black-box security is in the ideal cipher model and the adversary can guess one of the keys by making q_p queries to the primitive and q_e queries to PSV-Enc, such that $q_p q_e \approx 2^n$. It also restricts the key size to n, making it harder to use TBCs with small block but large tweakey. For instance, consider Algorithm 2 with a TBC with 128-bit tweakey and 64-bit block, and an adversary that successfully guesses the static part of the key K_m, the adversary can recover one of the ephemeral keys with $q_p q_e \approx 2^{64}$. Since the probability of guessing K_m is 2^{-64}, the adversary needs $q_p q_e \approx 2^{128}$ and $q_p \geq 2^{64}$ to get a close to 1 advantage of guessing a full key. If q_e is limited to 2^{32} blocks, then q_p can go up to 2^{96}. These attacks are captured in Theorem 1 by the computational term of the TPRP security. On the other hand, PSV-Enc with a 64-bit cipher would only tolerate $q_p \approx 2^{32}$ in this case, which is far from secure.

Based on this, while we focus our proof-of-concept in the rest of the paper on Deoxys-TBC, we believe our proposed scheme has much broader design space, and can be used with TBCs such as Skinny-64-128 or Skinny-64-192.

3.2 Comparison to BBB Secure PSV-Enc-like Encryption

As discussed earlier, TEDT and Romulus-T aim to make PSV-Enc-like constructions BBB secure using large tweaks. In particular, TEDT replaces the constants P_a and P_b with two counters. This prevents the formation of short periodic cycles, in case two of the keys collide. It also includes a random tweak T: Since TEDT is an Authenticated Encryption with Associated Data (AEAD) scheme, T

is the authentication tag generated using a PRF MAC. However, in our PRNG set-up, this tag does not affect the security as it is can be treated as a public constant. Romulus-T goes a step further where the counters are included as part of the tweak, which improves the security bounds slightly. These ideas do not help at least one of the aspects of the problem we are trying to solve. Namely, it is still not possible to use the unpredicatbility with leakage assumption proposed in [BGPS21], since this assumption requires a secret TPRP-style TBC, and not an ideal cipher which these schemes use. One the other hand, these ideas can be used in conjunction with our proposal. In such a combined construction, the TBC has larger tweak space and the tweak is appended by a public counter and two public constants. This makes sure that even if two keys collide, all the tweakeys are unique. This improves the black-box security to

$$\epsilon_{\mathtt{prng}} \leq \mathbf{Adv}^{\mathtt{tprp}}_{\tilde{E}}(2q_e, t).$$

However, when it comes to unpredictability with leakage, things are more subtle. If we want to maintain that the adversary can only observe two traces for the same ephemeral key, key collisions still affect the security. However, we could relax the assumption to n traces instead of two, and rely on bounding the probability of getting a multi-collision of size $> n$. This would again remove the birthday-bound term, but make the unpredictability assumption stronger and the analysis more involved. Since our goal is to introduce the possibility of using unpredictability assumptions with a static key to build leakage-resilient PRNGs, we leave a dedicated analysis of its BBB security with leakage as future-work.

We also note that the techniques proposed by Chen *et al.* [CLMP21] can be used to generalize the PSV-Enc/2-PRG construction to have higher security using a block cipher with $2n$-bit key and n-bit blocks, and 3 calls to the block cipher per output block. However, this approach is less efficient than ours as it requires three calls to the block cipher, while we only need two. One may argue that these calls are to a block cipher and not a TBC, which makes them cheaper, but this is not necessarily true in practice, as the difference in cost between a BC with $2n$-bit key and a TBC with n-bit key and n-bit tweak is minimal, and does not offset the cost of a full extra call. For instance, AES-256 requires 16 AES rounds, while Deoxys-128-256 requires 14 AES rounds and the total size of the key and tweak is $2n$ bits. In many cases, the same TBC is used as the block cipher, which makes such approach significantly more expensive than ours. Besides, the approach of Chen *et al.* still does not satisfy our goal of using the unpredictability assumption.

4 Lightweight Realization of Algorithm 2 Using the STK Framework

Consider an instance of the STK framework with n-bit block and $(2n)$-bit tweakey. A single round of the construction is described in Algorithm 5. L_0 and L_1 are two different linear transforms that satisfy certain security properties of the STK

construction. What we observe is that if one of the tweakey components, say T_1, is fixed, and the other component T_0 is selected uniformly at random, then the round key K_{r_i} has a uniform distribution, and the same applies for the output S. However, during one execution, the round keys are not independent (since T_0 is used in all the rounds up to a linear transformation). If the TBC consists of r rounds, then T_0 (up to different linear transformations) is used r times. It is a common assumption that if the implementation is protected against SPA with a small number of traces, and T_0 is used only in a small number of calls, then it cannot be recovered by an adversary. Hence, the only target for SCA in Algorithm 5 is T_1, which needs to be masked.

Algorithm 5 A single round of the STK framework.

1: $T_0, T_1 \xleftarrow{n} T$
2: $T_0 \leftarrow L_0(T_0)$
3: $T_1 \leftarrow L_1(T_1)$
4: $K_{r_i} \leftarrow T_0 \oplus T_1$
5: $S \leftarrow R_{r_i}(S) \oplus K_{r_i}$

Algorithm 6 A secure implementation single round of the STK framework.

1: $T_0, A_1, B_1 \xleftarrow{n} T$
2: $T_0 \leftarrow L_0(T_0)$
3: $A_1 \leftarrow L_1(A_1)$
4: $B_1 \leftarrow L_1(B_1)$
5: $K_{r_i} \leftarrow T_0 \oplus A_1$
6: $K_{r_i} \leftarrow K_{r_i} \oplus B_1$
7: $S \leftarrow R_{r_i}(S) \oplus K_{r_i}$

Algorithm 6 adopts this idea. During a call to the TBC, we consider $|T| = 3n$, and divide it into 3 components (or shares). Each of the three components appears indistinguishable from a block selected uniformly at random, except that at lines 3 and 4, $A_1 \oplus B_1 = T_1$. K_{r_i} in line 6 is an unmasked version of the round key. However, since A_1 is first added to T_0, which is sampled uniformly, then the adversary observing K_{r_i} gains no more information than the adversary observing T_0.

4.1 Application to Deoxys-TBC with unprotected round function

Theoretically while we show that the key unmasking is done securely, in practice, similar to masking schemes, it does require careful implementation on the device. In the following section, we describe a series of experiments that applies this design strategy in practice. These experiments are conducted on modified versions of the reference implementation of the Deoxys-TBC [JNPS21].

Measurement setup Our measurement setup to validate the secure unmasking of the round key consists of the Chipwhisperer CW308 UFO platform, with STM32F303 as the target board. The STM32F303RCT6 is an ARM Cortex-M4 CPU with 256KB flash, 48KB SRAM and 72MHz operating frequency. The device is programmed with a C implementation of Deoxys-TBC, compiled with arm-none-eabi-gcc compiler using -O3 optimization level. In the software implementation the entire round subtweakeys are computed upfront and stored, and the cipher round functions are called later, as seen in the power trace from Figure 3. The clock and communication to the target is handled by the Chip-

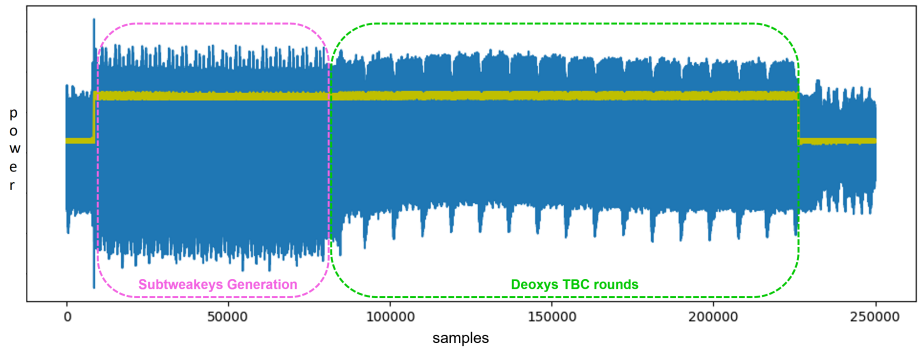

Fig. 3. Deoxys-TBC power trace showing subtweakey generation and cipher rounds.

whisperer Husky device. The target runs at 44 MHz and the power measurement of the device is captured using the LeCroy WavePro 404 oscilloscope at a sampling frequency of 250MS/s. We employed the Test Vector Leakage Assessment (TVLA) method [GJJR11] to validate the leakages during key unmasking.

Deoxys TBC designs We analyzed four different implementations of the Deoxys-TBC. They are summarized in the Table 1 along with the expected security goals and the experimental results. For the unprotected Deoxys-TBC instance (Deoxys-TBC-ORIG) with 384-bit tweakey, we used the reference C implementation [JNPS21]. As discussed in Section 3, the first 128 bits of the tweakey component is the static key K_m while the second 128 bits is the ephemeral key K_i. The third component is not used and is set to 0, but it can be dedicated to the counter in a higher level construction. In other words, the 384-bit key is assigned as: $K_m \| K_i \| 0^{128}$. For the protected Deoxys-TBC instance (Deoxys-TBC-P) with 384-bit tweakey the first 128 bits of the tweakey are masked, the second 128-bit component is selected uniformly at random. If the masked key is $K_m^a \| K_m^b$, then the protected implementation takes $K_m^a \| K_m^b \| K_i \| 0^{128}$.

We performed two types of TVLA:

- Plaintext TVLA: We fix K_m and acquire traces using fixed vs. random plaintext. In our construction, the plaintext is a public constant, so plaintext leakage bears little value to our analysis. However, this was done to verify the soundness of our approach: when one of the tweakey components is changing randomly, the round function should not leak.
- Key TVLA: In this case, we fix the plaintext and acquire two sets of traces, one with K_m fixed to a single value and one with K_m changing randomly. This is done to make sure that the key schedule does not experience 1^{st} order leakage and that we can detect glitches and micro-architectural leakage.

We have also performed TVLA without changing K_i to make sure that our set-up can detect leaky implementations.

Implementation	Description	Security Goal	Result
Deoxys-TBC-ORIG	Reference implementation provided with [JNPS21]	Leakage in the key schedule	PASS
Deoxys-TBC-P	One of the 128-bit components of the tweakey is masked according to Algorithm 6	No 1^{st} order leakage	FAIL
Deoxys-TBC-P-RP	Similar to Deoxys-TBC-P but the CPU registers are randomly pre-charged	No 1^{st} order leakage	PASS
Deoxys-TBC-P-SH	Similar to Deoxys-TBC-P but the operations used to compute the round keys are randomly shuffled	No 1^{st} order leakage	PASS

Table 1. Tested implementations with their security goals.

With the protected Deoxys-TBC-P design, during the key TVLA, we observed leakages in multiple rounds of the round key generation function. To understand the root-cause we generated the assembly code for the roundkey generation function. After analyzing it we concluded that the observed leakages were potentially due to micro-architectural and transient glitches that invalidate our assumptions on masking. This shows that, similar to any masking schemes, it does require some care when implementing the design on a device, as there could be leakages due to transient glitches from platform architecture and compiler optimizations. To address these micro-architectural leakages for the current platform, we improved the Deoxys-TBC-P design by pre-charging the CPU user registers with random values before each round of the tweak key generation (Deoxys-TBC-P-RP). This pre-charging had a 4.01% overhead for the round key generation and 1.24% increase for the overall cipher. Similarly, we also tested a secure design where the sub-parts of the tweaks that is XOR-ed during the secure unmasking are handled in a random shuffled order (Deoxys-TBC-P-SH). This countermeasure resulted in a 34.82% overhead for the round key generation and 10.62% overhead for the overall cipher. This shuffling Deoxys-TBC-P-SH design, while comparatively slower, should be secure and should eliminate any potential micro-architectural leakages, similar to the one discussed above, on any platform. We used an LFSR to generate the randomness for the Deoxys-TBC-P-RP and Deoxys-TBC-P-SH designs. This is done as an example, and in a real-world implementation a secure random source should be used. It is expected that a practical system would have its own randomness source.

Experimental results Figure 4 shows the plaintext TVLA leakage for the Deoxys-TBC-ORIG implementation with 1,000 traces, when the tweak is fixed. And figure 5 shows the plaintext TVLA leakage with 1 million traces, when the tweak is selected uniformly at random. Using a random tweak with the unprotected design eliminates the observed plaintext TVLA leakages from the cipher rounds.

Similarly, Figure 6 shows the key TVLA leakage for the Deoxys-TBC-ORIG implementation with a fixed tweak for 1,000 traces. As expected, the unprotected design shows TVLA leakages with t-values over the 4.5 bounds, in the round key generation.

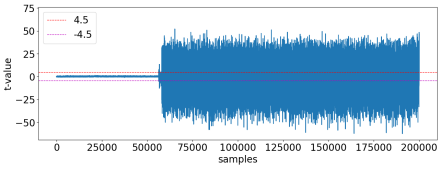

Fig. 4. Plaintext TVLA: Deoxys-TBC-ORIG with fixed tweak, 1000 traces.

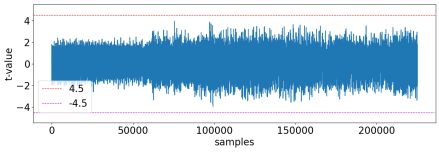

Fig. 5. Plaintext TVLA: Deoxys-TBC-ORIG with random tweak, 1 million traces.

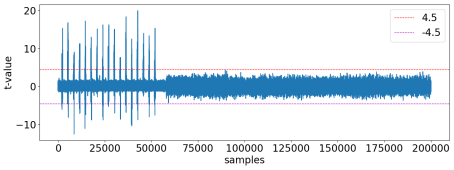

Fig. 6. Key TVLA: Deoxys-TBC-ORIG with random tweak, 1000 traces.

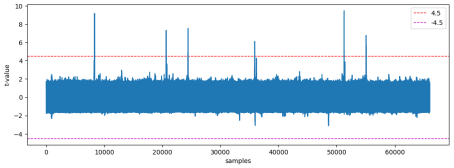

Fig. 7. Key TVLA: Deoxys-TBC-P subtweakey generation with random tweak, 10000 traces.

For the protected Deoxys-TBC-P design as discussed in algorithm 6 and in section 4, the masked key gets securely unmasked during the subtweakey generation. When the design is used with a random tweak, it is expected be secure against a key TVLA, but we observed leakage in the roundkey generation as shown in the Figure 7. By generating and analyzing the assembly instructions of this implementations, we attributed the leakage to transient glitches and micro-architectural leakage, which understandably are more probable during our secure unmasking operation. We studied the register pre-charging (Deoxys-TBC-P-RP) and shuffled order XOR (Deoxys-TBC-P-SH) designs to eliminate such platform dependent leakages. Figure 8 and Figure 9 shows the key TVLA with a random tweak for the Deoxys-TBC-P-RP and Deoxys-TBC-P-SH designs, respectively. Both the protected designs show no leakage and are secure against the key TVLA with 1 million traces. We refer to the supplementary materials for a full list of TVLA graphs.

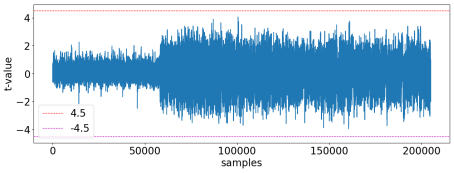

Fig. 8. Key TVLA: Deoxys-TBC-P-RP with random tweak, register pre-charge, 1 million traces.

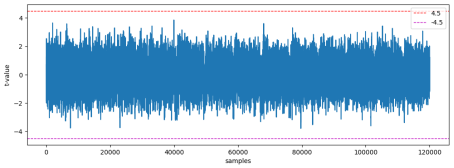

Fig. 9. Key TVLA: Deoxys-TBC-P-SH with random tweak, secure shuffled operations, 1 million traces.

Table 2 summarises the total time taken by the target for the round key generation and the full cipher operation, at 44MHz, for all the four implementations discussed in the section 4.1. It is to be noted that for Deoxys-TBC-P-RP and Deoxys-TBC-P-SH implementations, the time measured is inclusive of randomness generation required for register pre-charging and for creating the shuffle buffer.

Algorithm	Roundkey generation (μs)	Deoxys-TBC (μs)
Deoxys-TBC-ORIG	213.71	798.96
Deoxys-TBC-P	263.42	850.11
Deoxys-TBC-P-RP	273.99	860.68
Deoxys-TBC-P-SH	355.15	940.41

Table 2. Time needed for the round key generation and the full cipher execution of different implementations.

For reference, a first-order masked AES-128 [ans] implementation requires 1375.38 μs on the same platform. It is to be noted that the Deoxys-TBC-ORIG implementation used for the work is the reference implementation without any modifications and hence it can be optimized further.

5 Conclusions

In this paper, we studied a new PRNG construction inspired by the PSV-Enc construction and leveraging the superposition property of STK-based ciphers. We have provided theoretical analysis of our construction both as a black-box PRNG and as an unpredictable-with-leakage PRNG, based on the unpredictability-with-leakage assumption for TBCs, popularized in [BGPS21]. We have also provided experimental proof-of-concept results using TVLA on Deoxys-TBC showing that the cost of eliminating first-order observable leakage in our proposal ranges from $\approx 1 - 10\%$.

Applications The proposed PRNG can be used in any setting where a leakage-resilient PRNG is needed and that the protocol can maintain a $2k+n$-bit state, where $2k$ bits are the masked static key K_m and n bits are the ephemeral key K_i. It was shown in [IKMP20] that STK-based TBCs allow maintaining a static key at no additional (storage) cost beyond the implementation of the TBC itself, due to the properties of the tweakey schedule. Note that we require k to be a multiple of n so that the security arguments on the STK framework hold and that the mask is refreshed after each TBC call. Two notable applications come to mind:

- Stream ciphers: we can use our construction to build a leakage-resilient stream cipher with (K_m, K_0) as the initial key. We note that using a leakage-resilient PRNG as a stream cipher requires care of decryption leakage and/or authentication, as discussed in details in [BGP+19,BBC+20].
- Subkey generation: we can use our construction as a stateful subkey generation function in a bigger scheme. For example, it can be used as the rekeying function in the rekeying-based ΘCB mode proposed by Mennink in [Men20].

Future Work Several natural follow-up research directions can arise from our work:

- Evaluating our PRNG with a small-block TBC, *e.g.*, Skinny-64-192.
- Evaluating the cost of implementing this PRNG in hardware. While our software experiments show that the cost of implementation is extremely cheap ($5\% \sim 18\%$ overhead), it also shows that it is not trivial. Hardware may present a new set of challenges that we may need to study.
- Designing a complete AEAD scheme using our PRNG as an underlying primitive, either as the encryption part using levelled implementations, or as the subkey generation function for a rekeying-based AEAD.
- Security analysis of our scheme combined with tweak counters and nonce using the unpredictability with leakage assumption, or another BBB-secure variant.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments. We would like to also thank Thomas Peyrin (NTU) for sharing the reference implementation of Deoxys-TBC that allowed us to perform our experiments. We acknowledge the generous support of Seagate Technology towards this study.

References

[ans] Masked aes-128 implementation in c for the stm32f3/stm32f4 platforms. https://github.com/ANSSI-FR/SecAESSTM32/tree/3b9ed68a4576255636634ec539079476cd5bbc92.

[BBC+20] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography: a practical guide through the leakage-resistance jungle. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part I 40*, pages 369–400. Springer, 2020.

[BGP+19] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Tedt, a leakage-resist aead mode for high physical security applications. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):256–320, Nov. 2019.

[BGPS21] Francesco Berti, Chun Guo, Thomas Peters, and François-Xavier Standaert. Efficient leakage-resilient macs without idealized assumptions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 95–123. Springer, 2021.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The skinny family of block ciphers and its low-latency variant mantis. In *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II 36*, pages 123–153. Springer, 2016.

[BKP+18] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext integrity with misuse and leakage: definition and efficient constructions with symmetric primitives. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pages 37–50, 2018.

[CLMP21] Yu Long Chen, Atul Luykx, Bart Mennink, and Bart Preneel. Systematic security analysis of stream encryption with key erasure. *IEEE Transactions on Information Theory*, 67(11):7518–7534, 2021.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Transactions on Information Forensics and Security*, 15:2542–2555, 2020.

[DEM+17] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. Isap–towards side-channel secure authenticated encryption. *IACR Transactions on Symmetric Cryptology*, pages 80–105, 2017.

[DEMM14] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, and Florian Mendel. On the security of fresh re-keying to counteract side-channel and fault attacks. In *International Conference on Smart Card Research and Advanced Applications*, pages 233–244. Springer, 2014.

[DJS19] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: the case of authenticated encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 209–240. Springer, 2019.

[DKM+16] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. Towards fresh and hybrid re-keying schemes with beyond birthday security. In *Smart Card Research and Advanced Applications: 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers 14*, pages 225–241. Springer, 2016.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 293–302. IEEE, 2008.

[DS09] Yevgeniy Dodis and John Steinberger. Message authentication codes from unpredictable block ciphers. In *Advances in Cryptology-CRYPTO 2009: 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 267–285. Springer, 2009.

[GGM85] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In *Advances in Cryptology: Proceedings of CRYPTO 84 4*, pages 276–288. Springer, 1985.

[GIK+22] Chun Guo, Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Security proof for romulus-t, 2022.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Joshua Jaffe, and Pankaj Rohatgi. A testing methodology for side channel resistance. 2011.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the titans: The romulus and remus families of lightweight aead algorithms. *IACR Transactions on Symmetric Cryptology*, 2020(1):43–120, May 2020.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, pages 463–481. Springer, 2003.

[JNP14] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Tweaks and keys for block ciphers: the tweakey framework. In *Advances in Cryptology–ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014, Proceedings, Part II 20*, pages 274–288. Springer, 2014.

[JNPS21] Jérémy Jean, Ivica Nikolic, Thomas Peyrin, and Yannick Seurin. The deoxys AEAD family. *J. Cryptol.*, 34(3):31, 2021.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19*, pages 388–397. Springer, 1999.

[Men20] Bart Mennink. Beyond birthday bound secure fresh rekeying: Application to authenticated encryption. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 630–661. Springer, 2020.

[MSGR10a] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In *Progress in Cryptology–AFRICACRYPT 2010: Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings 3*, pages 279–296. Springer, 2010.

[MSGR10b] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In *Progress in Cryptology–AFRICACRYPT 2010: Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings 3*, pages 279–296. Springer, 2010.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *International conference on information and communications security*, pages 529–545. Springer, 2006.

[nxp] Leakage resilient primitive (lrp) specification. https://www.nxp.com/docs/en/application-note/AN12304.pdf.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 96–108, 2015.

[SPY13] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In *Annual Cryptology Conference*, pages 335–352. Springer, 2013.

[UHIM23] Rei Ueno, Naofumi Homma, Akiko Inoue, and Kazuhiko Minematsu. Fallen sanctuary: A higher-order and leakage-resilient rekeying scheme. *Cryptology ePrint Archive*, 2023.

[VCMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In *Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18*, pages 740–757. Springer, 2012.

Suplementary Material

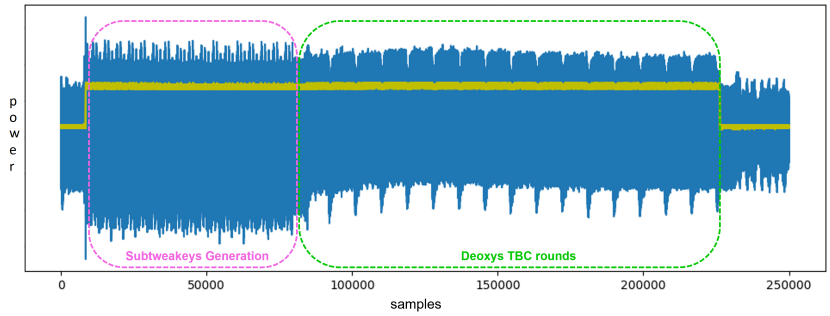

Fig. 10. Deoxys-TBC power trace showing subtweakey generation and cipher rounds.

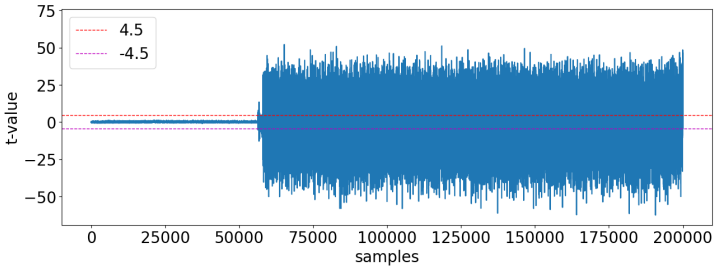

Fig. 11. Plaintext TVLA: Deoxys-TBC-ORIG with fixed tweak, 1000 traces.

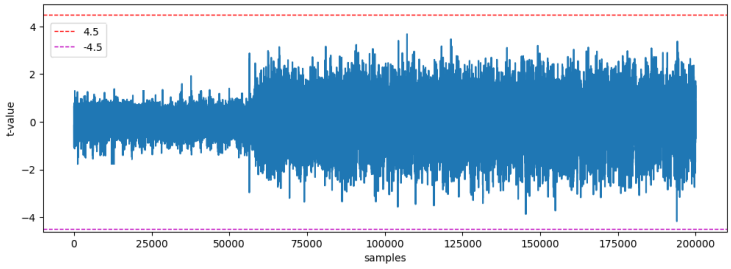

Fig. 12. Plaintext TVLA: Deoxys-TBC-ORIG with random tweak, 1000 traces.

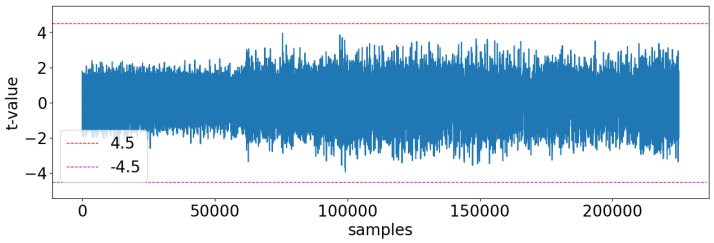

Fig. 13. Plaintext TVLA: Deoxys-TBC-ORIG with random tweak, 1 million traces.

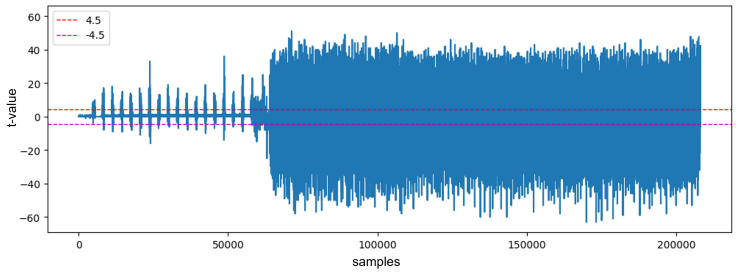

Fig. 14. Key TVLA: Deoxys-TBC-ORIG with fixed tweak, 1000 traces.

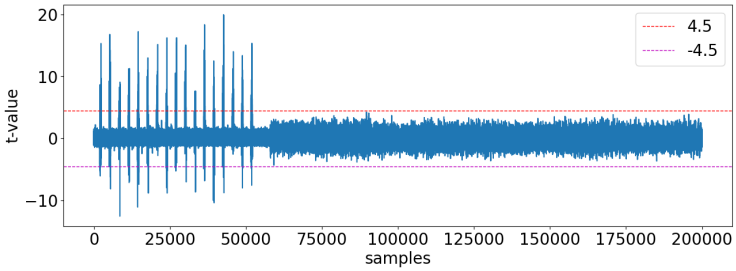

Fig. 15. Key TVLA: Deoxys-TBC-ORIG with random tweak, 1000 traces.

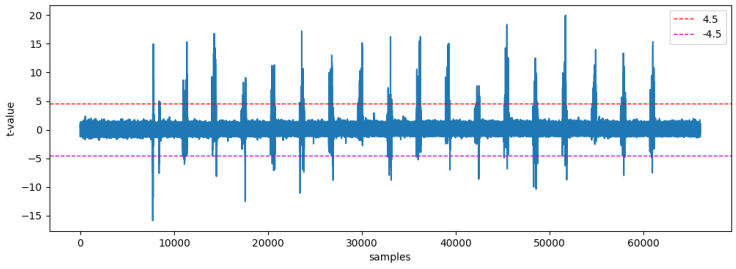

Fig. 16. Key TVLA: Deoxys-TBC-P subtweakey generation with fixed tweak, 1000 traces.

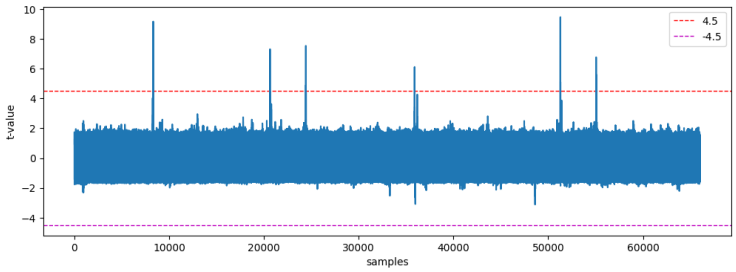

Fig. 17. Key TVLA: Deoxys-TBC-P subtweakey generation with random tweak, 10000 traces.

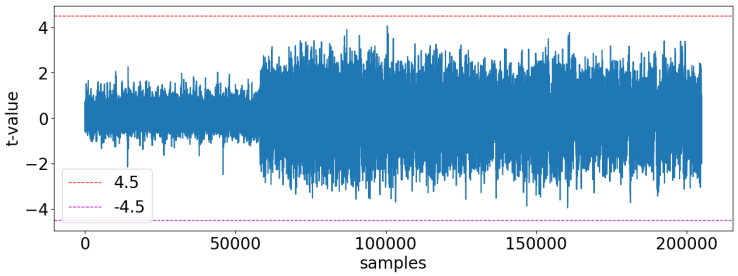

Fig. 18. Key TVLA: Deoxys-TBC-P-RP with random tweak, register pre-charge, 1 million traces.

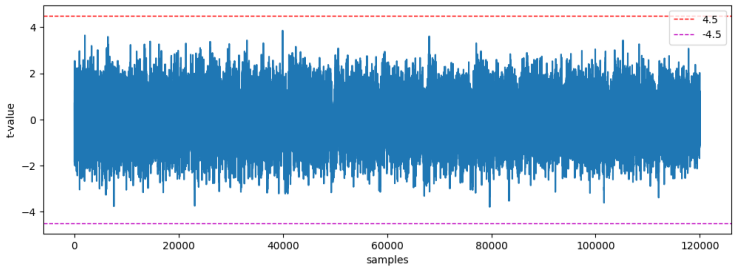

Fig. 19. Key TVLA: Deoxys-TBC-P-SH key generationn with random tweak, secure shuffled operations, 1 million traces.