


NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

# Lightweight Leakage-Resilient PRNG from TBCs using Superposition

Mustafa Khairallah[†], Srinivasan Yadhunathan[†], and Shivam Bhasin[‡]

April 10, 2024

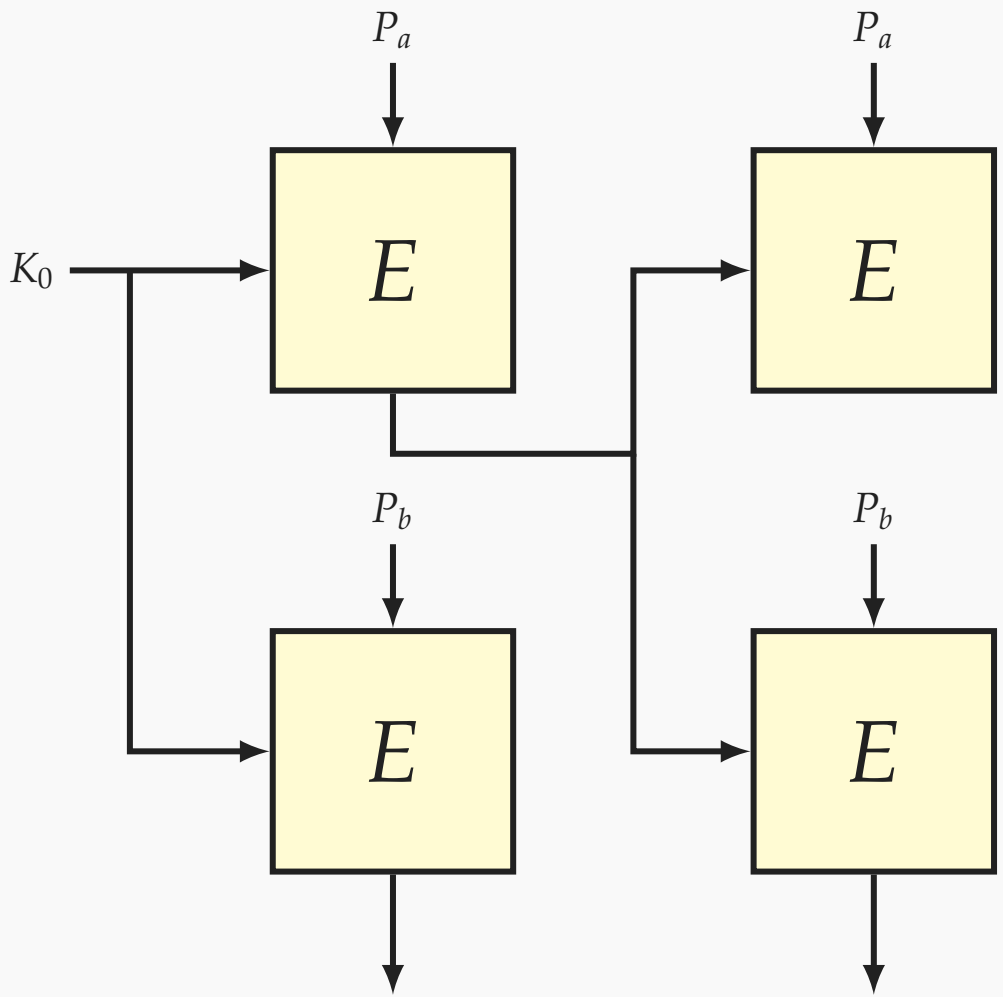
[†] Seagate Research Group, Singapore
[‡] Nanyang Technological University, Singapore

# Outline

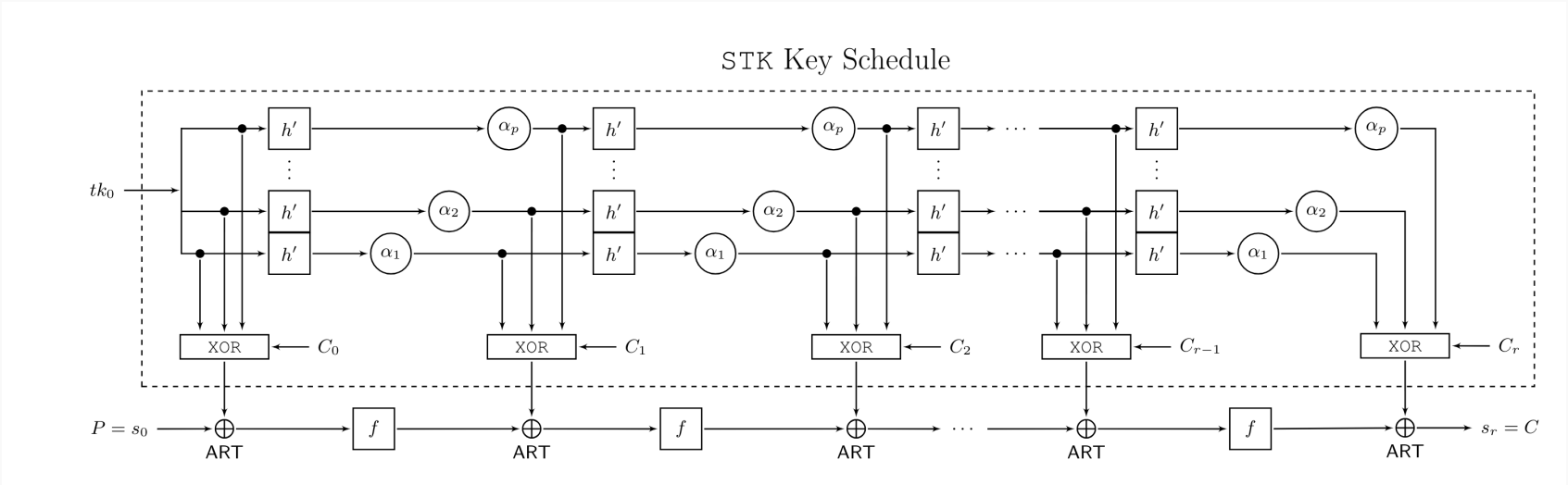
1. Background
2. Unpredictability with Leakage
3. Practical Realization: Application to Deoxys-TBC

# Background

# 2-PRG/PSV-ENC



# The STK construction




[1]

[1] Image courtesy of: https://eprint.iacr.org/2014/831.pdf

# Limitations of 2-PRG

- As a PRNG, PSV-Enc can only achieve birthday bound security, even in the black box setting.
- ★ This can be improved by careful choice of constants and/or use of tweakable block ciphers.

# Limitations of 2-PRG

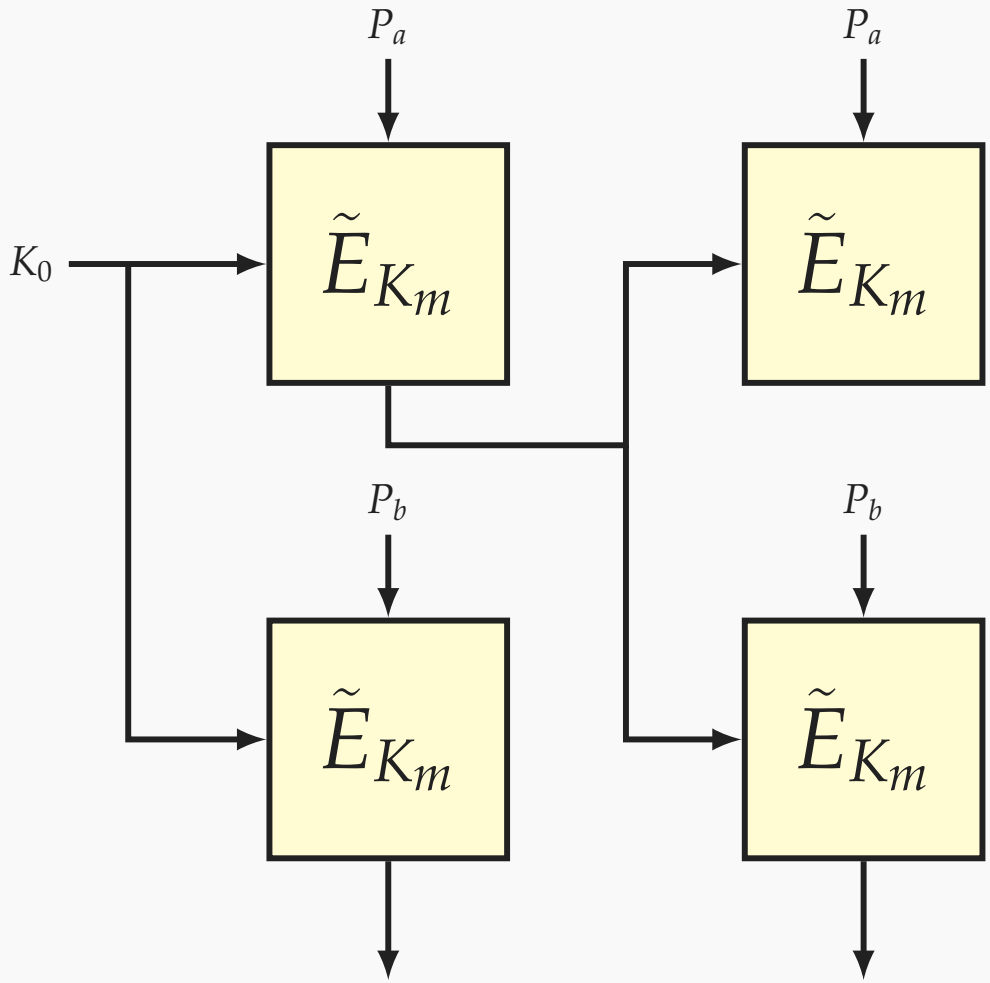
- **The BC calls in PSV-Enc do not (and should not) share a common secret key, making its analysis in the black-box setting rely on easier ideal cipher model or multi-key analysis of the BC.**
- **In terms of leakage resilience, this makes it harder to use the unpredictability with leakage assumption, recently introduced by Berti *et al.* [2]. This limitation, unlike the previous one, cannot be addressed using a TBC.**

[2] https://eprint.iacr.org/2021/1250.pdf

# Limitations of 2-PRG

- **PSV-Enc is designed such that the subkey size is equal to the block size $n$.**
- **For 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.**

# Proposed PRNG



# Unpredictability with Leakage

# UNPREDICTABLE TBC WITH FORWARD LEAKAGE

Initialize :
$K \xleftarrow{\$} \{0,1\}^k$
$\mathcal{L} \leftarrow \phi$
**for** $X \in \{0,1\}^n$ **do**
    $\mathcal{T}[X] \leftarrow 0$
**end for**

Enc$(M, T)$ :
**if** $\mathcal{T}[T] = 2$ **then**
    **return** $\perp$
**end if**
$\mathcal{T}[T] \leftarrow \mathcal{T}[T] + 1$
$C \leftarrow \tilde{E}_K^T(M)$
$\mathsf{l_e} \leftarrow \mathsf{L_{Eval}}(M, T; K)$
$\mathcal{L} \leftarrow \mathcal{L} \cup \{(M, T, C)\}$
**return** $(C, \mathsf{l_e})$

Finalize :
$(M, T, C) \leftarrow \mathbf{A}^{\mathsf{L},\mathsf{Enc}}$
**if** $(M, T, C) \in \mathcal{L}$ **then**
    **return** $0$
**else if** $C = \tilde{E}_K^T(M)$ **then**
    **return** $1$
**else**
    **return** $0$
**end if**

# Unpredictability of the next block

Initialize :
$K \xleftarrow{\$} \{0,1\}^k$

PRNG$(q)$ :
$X \leftarrow \varepsilon$
$\mathsf{l_p} \leftarrow \phi$
**for** $i \in \{1, \ldots, q\}$ **do**
    $X \leftarrow X \| \tilde{R}(K, i, 0^n)$
    $\mathsf{l_p} \leftarrow \mathsf{l_p} \cup \{\mathsf{L_R}(K, i, 0^n)\}$
**end for**
**return** $(X, \mathsf{l_p})$

Finalize :
$C \leftarrow \mathbf{A}^{\mathsf{L},\mathsf{PRNG}}$
**if** $C = \tilde{R}(K, q+1, 0^n)$ **then**
    **return** $1$
**else**
    **return** $0$
**end if**

# Main result

**Algorithm 1** 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

*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 1. 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* $t' = O(t + q_e)$.

# Interpertation and limitations

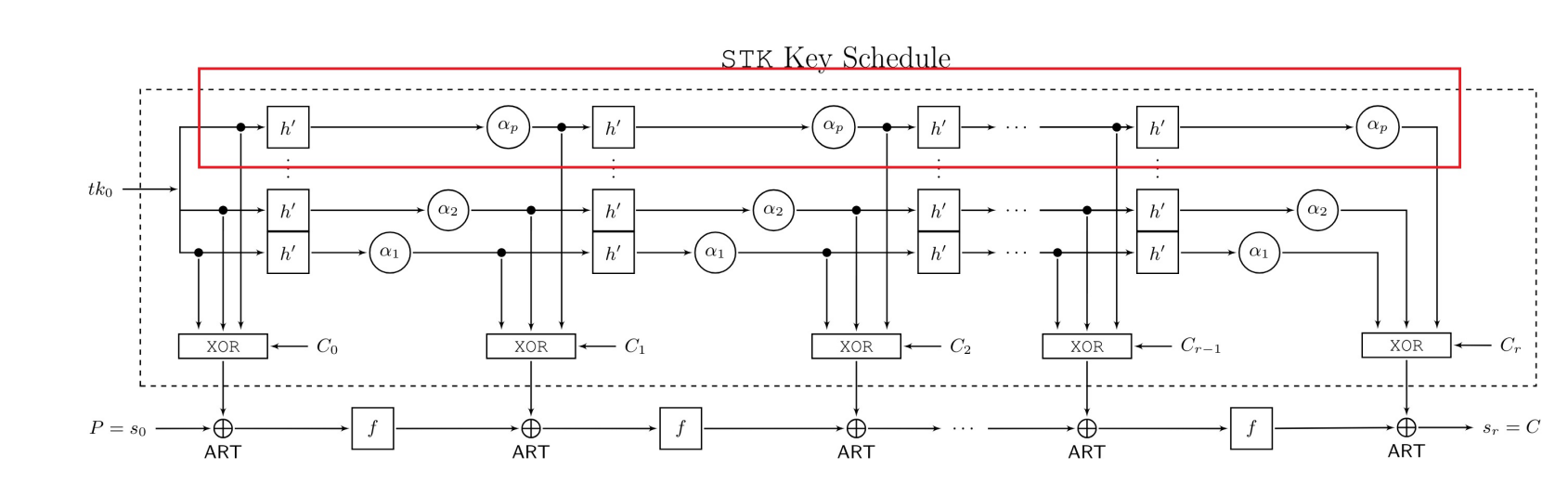
- If $\tilde{E}$ is secure against key recovery, with $K_m$ as the key, and secure against 2-trace attacks trying to leak any $K_i$, then the PRNG remains unpredictable (up to birthday bound).
- This eliminates the TMDT attack but keeps the birthday bound. In reality, this birthday bound security translates to the length of the queries and not the overall complexity when multiple seeds are used. Thus, it may be improved by dedicated analysis of the schemes using this construction.
- It can also be improved using a counter, increasing the tweak size.

# Practical Realization: Application to Deoxys-TBC

# Challenges of realization

- Building an unpredictable TBC with single-key without heavy countermeasures.
- How to verify the security of this primitive?
- Understanding the sources of leakage in the implementation.

# LEAKAGE IN STK TBCS




- The plaintexts are public and fixed.
- One tweakey changes every two calls.
- One tweakey is fixed.

# Masking and securely unmasking the fixed key

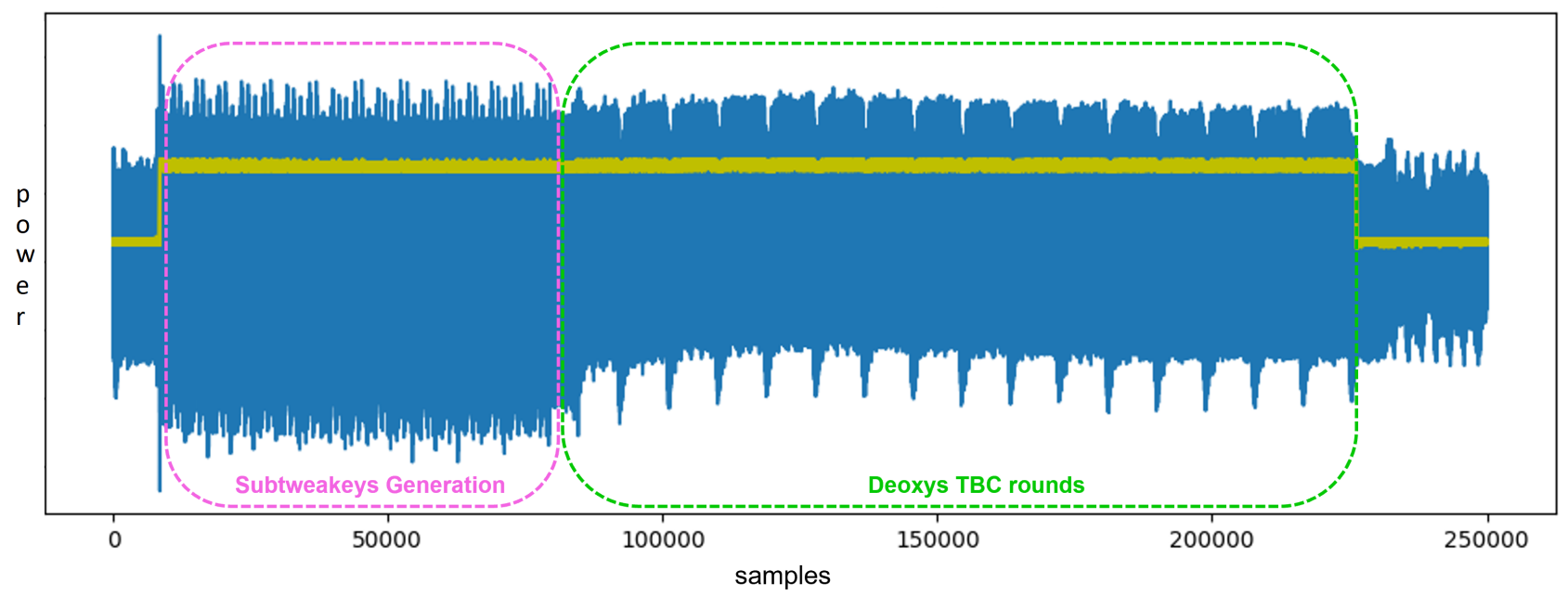
**Algorithm 2** A single round of the STK framework.

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: $S \leftarrow R_{r_i}(S) \oplus K_{r_i}$

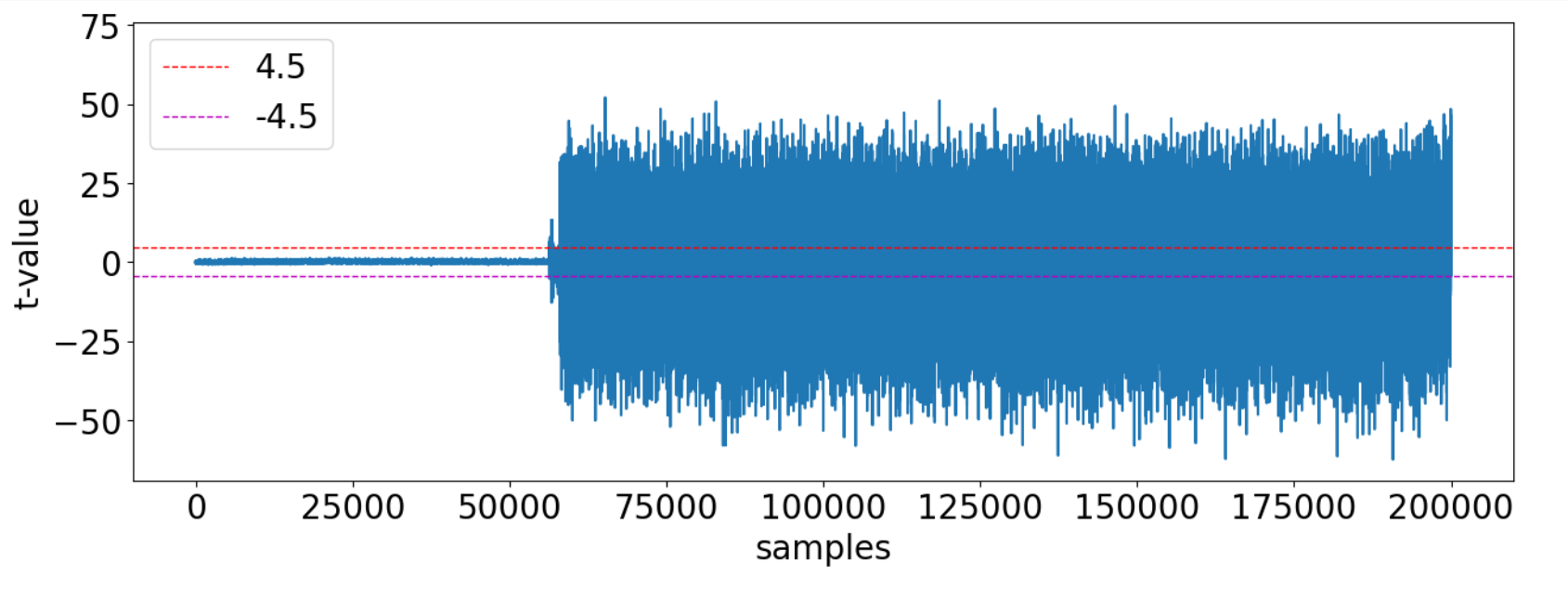
**Algorithm 3** A secure implementation single round of the STK framework.

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: $S \leftarrow R_{r_i}(S) \oplus K_{r_i}$

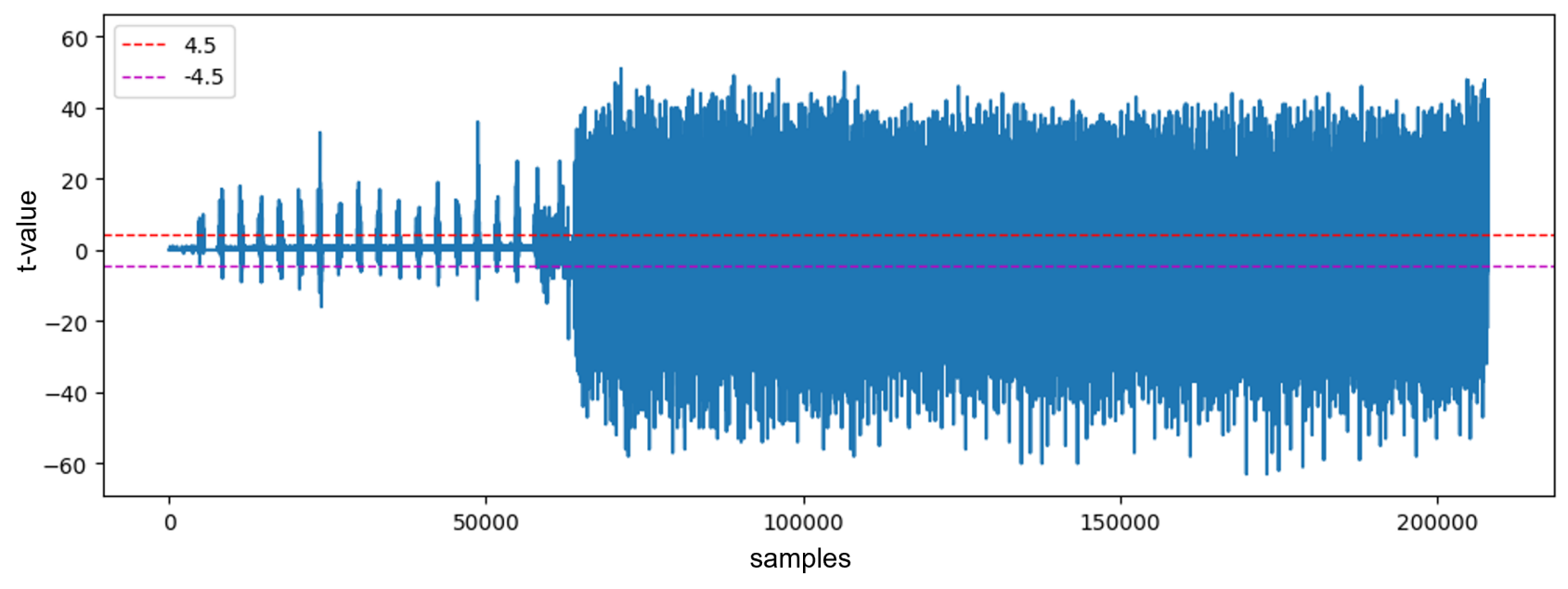
# DEOXYS SOFTWARE IMPLEMENTATION



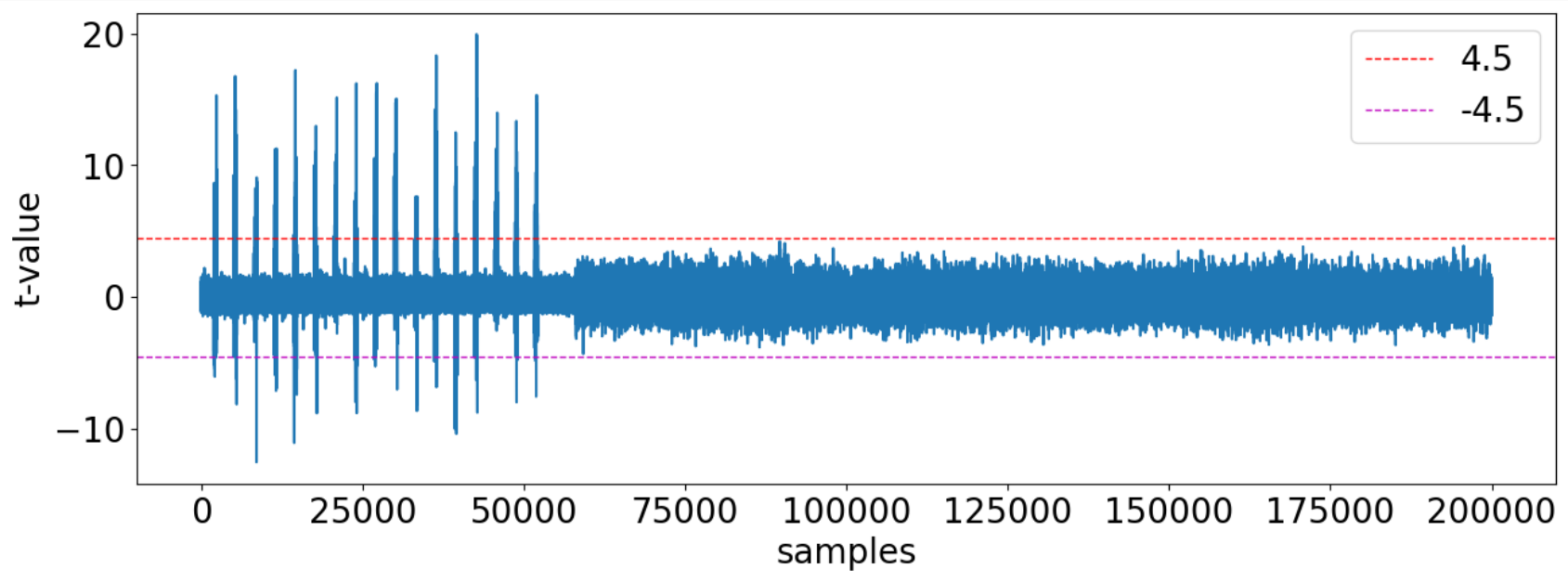
# PLAINTEXT TVLA



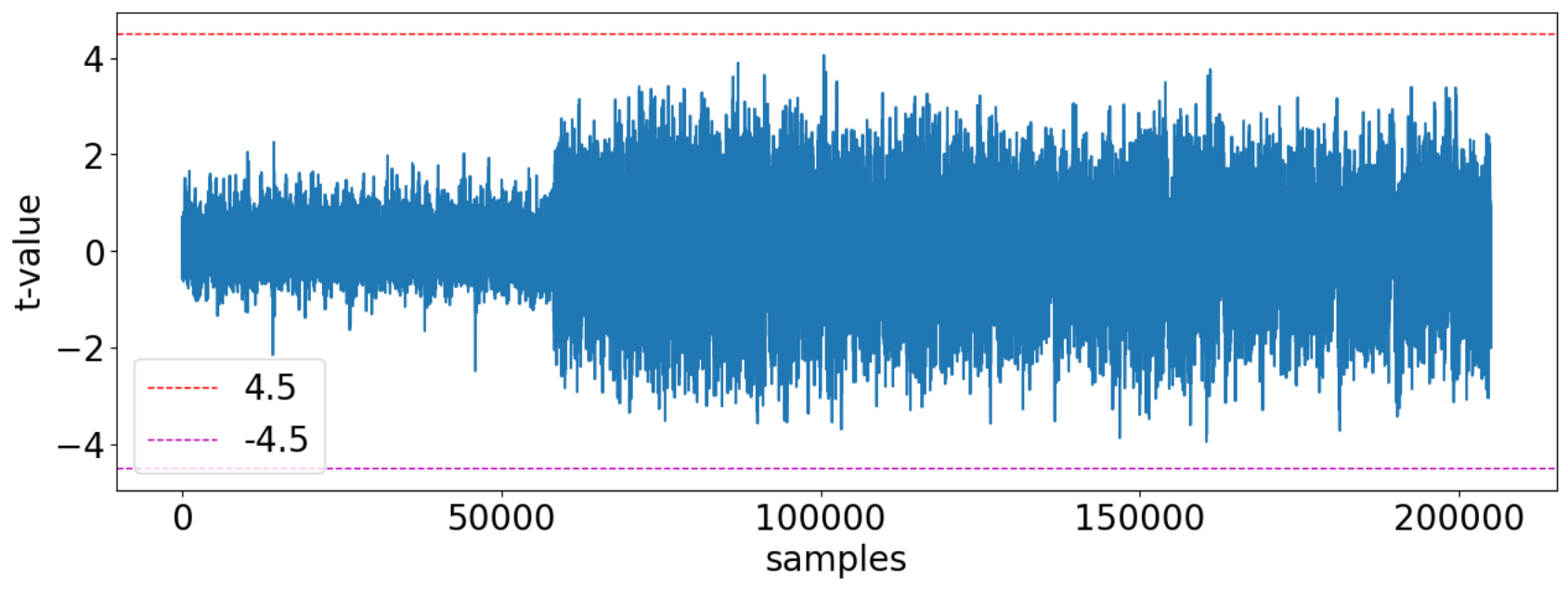
# KEY TVLA: NO RANDOMNESS, NO KEY MASKING



# KEY TVLA: RANDOM TWEAK, NO KEY MASKING



# KEY TVLA: RANDOM TWEAK, KEY MASKING



# Open Questions

- More experiments to verify the claims.
- The physical security is based on the tweak being random, while theoretically it is only unpredictable.
- Using this construction in modes such as AEAD.

Thanks for listening.