INTRINSIC ID

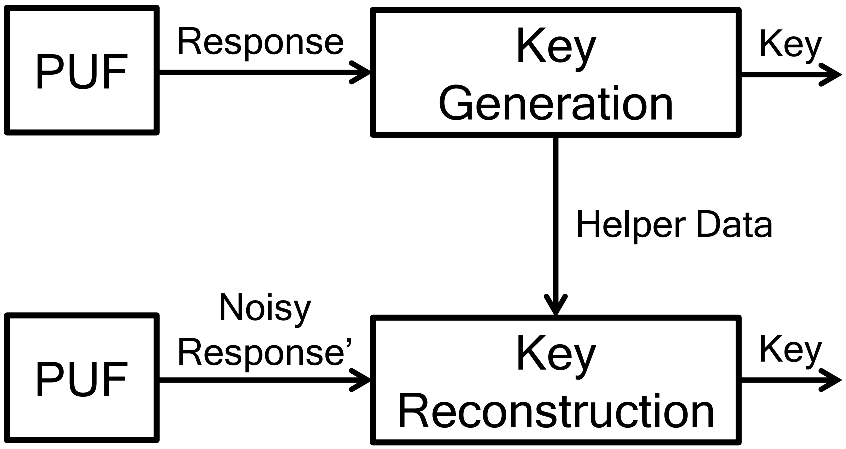
Secure your digital life™

Secure Key Generation from Biased PUFs

Roel Maes, Vincent van der Leest, Erik van der Sluis (Intrinsic-ID)
Frans Willems (TU Eindhoven)

Introduction

- PUF-based key generation



- Reliability:

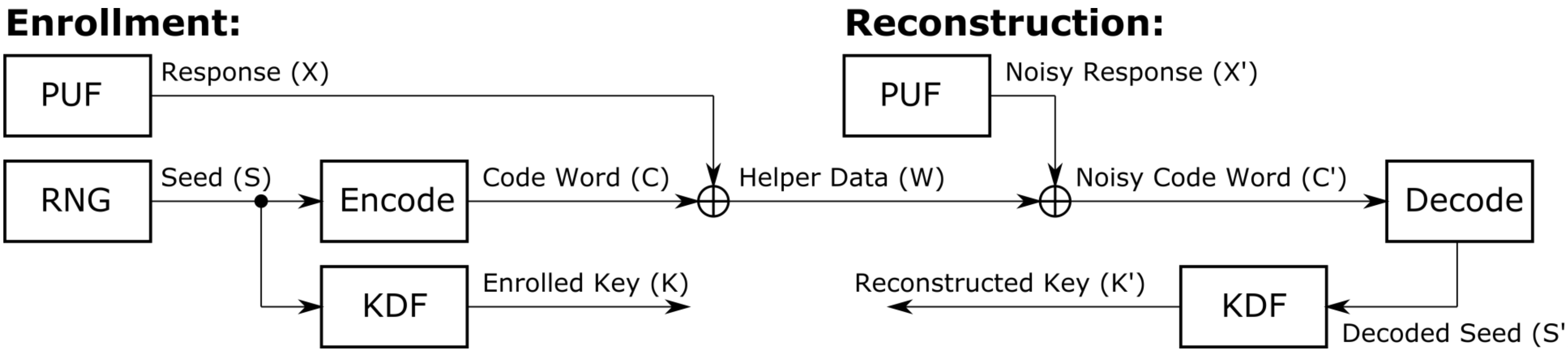
  If Response ≈ Noisy Response **then** Key = Key'

- Security:

  <u>**If** Response is sufficiently unpredictable (w.r.t. its length)</u> **then** Key is fully unpredictable, **even though** Helper Data is known

- What if PUF response is not full-entropy?

Setting: PUF-based Key Generation



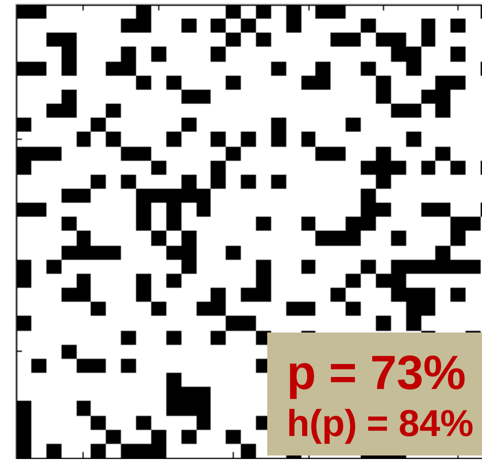
- Code-offset construction
  - Helper data = offset between PUF response and random code word
  - Key = derived from random seed which determines code word
- Security?
  - KDF(.) = cryptographically secure key derivation function
  - S = input <u>with sufficient entropy</u> to derive a key from
- $H(S \mid W) = ?$
  - $H(S \mid W) = H(S) - I(S\,;W) = |S| - I(S\,;W) = |S| - I(S\,;X + \text{Encode}(S)) = ?$

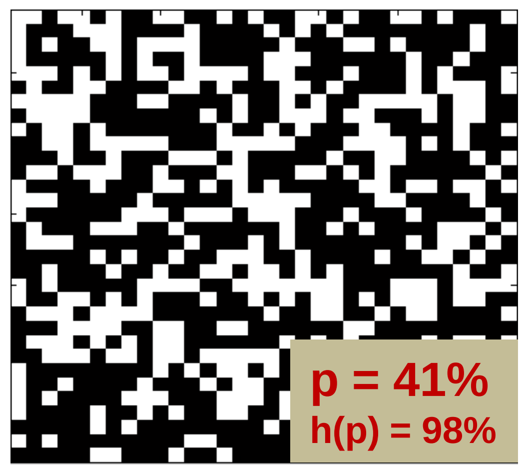
Leakage Problem: General

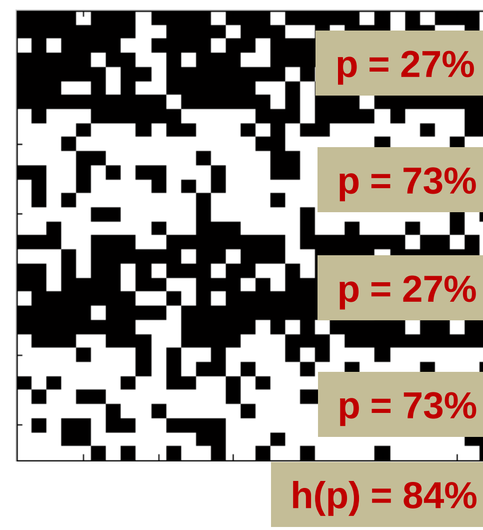
- $H(\text{S} \mid \text{W}) = \underbrace{|\text{S}|}_{\text{Initial Seed Entropy}} - \underbrace{I(\text{S} ; \text{W})}_{\text{Entropy Leakage}}$ $\boxed{\text{= Entropy left for key derivation}}$
- Entropy leakage?
  - $I(\text{S} ; \text{W}) = I(\text{S} ; \text{X} + \text{S}*\mathbf{G})$ ($\mathbf{G}$ = generator matrix of block code)
    $= |\text{S}| - [H(\text{X}) - H(\text{X}*\mathbf{H}^\text{T})]$ ($\mathbf{H}$ = parity-check matrix)
  - If X fully random ($H(\text{X}) = |\text{X}|$), then $I(\text{S} ; \text{W}) = 0$
    → no entropy leakage! and $H(\text{S} \mid \text{W}) = |\text{S}|$
  - If X not fully random, then $I(\text{S} ; \text{W}) \geq 0$
    → possible entropy leakage and $H(\text{S} \mid \text{W}) = H(\text{X}) - H(\text{X}*\mathbf{H}^\text{T})$
- $H(\text{X}*\mathbf{H}^\text{T}) = ?$
  - Depends on distribution of X *and* on code structure $\mathbf{H}^\text{T}$
  - Difficult to compute exactly for the general case
  - <u>Upper bound:</u> $H(\text{X}*\mathbf{H}^\text{T}) \leq |\text{X}*\mathbf{H}^\text{T}| = (n - k)$ (for an (n, k) block code)
    → results in upper bound on leakage, or <u>lower bound on remaining entropy</u>

Leakage Problem: Bias Only

- X in $\{0,1\}^n$ not fully random because of <u>bias only</u>
  - Most common and obvious cause of PUF non-randomness
  - p-biased PUF → for an unseen response bit $Pr(X_i = 1) = p$
  - $H(X) = n*h(p)$
    ($h(.)$ = binary entropy function)
- $H(X*\mathbf{H}^T) = ?$
  1. For simple codes (e.g. repetition) → <u>closed expression</u>
  2. For short codes (e.g. n < 32) → <u>exhaustively</u> determine distribution of $X*\mathbf{H}^T$
  3. Otherwise → use <u>upper bound</u> (n – k)

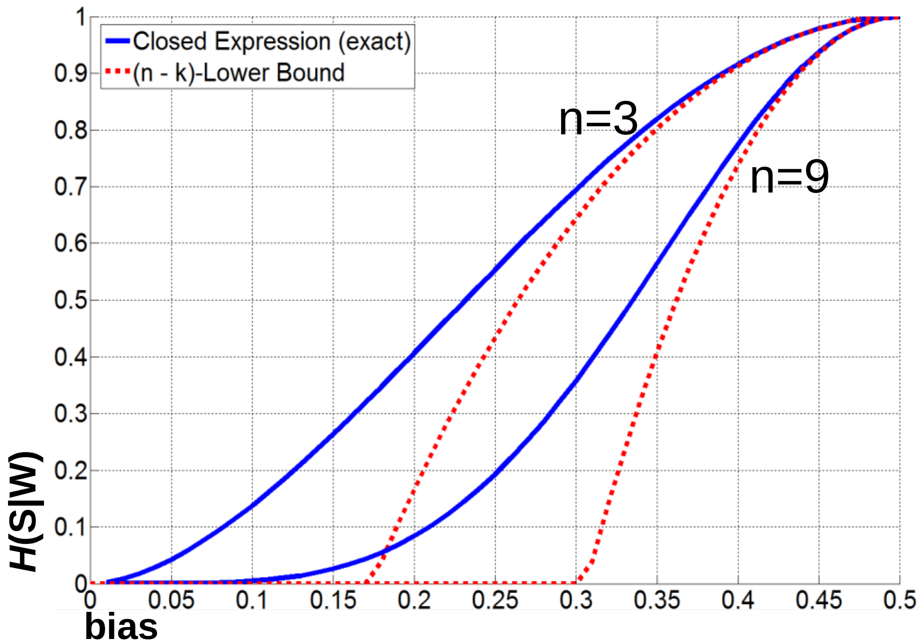






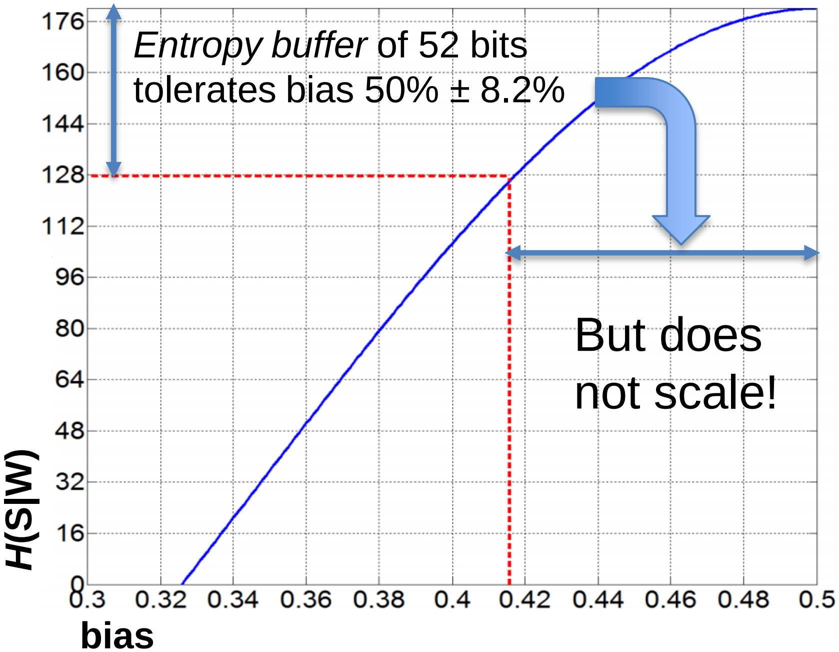
Leakage Problem: Effect of Bias

- For repetition codes:



  - Lower bound very pessimistic for bias not close to 50%
    (cf. "repetition code pitfall", Koeberl et al., HOST-2014)
  - But still significant entropy loss due to bias

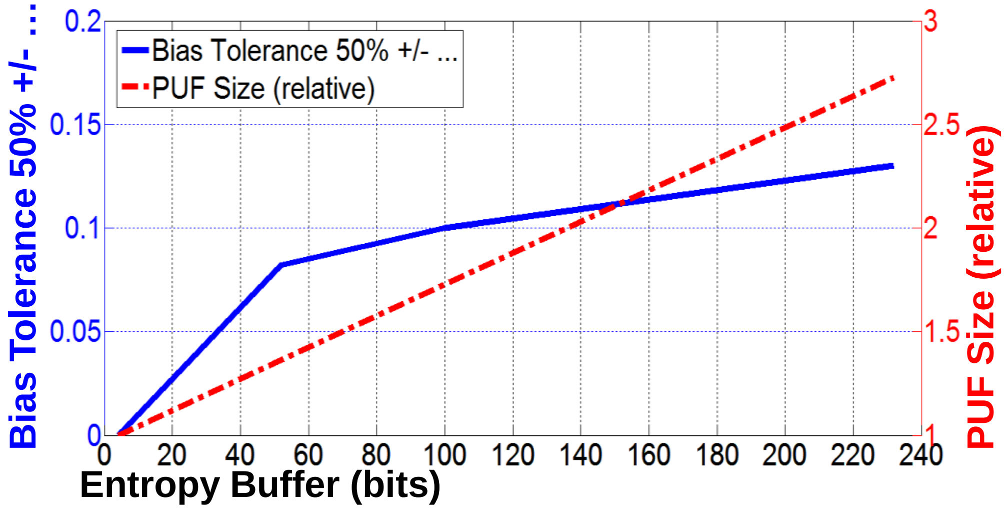
- For full key generator (ex.):

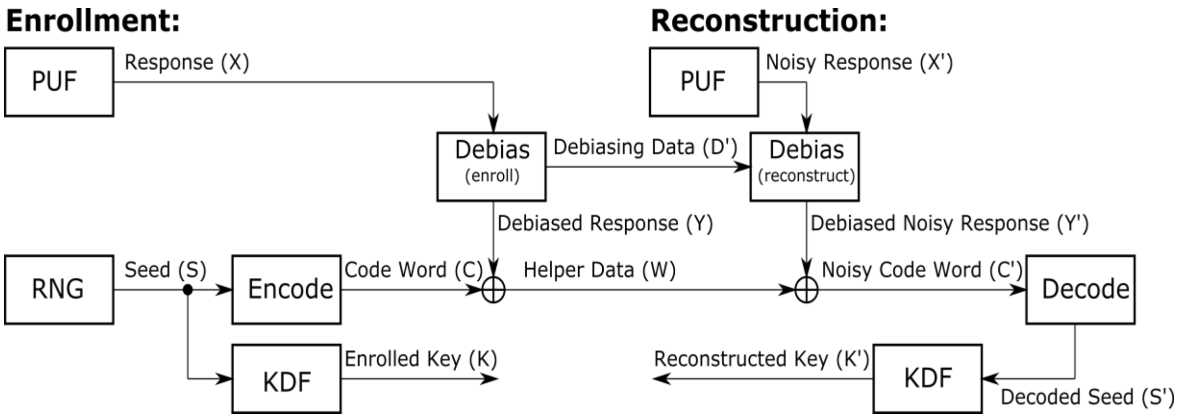


  - Based on concatenated Repetion(8,1) o Golay(24,12) code
    (van der Leest et al., CHES-2012)
  - Generates 128-bit key with $>1\text{-}10^{-6}$ reliability in presence of <15% noise
  - **Secure for 41.8% < bias < 58.2%**

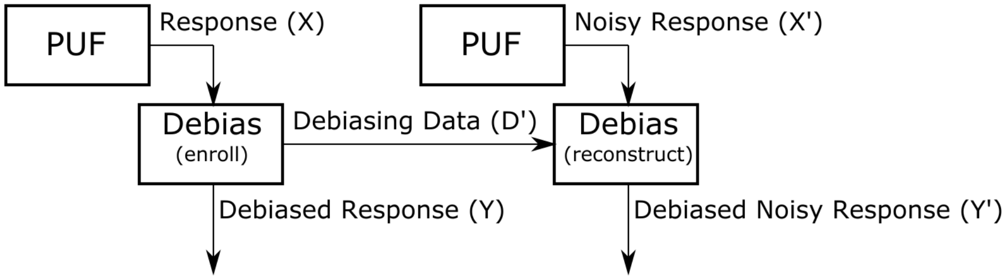
Solution: Debiasing

- Bias tolerance does not scale with entropy buffer
  - PUF size does scale with entropy buffer!
  - Bias tolerance limited even when buffer $\rightarrow \infty$
- Other solution needed
  - For bias levels above limit
  - For PUF size efficiency
- **Debiasing** prior to code-offset
  - Debiasing (helper) data





Solution: Criteria



1. Reliability

Debiasing cannot compromise reliability of key generation
(e.g. hash(X) removes bias but blows up bit error rate of PUF response)

2. Efficiency

If $|Y| < |X|$ then debiasing induces overhead
→ debiasing overhead should be limited and as small as possible

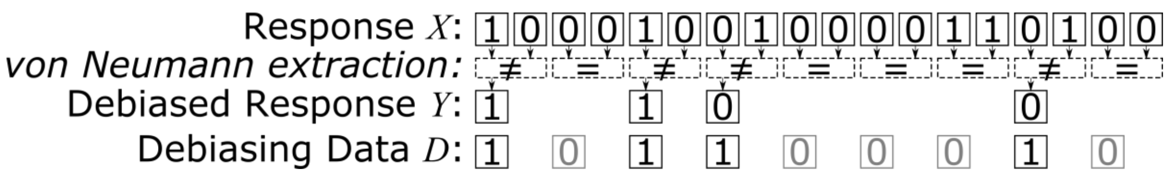
3. Leakage

a) Debiasing should take care of leakage due to bias, also for large bias
b) Debiasing data should not induce additional leakage: $I(S ; W) = I(S ; (W, D))$

4. Reusability

Classic code-offset construction is reusable (cf. Boyen, ACM-CCS-2004):
one enrollment leaks the same as many enrollments: $I(S ; W) = I(S_i ; (W_1, W_2, \ldots))$
It would be nice to keep this property: $I(S ; (W, D)) = I(S_i ; (W_1, D_1, W_2, D_2, \ldots))$

Debiasing Variant 1: “Classic” Von Neumann



Consider consecutive pairs:

- Discard (0, 0) and (1, 1)
- Retain first bit of (0, 1) and (1, 0)
- Discard/retain choice is stored in debiasing data

1. **Reliability:** Bit error rate is hardly affected

   Main advantage of Von Neumann-like methods!

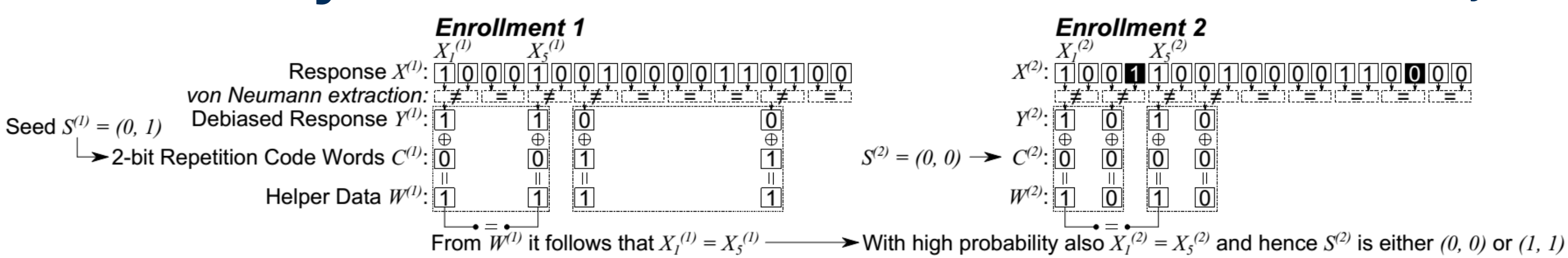
2. **Efficiency:** debiasing overhead factor > 4

   Function of bias and reliability, e.g.: bias = 30% and |Y| = 1000 bits are needed with reliability > $1 - 10^{-6}$, then |X| needs to be ≥ 5334 → overhead factor 5.3

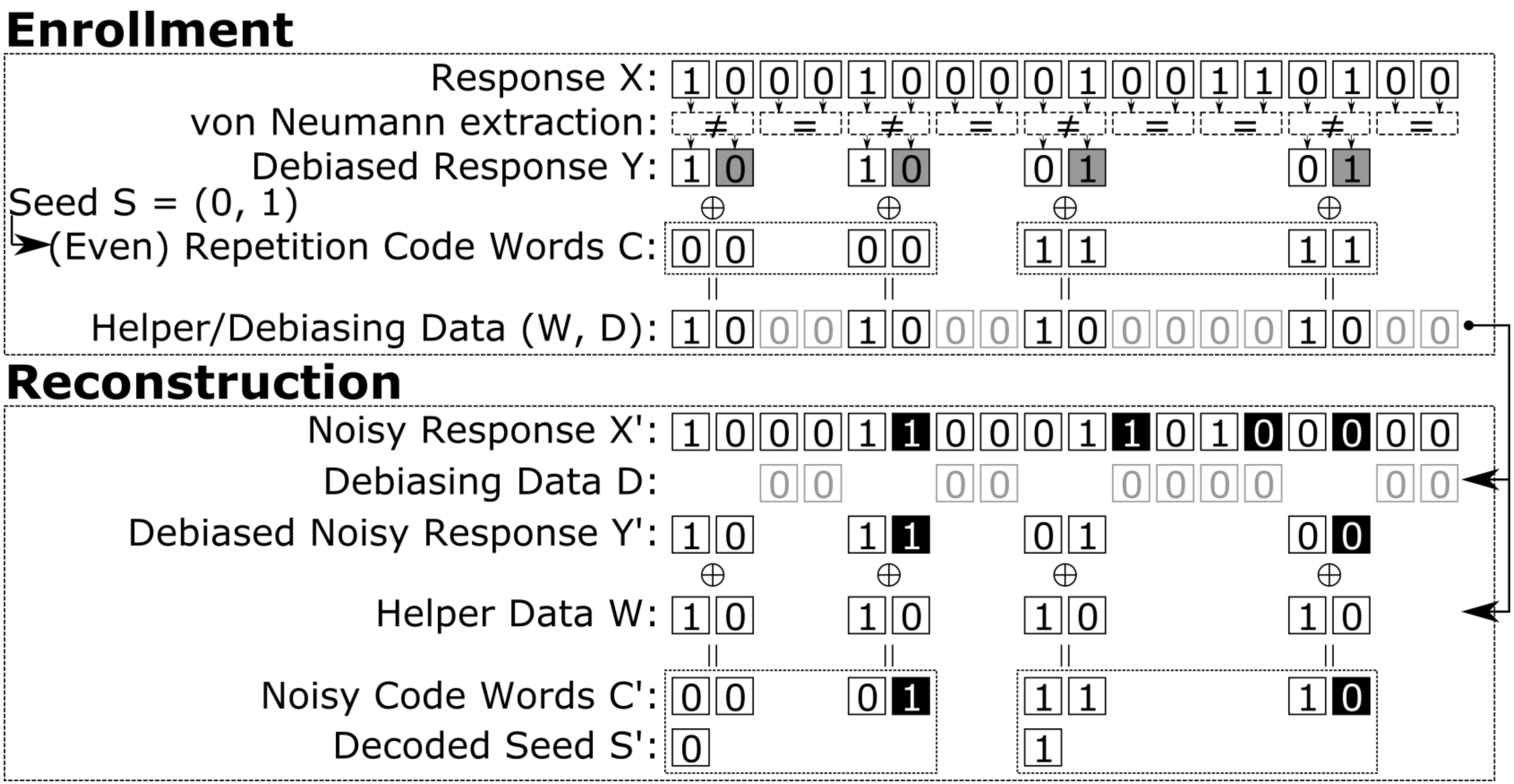
3. **Leakage:** $I(S\ ;\ (W, D)) = 0$

   No more leakage, regardless of level of bias! (proof in full version)

4. **Reusability:** Not reusable! Due to stochastic nature caused by bit errors



Debiasing Variant 2: Pair-Output Von Neumann



- Same as classic V.N., but :
  - Retain full pairs instead of only first bit
  - Inner code is even-length repetition code

Debiasing Variant 2: Pair-Output Von Neumann

1. **Reliability:** Hardly affected (same as classic V.N.)
2. **Efficiency:** Improvement w.r.t classic V.N. with factor ~2: debiasing overhead factor > 2

   Function of bias and reliability, e.g.: bias = 30% and |Y| = 1000 bits are needed with reliability > $1 - 10^{-6}$, then |X| needs to be ≥ 2794 → overhead factor 2.8
3. **Leakage:** $I(S\ ;\ (W, D)) = 0$

   No leakage! Regardless of level of bias! (proof in full version)

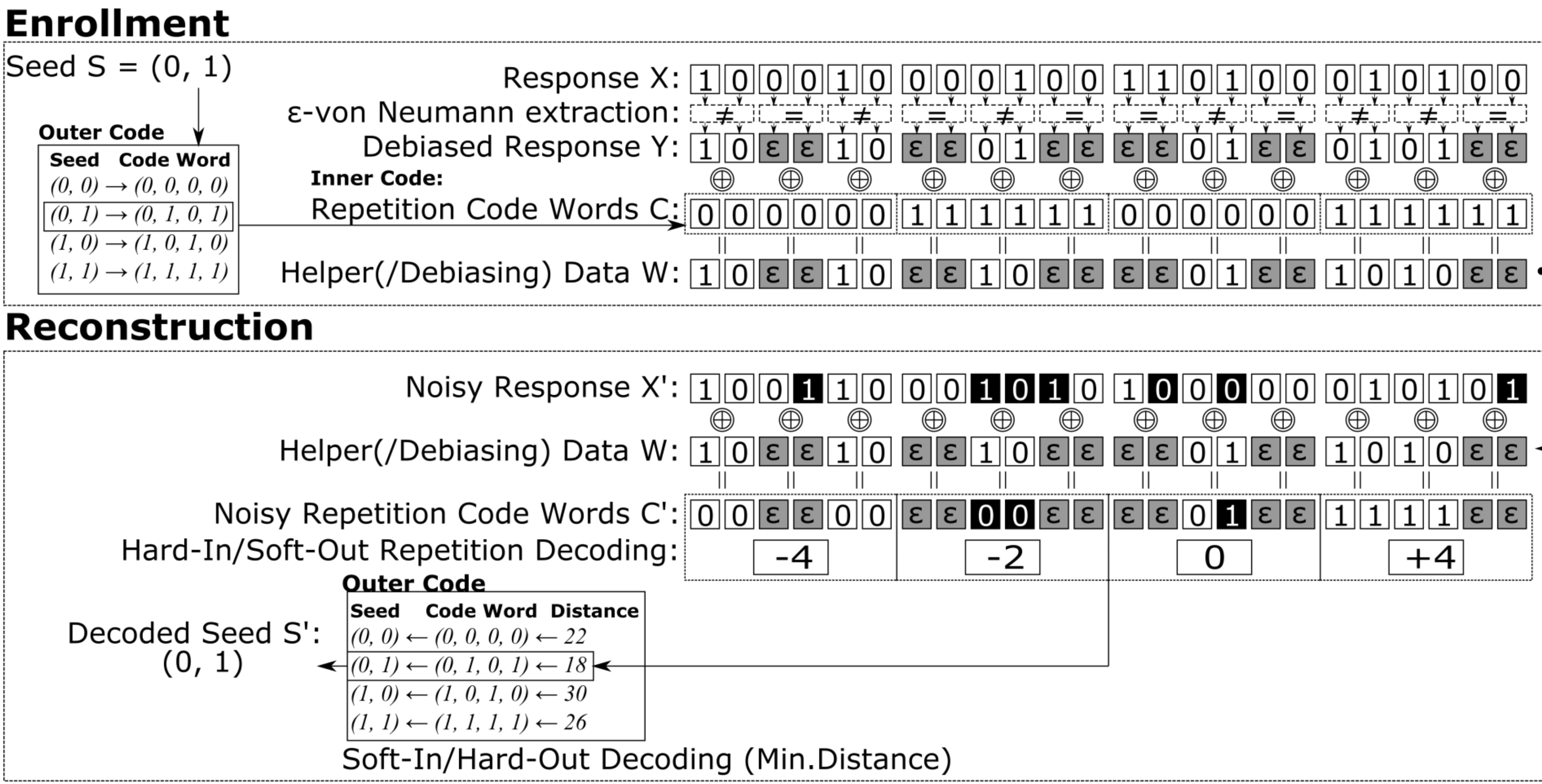
   Surprising given that Y has bit dependencies…

   Trick: Entropy loss due to bit dependencies coincides exactly with entropy loss of repetition code → no additional loss!
4. **Reusability:** Not reusable! (same as classic V.N.)

Variant 2+: *Multi-pass Tuple-Output Von Neumann*

- Reconsider discarded bits in a new pass, now considering quadruplets…
- Same properties, but further improved efficiency: overhead factor 1.5

Debiasing Variant 3: Erasure Von Neumann



- Same as pair-output V.N., but *erase* pairs i.s.o. discarding
  - Requires errors-and-erasures decoding at reconstruction

Debiasing Variant 3: Erasure Von Neumann

1. **Reliability:** Affected by introduction of erasures!

   Better code needed

2. **Efficiency:** No bits are discarded, but code rate is affected to deal with additional erasures

   Reliability and efficiency need to be considered together…

3. **Leakage:** $I(S\,;(W, D)) = I(S\,;W) = 0$

   No leakage! Regardless of level of bias! (proof in full version)

4. **Reusability:** Reusable! (proof in full version)

   Debiasing is no longer stochastic (not affected by eventual bit errors)

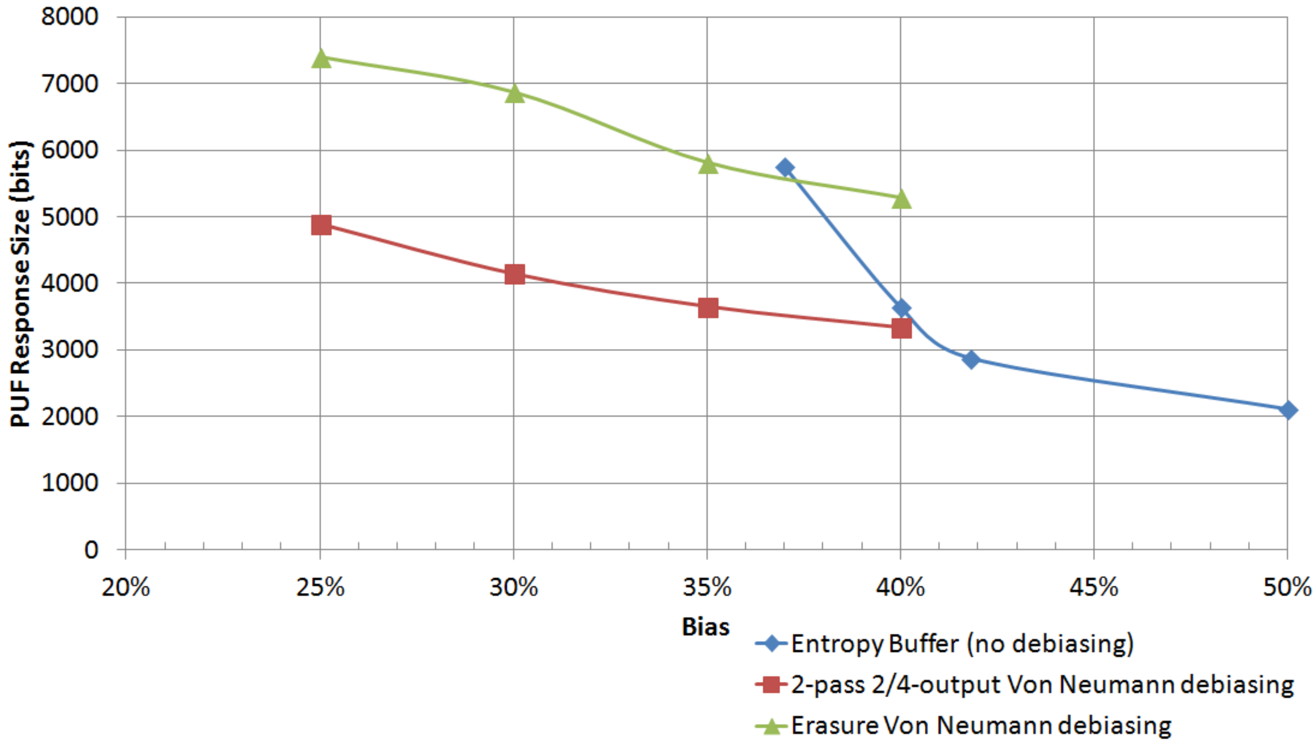
Variant 3+? No!

- As this will compromise reusability again

Comparison of Solutions

- Fair comparison:
  - Channel model: 0-bits and 1-bits have a different error rate
  - Error-rate and bias are related: e.g. 100% biased PUF must have error rate 0%
- Comparison based on repetition-Golay key generator:

  (128-bit key, 15% noise, 1-10^{-6} reliability)



Concluding Remarks

- PUF error rate and bias (unpredictability) are equally important and closely related metrics
- PUF bias might cause entropy leakage and affect security of key generator:
  - Earlier constructions are not always secure for biased PUFs
  - Entropy buffer solution works for small bias (close to 50%), but does not scale
- We proposed debiasing solutions based on Von Neumann:
  - No more entropy leakage, regardless of bias level!
  - Overhead cost can be reduced by clever optimizations (pair output, multi-pass)
  - Bias outside [40%-60%]: debiasing is better than entropy buffer
  - Maintaining reusability comes at a cost
- Future work:
  - Improve efficiency, in particular combined with reusability
  - Other leakage models (bit correlations, …)

Full version of the paper: https://eprint.iacr.org/2015/583.pdf