

Practical Attacks Against EALPN Using Belief Propagation

Pierre Galissant¹, Guilhem Jazeron¹, and Léo Perrin¹

INRIA, Paris, France,
`first-name.last-name@inria.fr`

Abstract. Weak and shallow pseudo random functions form a new class of symmetric primitives that is seeing a growing relevance, from their use in the initialization phase of some MPC protocols to the internals of some FHE-based protocols. However, their security is hard to ensure: “weak” means that chosen plaintext attacks are irrelevant, and “shallow” means that their construction cannot be round-based: in this exotic setting, how can we define secure PRFs?

In this paper, we investigate EALPN, a recent proposal of this type. We identify critical blind spots in the initial asymptotic analysis of the designers: the role of ℓ , one of its parameters, turns out to be crucial. Indeed, we present a key recovery attack with a complexity that is polynomial in all other parameters, and whose complexity grows exponentially but still slowly with ℓ . As a consequence, some parametrizations intended to provide 128 bits of security only offer 37 bits. Our attack is based on novel use of the belief propagation algorithm. Our analysis of its efficiency in our specific setting can be of independent interest.

We have implemented our attacks, and recover an equivalent key for an instance intended to be 128-bit secure in about 10 minutes on a desktop.

Keywords: EALPN, shallow weak PRF, cryptanalysis, belief propagation

1 Introduction

The design of a symmetric primitive, such as a block cipher, can now be dictated by trusted approaches: a round function is iterated multiple times, said round function being based on on a localized non-linear function (typically, a so-called *S-box*), a linear bijection of the full state, and (if relevant) a key addition. The goal is to prevent powerful attacks, in particular differential cryptanalysis [5], that assume a powerful adversary that is able to query the keyed primitive however they like, and as much as they like—provided that the total workload remains under the intended security level.

However, this entire ecosystem of attack techniques and counter-measures loses all relevance in an emerging setting of growing relevance: they are suited neither to the design, nor to the analysis of so-called *weak shallow Pseudo-Random Function* (ws-PRFs). These pseudo-random functions are defined by their relaxed security requirements (hence, “weak”), and the particularly harsh

upper bound on their multiplicative depth (thus, “shallow”). They are known to be required in a broadening set of applications, from the design of Oblivious PRFs [14] (e.g. in combination with TFHE [2]), to the initialization phase of some specific zero-knowledge protocols [1]. A more thorough introduction to the design space and potential use-cases of such functions can be found in [8].

How then can we design ws-PRFs that are both secure *and* efficient (for the relevant definition of “efficient”)? Round-based approaches are not applicable due to the constraints on multiplicative depth, and design techniques built to prevent differential attacks are irrelevant since such attacks do not work without chosen plaintexts. To solve this problem, multiple approaches have been proposed, such as the combination of arithmetic modulo 2 and 3 [6]. However, many variants of this primitive have been successfully analyzed, for instance in [3,18,11,17,4].

Our Contributions. In this paper, we show the limits of another design paradigm: that of EALPN. This ws-PRF was introduced by Boyle et al. in [9]. Informally, it takes as input a set of integers, is keyed by another set of integers, and (as argued in [8]) its output can be computed as the sum of Boolean predicates checking if a specific coordinate of the input is greater than a specific coordinate of the key. While it might not be obvious at first glance, this construction also has a clean description in terms of Expand-Accumulate codes, hence its name.

In what follows, we will exhibit attacks against EALPN of a nature that its designers did not foresee, and which as consequence have a devastating on some of the parameters that they suggested. Our best attack against a version intended to provide 128 bits of security with a query budget of 2^{45} samples requires 2^{41} operations and 2^{19} samples. In practice, this corresponds to a wall clock time of 10 minutes on a desktop (Intel Xeon w3-2423) with 32GB of RAM memory, of which we use around 22GB for the attack. Note also that our attack is not a mere distinguisher: it recovers an equivalent of the master key that yields an EALPN instance equal to the targeted one with probability at least 0.98. The code to replicate our results is available at:

https://github.com/314gali/EALPN_BP.

This breakthrough was enabled by the creative use of a *Belief Propagation* algorithm [15]. While these have seldom been used in symmetric cryptography (if at all), we show that they are particularly suited to the analysis of this primitive. We consider the addition of such techniques to the symmetric cryptanalysis toolbox to be a significant contribution as well: the designers of future ws-PRFs will have to take this attack vector into account.

Attack Overview. Instead of recovering the exact key used in the instance under scrutiny, our attack recovers a *quasi-equivalent* key: the corresponding key yields an EALPN instance that returns the same output with very high probability (above 0.98). This is achieved by considering an alternative construction,

built from the available observations of EALPN, and which is functionally equal to the observed instance up to some probability. This construction relies on a massive reduction of the size of the domain of both inputs and keys, which then allows to efficiently search for the “approximation” of the true keys in this much smaller space. The noise induced by these approximations (we do not recover the exact key, and the reduction implies errors) is of a structured, low-probability nature, and we describe how to get rid of it in practice by using a belief propagation algorithm on a sufficient amount of known input/output pairs. We also investigate how the low probability of these errors allows simulating the EALPN instance target with high probability, using only the knowledge of the positions of the key in the “reduced” space. Indeed, for some EALPN variants, this quantity turns out to be surprisingly low.

Outline. We start in Section 2 with a detailed description of EALPN, its alternative representations, and the state-of-the-art of its cryptanalysis. The necessary background on belief propagation is then recalled in Section 3.1, along with the other tools that we rely on for our attack. We set up the core concepts behind our attack, and we present its principle in Section 4. Then, we study its complexity (both asymptotically and experimentally) in Section 6, before concluding this paper in Section 7.

2 EALPN: Description and State-of-the-Art

We first need to lay the foundations of our work. First, we summarize our notations in Section 2.1, and we recall the mathematical description of the security model of weak PRFs in Section 2.2. We then describe EALPN, its parameters, and its alternative representations in Section 2.3: our attacks are best described in the alternative representation of [8]. Finally, while EALPN has not received as much analysis as e.g. the functions in the mod 2/mod 3 family, the authors of [16] did find a powerful distinguisher against this primitive, against which we will compare ourselves. We present this attack in Section 2.4, but we will already introduce a variant of EALPN that they introduce to prevent this attack in Section 2.3. We also provide an alternative representation of their fixed variant, which may be of independent interest.

2.1 Notations

We introduce here some notations that will be used in what follows:

- For $x \in \mathbb{R}$, the *rounding* of x , that is the closest integer to x , is denoted $[x]$. Similarly, $\lceil x \rceil$ designates the *ceiling* of x , that is the closest integer *above* x .
- For $i \leq j \in \mathbb{Z}$, $\{i, \dots, j\}$ represents the *integer interval* of integers between i and j , both ends included, that is the set $\{k \in \mathbb{Z}, i \leq k \leq j\}$.
- The finite field with two elements, that is the set $\{0, 1\}$ with binary operation XOR and AND (denoted as $\oplus$ and $\cdot$ respectively), is denoted $\mathbb{F}_2$.

- For $v \in \mathbb{F}_2^n$ a binary vector, and for $0 \leq i \leq j \leq n - 1$, $v[i]$ represents the i -th element of v while $v[i : j]$ is the subvector of v starting at index i and ending at index $j - 1$.
- The symbol $\propto$ means “is proportional to” when computing probabilities.

2.2 wPRF Security.

From the point of view of complexity theory, a wPRF is a family of keyed functions such that the input-output behaviour on *known* inputs is indistinguishable from a family of random functions. However, for the cryptanalysis of concrete instances, it is natural to work with *computational indistinguishability*. As a consequence, we work with the definition of [8].

Definition 1 (wPRF security [8]). *Let n, m, k and q be positive integers. A family of functions $F_K : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$, indexed by a key $K \in \mathbb{F}_2^k$, is said to achieve (q, T, ε) -weak PRF security if, for every distinguisher Dist running in time at most T and making at most q queries on inputs $x_1, \dots, x_q \in \mathbb{F}_2^n$ drawn independently and uniformly at random, we have:*

$$\left| \Pr \left[\text{Dist}(\{(x_i, F_K(x_i))\}_{1 \leq i \leq q}) = 1 \right] - \Pr \left[\text{Dist}(\{(x_i, \mathcal{R}(x_i))\}_{1 \leq i \leq q}) = 1 \right] \right| \leq \varepsilon,$$

where the key K is sampled uniformly at random from $\mathbb{F}_2^k$, and $\mathcal{R}$ is a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ sampled from a uniform distribution.

For this definition to be useful, the time complexity bound is set to $T \approx 2^\lambda$, where λ is the intended security level, and the goal of the attacker is to maximize $\varepsilon \leq 1$ (see [8]). One of the most important parameter is the number of queries q : it severely restricts potential attacks against the wPRF. As we will see in Section 2.3, the number of queries q that is allowed for the variants of EALPN we consider is significantly way smaller than the time T . Thus, the rare events needed for usual cryptanalysis techniques, such as a specific input difference for differential cryptanalysis, happen at best with a very small probability.

2.3 EALPN

EALPN is a ws-PRF proposed by Boyle et al. in [9]. We start by succinctly presenting its original code-based construction so as to describe its underlying code. Then, we introduce the alternative, equivalent description of [8] on which the rest of this paper is based.

The Code Construction. Let N, t, ℓ be integers such that t divides $5N$ and let $\omega = 5N/t$. EALPN then operates as follows.

1. Pick a matrix $H \in \mathbb{F}_2^{N \times 5N}$ randomly such that each row has Hamming weight ℓ , and consider the lower triangular matrix $\Delta_{5N} \in \mathbb{F}_2^{5N \times 5N}$ that is filled with 1s on and under the diagonal. The parity check matrix of EALPN is defined as $M = H \times \Delta_{5N}$.

2. Draw a random noise vector e such that it has a single 1 per slice of size t . It is the binary encoding of a vector of integer picked from the cartesian product $\{1, 2, 4, \dots, 2^{w-1}\}^t$.

The EALPN assumption states the hardness of distinguishing the distributions $(H, H \times \Delta_{5N} \cdot e)$ and (H, y) , where y is drawn uniformly in $\mathbb{F}_2^N$.

The Comparison-Based Representation. The authors of [8] presented a different but equivalent representation of this function. Remark that the *accumulator* Δ_{5N} transforms the vector e into a vector $\Delta_{5N} \cdot e$ that is composed of alternating stripes of 1 and bands of 0. The key observation is to introduce the predicate *greater-than* defined as follows over $\{1, \dots, \omega\}^2$:

$$\text{GT}(a, b) = \begin{cases} 1 & \text{if } a > b \\ 0 & \text{otherwise} \end{cases}.$$

Now, for each of the odd blocks of $f = \Delta_{5N} \cdot e$, cut it into t blocks of size ω and flip each odd-indexed block from little-endian to big-endian. That way, we get a vector $\tilde{f}$ that is composed t blocks of size ω starting with zeroes and followed by ones until the end of the block. For consistency, apply the same transformation to H to get $\tilde{H}$, such that $H \cdot f = \tilde{H} \cdot \tilde{f}$. Then, for any line of the matrix $\tilde{H}$, if there is a one in the τ_i -th block of the line at position x_i and the first one in the τ_i -th block of $\tilde{f}$ is at position K_{τ_i} , then, the product can be written: as the binary sum $\bigoplus_{i=1}^{\ell} \text{GT}(x_i, K_{\tau_i})$. This sum only contains ℓ terms since each line of H has Hamming weight ℓ . We can then state the definition that we will use for the rest of the paper.

Definition 2 (EALPN (representation from [8])). Let N, t, ℓ be integers such that t divides $5N$ and let $\omega = 5N/t$. The instance of EALPN with key $K = (K_1, \dots, K_t) \in \{1, \dots, \omega\}^t$ is defined for any input $(x, \tau) \in \{1, \dots, \omega\}^\ell \times \llbracket 1, t \rrbracket^\ell$ with $\forall i, j \in \{1, \dots, t\}, \tau_i \neq \tau_j$ by:

$$\text{EALPN}_K(x, \tau) = \bigoplus_{i=1}^{\ell} \text{GT}(x_i, K_{\tau_i}) . \quad (1)$$

Target Parameters. To have a concrete instance for cryptanalysis, we follow the updated parameters of in [21]. For the level of security $\lambda = 128$, the authors propose *Aggressive* and a *Standard* instance of EALPN:

- Aggressive: $N = q = 2^{45}$, $t = 660$, $\ell = 11$, $\omega = 2^{37.85}$
- Standard: $N = q = 2^{45}$, $t = 85$, $\ell = 3 \log(5N)$, $\omega = 2^{40.91}$

Following the recommendations of the authors of [8], we will focus on attacks for which $\epsilon \approx 1$ and $T \approx 2^\lambda$. The instance on which we propose an attack is the *Aggressive* instance.

2.4 State-of-the-art of the Cryptanalysis

Attacks in the Linear Test Framework The model in which the authors of EALPN argue for security is the linear test framework. This model is particularly relevant as it encapsulates many traditional attacks against codes constructions such as information set decoding, generalized birthday attacks and the BKW algorithm. In this model, as recalled in [8], an attack against EALPN would go as follows.

1. The attacker is given the parity-check matrix $M = H \cdot \Delta_{5N}$ and computes a vector v from the coefficients of the matrix.
2. A noise vector e is sampled and the attacker wins if the scalar product $v^\top \cdot M \cdot e$ is significantly biased. More precisely, a security level λ , requires a bias equal to $2^{-\lambda}$.

The key argument of the authors of [9] against linear attacks is to argue that a code with a large Hamming distance is not vulnerable to such attacks and that the code of EALPN has a large Hamming distance with high probability. However, the work of Raghuraman et al. [21] shows that the minimum distances assumed by the authors of EALPN are overestimated, which lead to an update to the parameters we are using in this paper. For instance the parameter l in the *aggressive* instance was increased from 7 to 11.

Attack from Dory’s authors and countermeasure In [16], the authors present a distinguishing attack against the original EALPN with parameters $l = 11$ and $t = 660$, and propose a countermeasure that prevents it.

The distinguishing attack. Re-writing it in the “greater-than” representation, there attack works as follows.

1. For all $j \in \{1, \dots, t\}$, replace the key K_j with $\tilde{K}_j = \frac{\omega}{2}$.
2. For each output $Y^{(i)}$, compute the quantity $Z^{(i)} = Y^{(i)} \oplus \bigoplus_{j=1}^l \text{GT}(X_j^{(i)}, \tilde{K}_{\tau_j^{(i)}})$,
which is then equal to $Y^{(i)} \oplus \bigoplus_{j=1}^l \text{GT}(X_j^{(i)}, \frac{\omega}{2})$
3. The event $Z^{(i)} = 0$ is the same as $Y^{(i)} = \bigoplus_{j=1}^l \text{GT}(X_j^{(i)}, \frac{\omega}{2})$, which in turn is the event that $\bigoplus_{j=1}^l \left(\text{GT}(X_j^{(i)}, \frac{\omega}{2}) \oplus \text{GT}(X_j^{(i)}, K_{\tau_j^{(i)}}) \right) = 0$
4. By a reasoning similar to the one conducted in the proof of Lemma 1 $\text{GT}(X_j^{(i)}, \frac{\omega}{2}) \oplus \text{GT}(X_j^{(i)}, K_{\tau_j^{(i)}})$ has bias $\frac{1}{2} + \frac{1}{\omega}$ and then, by the piling-up lemma, $Z^{(i)}$ will have bias $\varepsilon = (\frac{1}{2} + \frac{1}{\omega})^l \geq \frac{1}{2^l}$.
5. Since the number N of observable samples is much greater than $2^{2\ell} = 2^{22} > \varepsilon^{-2}$, the distinguisher simply collects enough samples, and outputs “EA-LPN” if $(Z^i)_{i \in \{1, \dots, N\}}$ is biased towards zero, otherwise it outputs “random”. The time complexity of this approach is then at least of $2^{22} \cdot \ell \approx 2^{26.5}$ (for each sample, ℓ XOR operations are needed).

In [16], the authors claim that their attack can also recover the key in time $2^\ell \cdot N \cdot 2\ell \approx 2^{60.46}$. However, a careful reading confirmed by the authors of this paper indicates that the algorithm that is actually needed is more complex than what they first considered.

The countermeasure. The weakness of the original EALPN identified in [16] stems from the fact that each evaluation of EALPN only involves l secret values. A natural counter-measure was then suggested in the same paper. For each slice $e[i\omega : (i+1)\omega]$ of size ω , uniformly draw a bit $\beta_i \in \mathbb{F}_2$, and then:

- if $\beta_i = 0$, then the whole slice is set to be 0;
- otherwise, if $\beta_i = 1$, draw a position j_0 uniformly from $\{0, \dots, \omega - 1\}$, and set $e[i \cdot \omega + j_0] = 1$ (leaving the rest of the slice to be 0), like in the original EALPN proposal.

The parameter t is also doubled, so that the hamming weight of e stays in average equal to t . Much like we recalled in Section 2.3, this can also be rephrased in terms of sums of comparisons.

Indeed, the *accumulator* Δ_{5N} still transforms the vector e into a vector $f = \Delta_{5N} \cdot e$ that is composed of alternating stripes of 1 and stripes of 0. The difference compared to before is that, due to the β_i s, we no longer know that all even blocks start with a zero, and all odd blocks with a one. We must then proceed as follows.

1. Construct a non-relaxed key vector $\tilde{e}$ as follows. For each slice i of length ω :
 - If $\beta_i = 0$, then set $\tilde{e}[i \cdot \omega] = 1$, while the rest of the coordinates of the slice are kept to zero.
 - Otherwise, if $\beta_i = 1$, then set $\tilde{e}[i \cdot \omega : (i+1) \cdot \omega] = e[i \cdot \omega : (i+1) \cdot \omega]$
2. As before, this naturally introduces to a notion of keys: let $j_i \in \{0, \dots, \omega - 1\}$ be the position where the 1 lies within the i -th slice. Then, set $K_i = j_i + 1 \in \{1, \dots, \omega\}$. Notice that, for all $1 \leq i \leq t$, the slice $e[i \cdot \omega : (i+1) \cdot \omega]$ is equal to $\beta_i \cdot \tilde{e}[i \cdot \omega : (i+1) \cdot \omega]$. Denote as $\beta \star \tilde{e}$ the vector whose coordinates are the $\beta_i \cdot \tilde{e}[i \cdot \omega : (i+1) \cdot \omega]$.
3. Compute $f = \Delta_{5N} \cdot e$:

$$\begin{aligned} \Delta_{5N} \cdot e[\tau \cdot \omega + j] &= (\Delta_{5N} \cdot (\beta \star \tilde{e}))[\tau \cdot \omega + j] \\ &= \left(\bigoplus_{i=1}^{\tau-1} \beta_i \cdot \bigoplus_{l=0}^{\omega-1} \tilde{e}[i \cdot \omega + l] \right) \oplus \left(\beta_\tau \cdot \bigoplus_{l=0}^j \tilde{e}[\tau \cdot \omega + l] \right) \\ &= \left(\bigoplus_{i=1}^{\tau-1} \beta_i \right) \oplus \beta_\tau \cdot \text{GT}(j, K_\tau) . \end{aligned}$$

The sum $\bigoplus_{l=0}^{\omega-1} \tilde{e}[i \cdot \omega + l]$ is always equal to one since exactly one coordinate of $\tilde{e}$ is set to one in each slice.

4. Finally, we can express the output of **hardened**–EALPN: for any line of the matrix H , there are l slices having a one. If there is a one in the $\tau^{(i)}$ -th slice, and the one lies at position x_i , then the output produced by this line of H is

$$\text{hardened-EALPN}_{K,\beta}(x)(\tau) = \bigoplus_{i=1}^l \beta_{\tau_i} \cdot \text{GT}(x_i, K_{\tau_i}) \oplus c_{\tau_i} ,$$

where, for all $1 \leq \tau \leq t$, we have $c_\tau = \bigoplus_{j=1}^{\tau-1} \beta_j$.

Belief Propagation in symmetric cryptanalysis. In symmetric cryptanalysis, belief propagation was already used to implement *fast correlation attacks* on LFSR-based stream ciphers [19,10,12]. In this context, the keystream is modelled as the output of the LFSR seen through a binary symmetric channel, from which low-weight parity check equations are derived. An attack is built by first collecting these equations, and attempting then running belief propagation to decode a low-density parity-check code.

3 Belief Propagation and other Tools

3.1 Overview of the Belief Propagation Algorithm

Belief Propagation (BP) is an algorithm formalised by Pearl in 1988 [20] to solve inference problems in Bayesian networks. It takes as input a list of binary equations which are sums of functions evaluated on unknown inputs, and returns a solution that satisfies most of the equations. Intuitively, it looks locally at what we believe about each variable in light of the relevant equations, and what we believe about each equation in light of what we know of variable. Then, the algorithm iteratively updates these beliefs until a satisfactory solution is found. It was later recognised that the iterative decoder introduced earlier by Gallager [15] for Low Density Parity Check (LDPC) codes is an instance of BP, which has since become a staple decoder. As our technique is a variation of decoding LDPC codes using BP, we give a short overview of how it works.

General Principle. Let $(c_i)_{i \in \{1, \dots, m\}}$ be a set of equations of the form $Y^{(i)} = \bigoplus_{j \in \mathcal{NC}(i)} g_j^{(i)}(x_j)$, where each $g_j^{(i)}$ is a (possibly nonlinear) function with probabilistic output from some domain $\mathcal{A}$ to $\mathbb{F}_2$ which constrains the unknown variable x_j , and $Y^{(i)} \in \mathbb{F}_2$ are constants. Using the Tanner graph representation [23], we can build a bipartite graph with vertices $(c_1, \dots, c_m)$ representing the equations and vertices $(v_1, \dots, v_n)$ representing the unknown variables $(x_1, \dots, x_n)$. The vertices $(c_i)_{i \in \{1, \dots, m\}}$ will be called *check nodes*, while the vertices $(v_j)_{j \in \{1, \dots, n\}}$ will be called *variable nodes*. Then, there is an edge (v_j, c_i) if and only if j belongs to $\mathcal{NC}(i)$, that is if x_j appears constrained in equation c_i . We will call $\mathcal{NC}(i)$ the set of *neighbors* of check node c_i , and similarly, we will define for a variable node v_j , $\mathcal{NV}(j)$ the set of its *neighbors*, composed of all the check nodes to which v_j is connected in the Tanner Graph.

We will also denote as d_v and d_c the *average degrees* of variable nodes and check nodes, which are formally defined as the average value of $(|\mathcal{NV}(j)|)_{j \in \{1, \dots, n\}}$ and $(|\mathcal{NC}(i)|)_{i \in \{1, \dots, m\}}$. These quantities will be of particular interest in the complexity analysis of Section 6.

Remark 1. In the case of LDPC codes, the constraints are given by the *parity check matrix* H of the code, and the equations are given by the inner products between rows and the syndrome. Indeed, if we denote the i -th row of $H \in \mathbb{F}_2^{m \times n}$

as $H^{(i)} \in \mathbb{F}_2^n$, then equation c_i is:

$$\bigoplus_{j \in \{1, \dots, n\}, H_j^{(i)} \neq 0} x_j = 0 ,$$

so that we have both $\mathcal{NC}(i) = \{j \in \{1, \dots, n\}, H_j^{(i)} \neq 0\}$ for all $\forall i \in \{1, \dots, m\}$, and $g_j^{(i)}(x_j) = x_j$ for all $j \in \mathcal{NC}(i)$.

Example 1. Figure 1 shows an example of how the Tanner Graph associated to a matrix is constructed. In this specific example, $\mathcal{NC}(1) = \{1, 2, 4\}$, and $\mathcal{NV}(1) = \{1, 3\}$.

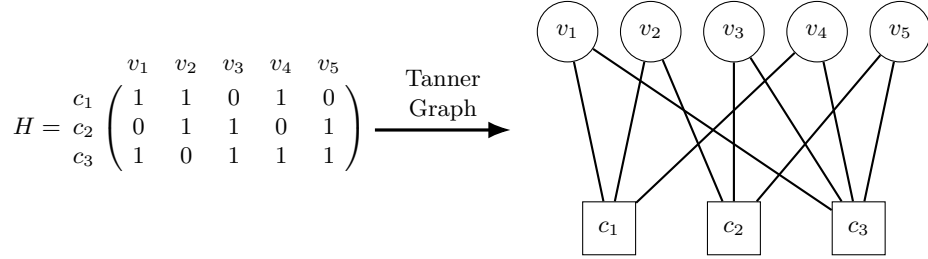


Fig. 1: Constructing the Tanner Graph associated to a binary matrix H .

We assume that each variable v_j was drawn uniformly in the set $\mathcal{A}$ and the equations $c_1, \dots, c_m$ serve as posterior information that we will use to statistically pin down the value of $x = (x_1, \dots, x_n)$. We will thus be starting from an *a priori* distribution on the values $x_j \in \mathcal{A}$, and the goal of the Belief Propagation algorithm is to compute the *a posteriori* distribution of each v_j conditioned on the random event C defined as: " $\forall i \in \{1, \dots, m\}, Y^{(i)} = \bigoplus_{j \in \mathcal{NC}(i)} g_j^{(i)}(x_j)$ ". We will also denote as C_i for $i \in \{1, \dots, m\}$ the random event stating that equation c_i holds, such that $C = \bigcap_{i=1}^m C_i$. Then, in order to compute the *a posteriori distribution*, the algorithm iteratively sends messages carrying its current *belief* from variable nodes to their neighboring check nodes in the Tanner graph, and reciprocally from check nodes to variable nodes. These messages carry only *extrinsic* information, that is, information that depends only on nodes in the graph which are not recipients of the message currently being computed.

3.2 Derivation of Formulas for BP.

More formally, we want to compute the posterior marginals $b_k(a) = \Pr(x_k = a \mid C)$ for $a \in \mathcal{A}$. That is, we compute the *belief* that the value of variable x_k is a knowing the constraints forced by equations $c_1, \dots, c_m$ on the vector

$x = (x_1, \dots, x_n)$. By the formula of total probabilities, we have:

$$b_k(a) = \sum_{z \in \mathcal{A}^n, z_k=a} \Pr(x = z \mid C) .$$

Then, since the equations c_i s are conditionally independent given v , and since each c_i only depends on $v_{\mathcal{NC}(i)} = (v_j)_{j \in \mathcal{NC}(i)}$, we have:

$$\Pr(C \mid v = z) = \prod_{i=1}^n q_i(z_{\mathcal{NC}(i)}) ,$$

where $q_i(z_{\mathcal{NC}(i)}) := \Pr(C_i \mid x_j = z_j, j \in \mathcal{NC}(i))$ is the likelihood of the observation C_i under the knowledge of the value of its variable neighbors. Furthermore, the probability that event C happens is a constant. Thus, Bayes law $\Pr(x = z \mid C) = \Pr(C \mid x = z)\Pr(x = z)/\Pr(C)$ implies that

$$\Pr(x = z \mid C) \propto \Pr(C \mid x = z)\Pr(x = z) = \prod_{i=1}^m q_i(z_{\mathcal{NC}(i)}) \prod_{j=1}^n \Pr(x_j = z_j) .$$

In the end, this yields that:

$$b_k(a) \propto \sum_{z \in \mathcal{A}^n, z_k=a} \prod_{i=1}^m q_i(z_{\mathcal{NC}(i)}) \prod_{r=1}^n \Pr(x_r = z_r) .$$

If we were to compute this sum as is, it would require $|\mathcal{A}|^{n-1}$ terms, each needing the computation two products, over m and n terms, which is prohibitive. Instead, the Belief Propagation algorithm relies on the following elementary distributivity remark: whenever two functions f and g depend on *disjoint* sets of variables A and B , the sum of their product factorizes into a product of sums. Writing $z_S := (z_r)_{r \in S}$ for the restriction of an assignment to a set of indices S , this can be written:

$$\sum_{z_{A \cup B} \in \mathbb{F}_2^{|A|+|B|}} f(z_A) g(z_B) = \left(\sum_{z_A \in \mathbb{F}_2^{|A|}} f(z_A) \right) \left(\sum_{z_B \in \mathbb{F}_2^{|B|}} g(z_B) \right) . \quad (2)$$

The BP algorithm assumes that each variable is only locally impacted by its neighbors, so that repeatedly applying Equation (2) inside the neighborhood of each check node yields the local approximation that $b_k(a)$ is proportional to:

$$p_k(a) \prod_{i \in \mathcal{NV}(k)} \left(\sum_{\substack{z_{\mathcal{NC}(i) \setminus \{k\}} \in \mathcal{A}^{\mathcal{NC}(i) \setminus \{k\}}}} q_i(z_{\mathcal{NC}(i) \setminus \{k\}} \cup \{x_k = a\}) \prod_{r \in \mathcal{NC}(i) \setminus \{k\}} \Pr(x_r = z_r) \right) . \quad (3)$$

Here $q_i(z_{\mathcal{NC}(i) \setminus \{k\}} \cup \{x_k = a\})$ is $\Pr(C_i \mid x_k = a \text{ and } \forall r \in \mathcal{NC}(i) \setminus \{k\}, x_r = z_r)$ and $p_k(a)$ is the *a priori* probability that $x_k = a$. While Equation (3) is an approximation for most graphs, it is exact for trees.

Computing Messages. We can now describe the messages $\mu_{v_j \rightarrow c_i}^{(r)}$ sent from variables to check nodes at round r of the algorithm and $\mu_{c_i \rightarrow v_k}^{(r)}$ sent from check to variable nodes at round t . Each carries the belief of its source, restricted to the corresponding factor of Equation (3) and stripped of the recipient's own contribution (*extrinsic* information).

Variable to check nodes. Starting from the *a priori* distribution on the v_j , the initial message from v_j to c_i is $\mu_{v_j \rightarrow c_i}^{(0)}(a) = p_j(a)$, the *a priori* probability that variable j has value a . For $t \geq 1$, it is the product of the incoming messages from the *other* neighboring check nodes with the *a priori* probability:

$$\mu_{v_j \rightarrow c_i}^{(t)}(a) \propto p_j(a) \prod_{i' \in \mathcal{NC}(j) \setminus \{i\}} \mu_{c_{i'} \rightarrow v_j}^{(t-1)}(a), \quad a \in \mathbb{F}_2. \quad (4)$$

Check to variable nodes. The message $\mu_{c_i \rightarrow v_k}^{(t)}$ gathers, from c_i , the extrinsic belief on x_k implied by the parity constraint of c_i together with the current messages of the *other* variables of $\mathcal{NC}(i)$. Applying Equation (2) to the corresponding factor of Equation (3), and letting $a \in \mathbb{F}_2$, we get that $\mu_{c_i \rightarrow v_k}^{(t)}(a)$ is proportional to:

$$\sum_{z_{\mathcal{NC}(i) \setminus \{k\}} \in \mathcal{A}^{\mathcal{NC}(i) \setminus \{k\}}} q_i(z_{\mathcal{NC}(i) \setminus \{k\}} \cup \{x_k = a\}) \prod_{r \in \mathcal{NC}(i) \setminus \{k\}} \mu_{v_r \rightarrow c_i}^{(t)}(z_r). \quad (5)$$

Finally, the belief at a variable node is obtained by combining its own likelihood p_k with all its incoming check messages:

$$b_k(a) \propto p_k(a) \prod_{i \in \mathcal{NV}(k)} \mu_{c_i \rightarrow v_k}^{(t)}(a), \quad a \in \mathcal{A}. \quad (6)$$

Note that Equations (4) and (5) are only exact for trees, and good approximations otherwise. See [13] for more details on the BP algorithm.

Recovering the unknowns. After multiple rounds of message passing, one has to make a “hard” decision from the distribution of belief $b_k(a)$. In what follows, we take this decision as the argmax of the belief. We also decide that the BP algorithm stops after n_r rounds or that the distribution of beliefs is stagnant up to a precision δ :

$$\max_{a \in \mathcal{A}} |b_k^{(r)}(a) - b_k^{(r-1)}(a)| \leq \delta$$

3.3 Computing messages in practice.

In practice, while doing the local approximation described in Equation (3) allows to go from products with n and m terms to products with on average d_v and d_c terms, and the sum to go through $|\mathcal{A}|^{d_c}$ values instead of $|\mathcal{A}|^m$, looping through the $|\mathcal{A}|^{d_c}$ values remains prohibitive.

However, since $q_i(z_{\mathcal{NC}(i) \setminus \{k\}} \cup \{x_k = a\})$ expands to $\Pr(C_i \mid x_k = a \text{ and } \forall r \in \mathcal{NC}(i) \setminus \{k\}, x_r = z_r)$, one can then view the quantity:

$$\sum_{\substack{z_{\mathcal{NC}(i) \setminus \{k\}} \\ \in \mathcal{A}^{\mathcal{NC}(i) \setminus \{k\}}}} q_i(z_{\mathcal{NC}(i) \setminus \{k\}} \cup \{x_k = a\}) \prod_{r \in \mathcal{NC}(i) \setminus \{k\}} \mu_{v_r \rightarrow c_i}^{(t)}(z_r)$$

as the probability that event C_i is true where x_k is fixed to a and the laws for variables x_r for $r \in \mathcal{NC}(i) \setminus \{k\}$ are given by $\mu_{v_r \rightarrow c_i}^{(t)}(\cdot)$, the beliefs on them at iteration t of BP, collected by check node c_i , which we will denote for simplicity as:

$$\Pr(C_i \mid x_r = a, \text{ extrinsic information at node } c_i, \text{ round } t).$$

Now, since C_i is in turn the random event stating that $Y^{(i)} = \bigoplus_{j \in \mathcal{NC}(i)} g_j^{(i)}(x_j)$, this probability finally becomes:

$$\Pr\left(\bigoplus_{j \in \mathcal{NC}(i)} g_j^{(i)}(x_j) = Y^{(i)} \mid x_r = a, \text{ extrinsic information at node } c_i, \text{ round } t\right)$$

which we can easily compute in reasonable time by studying the probability laws of the constraint functions under the knowledge of extrinsic information: $\Pr(g_j^{(i)}(x_j) = 0 \mid x_r = a, \text{ extrinsic information at node } c_i, \text{ round } t)$, and then applying the piling-up lemma, as each constraint function $g_j^{(i)}(x_j)$ is a random variable over $\mathbb{F}_2$ (this is exactly FFT in $\mathbb{F}_2$, also known as Walsh transform). This is exactly what we do in Section 4.3.

4 EALPN, Approximations and Modelling

4.1 Approximations of EALPN

The starting point of our cryptanalysis is the remark that EALPN with inputs and keys over $\{1, \dots, \omega\}$ can be approximated with a good precision by taking inputs over the “grid” $\{k \times \omega/\nu \mid k \in \{0, \dots, \nu\}\}$ of $Q = \nu + 1$ points.

Definition 3 (Index and Approximation). *Let $x \in \{1, \dots, \omega\}$ and $\nu \in \mathbb{N} \setminus \{0\}$. Let $\text{step} = \frac{\omega}{\nu}$ be the grid spacing, so that the Q grid points $\{k \cdot \text{step} \mid k \in \{0, \dots, \nu\}\}$ span the whole range $\{0, \dots, \omega - 1\}$. We then define two quantities:*

- the integer $\text{ind}_\nu(x) \in \{0, \dots, \nu\}$ corresponding to the index of the closest value to x on the grid.

$$\text{ind}_\nu(x) = \left\lfloor \frac{x}{\text{step}} \right\rfloor = \left\lfloor \frac{x \times \nu}{\omega} \right\rfloor ;$$

- the integer $\text{app}_\nu(x) \in \{k \times \text{step} \mid k \in \{0, \dots, \nu\}\}$ corresponding to the nearest approximation to x on the grid.

You may refer to Figure 2 to better understand these definitions.

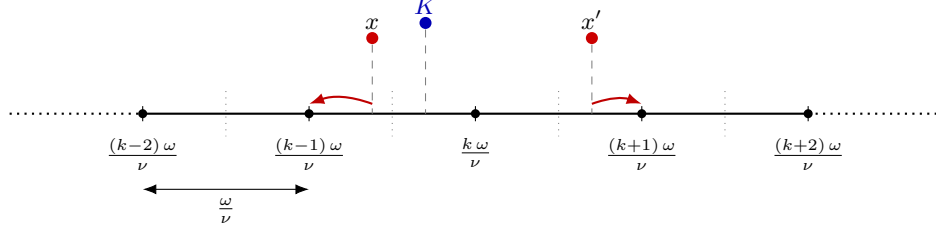


Fig. 2: Approximation of two values x and x' on the grid with $\text{ind}_\nu(x) = k - 1$, $\text{app}_\nu(x) = \frac{(k-1)\omega}{\nu}$ and $\text{ind}_\nu(x') = k + 1$, $\text{app}_\nu(x') = \frac{(k+1)\omega}{\nu}$.

Remark 2. The careful reader will have noticed that while inputs and keys are drawn from $\{1, \dots, \omega\}$, for the “grid” we take the set $\{k \cdot \text{step} \mid k \in \{0, \dots, \nu\}\}$. This is done simply as a way to ease the presentation, but in reality one should either use the interval $\{0, \dots, \omega'\} = \{0, \dots, \omega - 1\}$ (which is what we implicitly do in proofs) to draw values and key from (a translation of the same value for both inputs does not change the output of $\text{GT}(\cdot, \cdot)$), or add one to $\text{app}_\nu(x)$.

We stress that ν is a new parameter we introduce: it controls the fineness of the approximations the attacker will make. The higher ν , the more precise the approximations are. This is made formal below, in Lemma 1.

Lemma 1 (Approximation of GT). *Let $\nu \in \mathbb{N} \setminus \{0\}$, and let X and K be uniform random variables over $\{1, \dots, \omega\}$. We formally define the error probability p_{err} as $p_{\text{err}} = \Pr(\text{GT}(X, K) \neq \text{GT}(\text{app}_\nu(X), K))$. Then it holds that*

$$p_{\text{err}} = \frac{1}{4\nu} + \frac{1}{2\omega}.$$

Proof. For any $j \in \{0, \dots, 2\nu\}$, let I_j be the interval $\{j \times \text{step}/2, \dots, (j+1) \times \text{step}/2\}$. These intervals correspond to the half-intervals that compose the intervals $\{i \times \text{step}, \dots, (i+1) \times \text{step}\}$. The formula of total probabilities over the value of K yields:

$$p_{\text{err}} = \sum_{j=0}^{2\nu} \underbrace{\Pr(\text{GT}(X, K) \neq \text{GT}(\text{app}_\nu(X), K) \text{ and } K \in I_j)}_{p_{\text{err}}(I_j)}. \quad (7)$$

Then, as $\Pr(K = k) = \frac{1}{\omega}$ for any k , we have:

$$p_{\text{err}}(I_j) = \sum_{k \in I_j} \Pr(\text{GT}(X, K) \neq \text{GT}(\text{app}_\nu(X), K) | K = k) \times \frac{1}{\omega}$$

The probability $\Pr(\text{GT}(X, K) \neq \text{GT}(\text{app}_\nu(X), K) | K = k)$ is the same as $\Pr(\text{GT}(X, k) \neq \text{GT}(\text{app}_\nu(X), K))$. Thus, as $\text{app}_\nu(\cdot)$ corresponds to rounding, there is an approximation error for GT with key $k \in I_j$ only if X lies between k

and the other semi-interval bound. Informally, it is the situation where approximating X makes it “jump” from one side of k to the other. Whether this can happen depends on the direction of the rounding, as can be better seen in Figure 2. In any case, the length of the error interval is $|k - \frac{\text{step}}{2}|$. As a consequence, we have:

$$\sum_{k \in I_j} \Pr(\text{GT}(X, k) \neq \text{GT}(\text{app}_\nu(X), k)) = \sum_{k=0}^{\text{step}/2} \frac{|k - \text{step}/2|}{\omega} = \sum_{k=0}^{\text{step}/2} \frac{k}{\omega}.$$

Reincorporating this into Equation (7), we get:

$$p_{\text{err}} = \sum_{j=0}^{2\nu} \sum_{k=0}^{\text{step}/2} \frac{k}{\omega} \times \frac{1}{\omega} = \frac{1}{\omega^2} \sum_{j=0}^{2\nu} \frac{\text{step}/2(\text{step}/2 + 1)}{2} = \frac{1}{4\nu} + \frac{1}{2\omega}.$$

The previous lemma can also be interpreted as a description of the distribution of the following error. It is a view that will be used throughout the paper.

Definition 4 (GT Approximation Error). For $x, k \in \{1, \dots, \omega\}$, the approximation error of GT is

$$\text{Err}(x, k) = \begin{cases} 1 & \text{if } \text{GT}(x, k) \neq \text{GT}(\text{app}_\nu(x), k), \\ 0 & \text{otherwise.} \end{cases}$$

By definition of p_{err} , we have $p_{\text{err}} = \Pr(\text{Err}(X, K) = 1)$ for uniform X and K .

As we would intuitively expect, the bigger the ν (the finer the approximations), the lower this probability of error. For the aggressive version of EALPN, the value of ω is $5 \times 2^{45}/660 \approx 2^{39}$, and since the attacker-chosen ν will be kept under 100, it will hold that $\frac{1}{2\omega} \ll \frac{1}{4\nu}$. Thus, the error p_{err} can be approximated as $\frac{1}{4\nu}$. We can now describe our approximation of the whole PRF.

Proposition 1 (Approximation of EALPN). Let $X = (X_1, \dots, X_l)$ and $K = (K_1, \dots, K_l)$, where the X_i and the K_j are uniform and independent random variables over $\{1, \dots, \omega\}$. For any $\tau \in \llbracket 1, t \rrbracket^l$, we have:

$$\Pr(\text{EALPN}_K(X, \tau) \neq \text{EALPN}_K(\text{app}_\nu(X), \tau)) = \frac{1 - (1 - 2p_{\text{err}})^l}{2}. \quad (8)$$

We call $(x, \tau) \mapsto \text{EALPN}_K(\text{app}_\nu(x), \tau)$ a ν -approximation of EALPN_K .

Proof. $\text{EALPN}_K(x, \tau)$ and $\text{EALPN}_K(\text{app}_\nu(x), \tau)$ agree if and only if the sum $\bigoplus_{j=1}^l \text{Err}(X_j^{(i)}, K_{\tau_j^{(i)}})$ is equal to 0. We thus apply the piling-up lemma on $\text{Err}(\cdot, \cdot)$, whose law is given by p_{err} , to obtain the proposition.

The main consequence is that we can build a good approximation of EALPN by restricting its inputs to $\{0, \dots, \nu\}$, a much smaller set than the full $\{1, \dots, \omega\}$. For example, with $p_{\text{err}} \approx \frac{1}{400}$ (taking $\nu = 100$), recovering the 100-approximation would predict the output of the actual instance with probability equal to $1 - (1 - (1 - 2p_{\text{err}})^{11})/2 = 0.973$. Since the key K is a secret, the question now becomes:

How to compute $\text{EALPN}_K(\text{app}_\nu(x), \tau)$ without knowing K while respecting both the query and time budgets given by the designers' claims?

4.2 Modelling of the new problem

In order to find an efficient way to compute $\text{EALPN}_K(\text{app}_\nu(x), \tau)$, we will first restate the whole problem within our approximated framework. If we let $(X^{(i)}, Y^{(i)})$ be the i -th known input/output pair, then, for all i in $\{1, \dots, N\}$, we have

$$\begin{aligned} Y^{(i)} &= \underbrace{\bigoplus_{j=1}^l \text{GT}(X_j^{(i)}, K_{\tau_j^{(i)}})}_{\text{EALPN}_K(X^{(i)})} = \underbrace{\bigoplus_{j=1}^l \text{GT}(\text{app}_\nu(X_j^{(i)}), K_{\tau_j^{(i)}})}_{\text{EALPN}_K(\text{app}_\nu(X^{(i)}))} \oplus \text{Err}(X_j^{(i)}, K_{\tau_j^{(i)}}) \\ &= \bigoplus_{j=1}^l \text{GT}(\text{ind}_\nu(X_j^{(i)}) \cdot \text{step}, K_{\tau_j^{(i)}}) \oplus \text{Err}(X_j^{(i)}, K_{\tau_j^{(i)}}) \end{aligned}$$

This proves itself interesting because we now need only to focus on the *comparison coefficients* that we introduce below.

Definition 5 (Comparison coefficients). For $i \in \{0, \dots, \nu\}$ and $t \in \{1, \dots, t\}$, we denote $c_{i,j} \in \mathbb{F}_2$ and call comparison coefficient between K_j and the i -th element of the grid the quantity defined as $c_{i,j} = \text{GT}(i \cdot \text{step}, K_j)$.

Then, as we saw in Section 4.1, it is enough to recover the $\nu \times t$ comparison coefficients to construct an approximation of EALPN_K that is correct with an overwhelming probability, and is it much easier than focusing instead on recovering the true values of the keys (which can take ω^t values).

In fact, we can be more compact. Indeed, for a fixed key K_{j_0} the coefficients (c_{i,j_0}) are monotonous i.e. $(c_{i,j_0})_{i \in \{0, \dots, \nu\}} = (0, \dots, 0, 1, \dots, 1)$, because $a < b$ and $b < c$ implies $a < c$. Thus, it is enough to focus on recovering only the *threshold* where the switch happens.

Definition 6 (Thresholds of the key). Let $K = (K_1, \dots, K_t) \in \{1, \dots, \omega\}^t$ and $\nu \in \mathbb{N} \setminus \{0\}$. Let $\text{step} = \frac{\omega}{\nu}$ be the grid spacing. We define the threshold of the key word K_j as the integer $\text{tresh}_\nu(K_j) \in \{1, \dots, \nu\}$ such that:

$$c_{\text{tresh}_\nu(K_j)-1,j} = 0 \quad \text{and} \quad c_{\text{tresh}_\nu(K_j),j} = 1 .$$

Remark 3. Equivalently, the threshold of K_j is the unique integer $\alpha \in \{1, \dots, \nu\}$ such that $K_j \in \{(\alpha-1) \cdot \text{step}, \dots, \alpha \cdot \text{step} - 1\}$. It is also given by $\text{tresh}_\nu(K_j) = \lceil \frac{K_j}{\text{step}} \rceil$. This differs from the quantity $\text{ind}_\nu(K_j)$, which gives the index of the closest point of the grid, while $\text{tresh}_\nu(K_j)$ is the index of the point that is *right above* the key. The two quantities are related by $|\text{ind}_\nu(K_j) - \text{tresh}_\nu(K_j)| \leq 1$.

Remark 4. It is straightforward, knowing $\text{tresh}_\nu(K_j) = \alpha$, to compute the coefficients $c_{i,j}$. Indeed:

$$c_{i,j} = \begin{cases} 1 & \text{if } i \cdot \text{step} \geq \alpha \cdot \text{step} \\ 0 & \text{if } i \cdot \text{step} \leq (\alpha - 1) \cdot \text{step} . \end{cases}$$

Since $\text{step} > 0$, this means $c_{i,j} = \text{GT}(i, \text{tresh}_\nu(K_j) - 1)$.

Under this formulation an evaluation of EALPN_K becomes:

$$Y^{(i)} = \underbrace{\bigoplus_{j=1}^l \text{GT}(\text{ind}_\nu(X_j^{(i)}), \text{tresh}_\nu(K_{\tau_j^{(i)}})) \oplus \text{Err}(X_j^{(i)}, K_{\tau_j^{(i)}})}_{\text{EALPN}_K(\text{app}_\nu(X^{(i)}))} \quad (9)$$

Hence the goal of our attack is to recover the values $\text{tresh}_\nu(K_j)$, for all $j \in \{1, \dots, t\}$. At this stage, we have showed how to discretize the input x , how to discretize the secret key K , and how to build a quasi-equivalent function for EALPN given the discretized key. All that is left to do is recover said key.

Recovering Thresholds Using Belief Propagation. In order to recover the thresholds T_j , we adopt a statistic approach via the BP algorithm introduced in Section 3. Now that the notations are set for the modelling of the (approximated) problem we tackle, we can rephrase so as to fit the BP formulas from Section 3.2.

- The *variables* $x = (x_1, \dots, x_n)$ we will recover with BP are the thresholds $(T_1, \dots, T_t) = (\text{tresh}_\nu(K_1), \dots, \text{tresh}_\nu(K_t)) \in \{1, \dots, \nu\}^t$, hence the number of variables n is equal to t in our instance, and the alphabet $\mathcal{A}$ is $\{1, \dots, \nu\}$.
- The *equations* $c_1, \dots, c_m$ are given, for $i \in \{1, \dots, N\}$, by querying the PRF. They are of the shape $Y^{(i)} = \sum_{j=1}^l \text{GT}(X_j^{(i)}, K_{\tau_j^{(i)}})$. The number of equations m is then equal to N in our case.
- Equation (9) yields $Y^{(i)} = \bigoplus_{j=1}^l \text{GT}(\text{ind}_\nu(X_j^{(i)}), T_{\tau_j^{(i)}-1}) \oplus \text{Err}(X_j^{(i)}, K_{\tau_j^{(i)}})$, for all $i \in \{1, \dots, N\}$. Thus, the constraints functions $g_j^{(i)} : \mathcal{A} \rightarrow \mathbb{F}_2$ will be defined as: $g_j^{(i)} : x \mapsto \text{GT}(\text{ind}_\nu(X_j^{(i)}), x - 1) \oplus \text{Err}(X_j^{(i)}, K_{\tau_j^{(i)}})$. The probabilistic nature of their output is inherited from $\text{Err}(X_j^{(i)}, K_{\tau_j^{(i)}})$, given that the keys are modelled as random variables.

Then, following the presentation from Section 3.1, the goal of our attack is to compute, for all $j_0 \in \{1, \dots, t\}$ and $x \in \{1, \dots, \nu\}$ the probability that T_{j_0} is equal to x knowing the values $(Y^{(i)})_{i \in \{1, \dots, N\}}$, $(X_j^{(i)})_{i \in \{1, \dots, N\}, j \in \{1, \dots, l\}}$, and the N equations $Y^{(i)} = \bigoplus_{j=1}^l c_{\text{ind}_\nu(X_j^{(i)}), \tau_j^{(i)}} + \text{Err}(X_j^{(i)}, K_{\tau_j^{(i)}})$. Alternatively, this probability also corresponds the posterior laws of the key thresholds given N observations from an EALPN oracle.

4.3 Derivation of the probability formulas for Belief Propagation

Following 3.3, in order to apply the Belief Propagation algorithm on the problem described in the previous paragraph, we need to derive a closed formula for the

following quantities, for all $i \in \{1, \dots, N\}$, $j_0 \in \{1, \dots, l\}$ and $\alpha \in \{1, \dots, \nu\}$:

$$\Pr\left(Y^{(i)} \oplus c_{\text{ind}_\nu(X_{j_0}^{(i)}), \tau_{j_0}^{(i)}} \oplus \text{Err}(X_{j_0}^{(i)}, K_{\tau_{j_0}^{(i)}}) = \bigoplus_{j \neq j_0} c_{\text{ind}_\nu(X_j^{(i)}), \tau_j^{(i)}} \oplus \text{Err}(X_j^{(i)}, K_{\tau_j^{(i)}}) \mid T_{\tau_{j_0}} = \alpha\right)$$

In fact, for $j \in \{1, \dots, t\}$, $X \in \{1, \dots, \omega\}$, the random variable $c_{\text{ind}_\nu(X), j} \oplus \text{Err}(X, K_j)$ is equal to $\text{GT}(X, K_j)$ where the key is uniformly random.

Since $c_{\text{ind}_\nu(X_{j_0}^{(i)}), \tau_{j_0}^{(i)}} = \text{GT}(\text{ind}_\nu(X_{j_0}^{(i)}), T_{\tau_{j_0}^{(i)}})$, we hence need to compute:

$$\Pr\left(\left[\bigoplus_{j \neq j_0} \text{GT}(X_j^{(i)}, K_{\tau_j^{(i)}})\right] \oplus \text{Err}(X_{j_0}^{(i)}, K_{\tau_{j_0}^{(i)}}) = Y^{(i)} \oplus \text{GT}(\text{ind}_\nu(X_{j_0}^{(i)}), T_{\tau_{j_0}^{(i)}} - 1) \mid T_{\tau_{j_0}^{(i)}} = \alpha\right)$$

Let $\beta \in \{0, 1\}$ be the constant $Y^{(i)} \oplus \text{GT}(\text{ind}_\nu(X_{j_0}^{(i)}), T_{\tau_{j_0}^{(i)}} - 1)$. We then need:

$$\Pr\left(\left[\bigoplus_{j \neq j_0} \text{GT}(X_j^{(i)}, K_{\tau_j^{(i)}})\right] \oplus \text{Err}(X_{j_0}^{(i)}, K_{\tau_{j_0}^{(i)}}) = \beta \mid \text{tresh}_\nu(K_{\tau_{j_0}^{(i)}}) = \alpha\right) \quad (10)$$

The left-hand side being a sum of ℓ Bernoulli variables, that we assume to be independent, it is enough to study the distribution of each one of them in order to get the distribution of the sum (through the piling-up lemma).

Distributions of the terms of the sum. We need, for $j \neq j_0$, to study the distribution of the term $[\text{GT}(X_j^{(i)}, K_{\tau_j^{(i)}})]$. Since $X_j^{(i)}$ is fixed, the probability is exactly $\sum_{k=0}^{X_j^{(i)}} \Pr(K_{\tau_j^{(i)}} = k)$, which in turn we compute as:

$$\sum_{\alpha=1}^{\text{ind}_\nu(X_j^{(i)})-1} \Pr(K_{\tau_j^{(i)}} \in \{(\alpha-1) \cdot \text{step}, \dots, \alpha \cdot \text{step} - 1\}) + \Pr(K_{\tau_j^{(i)}} \in \{\text{ind}_\nu(X_j^{(i)}) \cdot \text{step}, \dots, X_j^{(i)}\}).$$

If we are at check node c_i , sending a message to variable $\tau_{j_0}^{(i)}$ at step r of BP, the probability $\Pr(K_{\tau_j^{(i)}} \in \{(\alpha-1) \cdot \text{step}, \dots, \alpha \cdot \text{step} - 1\})$ is captured by the extrinsic information carried by the message $\mu_{v_i \rightarrow c_{\tau_j^{(i)}}}^{(r)}(\alpha)$. Thus, for $j \neq j_0$, we have $\Pr(\text{GT}(X_j^{(i)}, K_{\tau_j^{(i)}}) = 1 \mid \text{extrinsic information at node } i, \text{ round } r)$ equals:

$$\sum_{\alpha=1}^{\text{ind}_\nu(X_j^{(i)})-1} \mu_{v_i \rightarrow c_{\tau_j^{(i)}}}^{(r)}(\alpha) + \frac{|\text{app}_\nu(X_j^{(i)}) - X_j^{(i)} - 1|}{\text{step}} \cdot \mu_{v_i \rightarrow c_{\tau_j^{(i)}}}^{(r)}(\text{ind}_\nu(X_j^{(i)})) \quad (11)$$

where "extrinsic information at node i , round r " means access to all the messages $\mu_{v_i \rightarrow c_{\tau_j^{(i)}}}^{(r)}(\cdot)$ sent at to check node i at iteration r by neighbors except $\tau_{j_0}^{(i)}$. Denote the quantity from Equation (11) as $\frac{1}{2}(1 - \varepsilon_{i,j})$.

For $\alpha \in \{1, \dots, \nu\}$, we then investigate the distribution of the following quantity: $[\text{Err}(X_{j_0}^{(i)}, K_{\tau_{j_0}^{(i)}}) | K_{\tau_{j_0}^{(i)}} \in \{(\alpha - 1) \cdot \text{step}, \dots, \alpha \cdot \text{step}\}]$.

For this quantity, the derivation depends on the value $X_{j_0}^{(i)}$. Indeed, if $X_{j_0}^{(i)}$ is not in the interval $\{(\alpha - 1) \cdot \text{step}, \dots, \alpha \cdot \text{step} - 1\}$, then the approximation can't cause an error. But, if $X_{j_0}^{(i)}$ lands in $\{(\alpha - 1) \cdot \text{step}, \dots, \alpha \cdot \text{step} - 1\}$, then an error can occur. An error will occur if and only if the key $K_{\tau_{j_0}^{(i)}}$ lands between $\text{app}_\nu(X_{j_0}^{(i)})$ and $X_{j_0}^{(i)}$, where the rightmost of the two is excluded. Since it is assumed that the key belongs to $\{(\alpha - 1) \cdot \text{step}, \dots, \alpha \cdot \text{step} - 1\}$, and the key is assumed to be uniformly distributed in this interval, it yields that $\Pr(\text{Err}(X_{j_0}^{(i)}, K_{\tau_{j_0}^{(i)}}) = 1 \mid \text{extrinsic information at node } i, \text{ round } r))$ equals:

$$\begin{cases} 0 & \text{if } X_{j_0}^{(i)} \notin \{(\alpha - 1) \cdot \text{step}, \dots, \alpha \cdot \text{step}\} \\ \frac{|X_{j_0}^{(i)} - \text{app}_\nu(X_{j_0}^{(i)}) - 1|}{\text{step}} & \text{if } X_{j_0}^{(i)} \in \{(\alpha - 1) \cdot \text{step}, \dots, \alpha \cdot \text{step} - 1\} \end{cases} \quad (12)$$

Denote this quantity as $\frac{1}{2}(1 - \varepsilon_{i,j_0}(\alpha, X_{j_0}^{(i)}))$ in what follows.

Derivation of the formula for check to variable messages We can now derive the formula used at iteration r in our belief propagation algorithm to pass messages from check node i for $i \in \{1, \dots, N\}$ to variable node τ for $\tau \in \{1, \dots, t\}$ (where there exists some j_0 such that $\tau_{j_0}^{(i)} \in \mathcal{NC}(i)$ and $\tau = \tau_{j_0}^{(i)}$): $\forall \alpha \in \{1, \dots, \nu\}, \mu_{c_i \rightarrow v_\tau}^{(r+1)}(\alpha) \propto \frac{1}{2}(1 + \varepsilon)$, where ε is computed as follows, according to the piling up lemma:

$$\varepsilon = \varepsilon_{i,j_0}(\alpha, X_{j_0}^{(i)}) \cdot \prod_{j \neq j_0} \varepsilon_{i,j} \quad (13)$$

5 Complexity analysis

In this section, we make a thorough analysis of the data, time and memory complexity of our attack. We start in Section 5.1 by establishing a bound on the number of samples that we need for the belief propagation algorithm to converge to a good solution. Then, in Section 5.3, we use detailed experiments to refine this bound, and explain why the practical complexity is smaller than anticipated. This helps us to refine our complexity analysis of Section 6.

5.1 Theoretical data complexity

From product to sum. Let $k = (k_1, \dots, k_t)$ be a fixed key. The output of the BP algorithm is a list of *thresholds* ($\text{tresh}_\nu(k_1), \dots, \text{tresh}_\nu(k_t)$) for the key

words. Let $\widehat{\mathbf{tresh}}_\nu(k_v)$ be the estimate for the variable node v after one round of belief propagation. By definition, $\widehat{\mathbf{tresh}}_\nu(k_v)$ is the value where b_v (the belief at the variable node v) reaches its maximum:

$$\widehat{\mathbf{tresh}}_\nu(k_v) = \operatorname{argmax}_{x \in \{1, \dots, \nu\}} b_v(x) .$$

Then, as the belief is computed as a product of messages (see Equation (6)), it is a usual technique to study the log-likelihood ratios L_v of the belief. For every candidate $x \neq \mathbf{tresh}_\nu(k_v)$, we define $L_v(x)$ and the distance d to its estimated value as

$$L_v(x) = \log \frac{b_v(\mathbf{tresh}_\nu(k_v))}{b_v(x)} , \text{ and } d := |x - \mathbf{tresh}_\nu(k_v)| .$$

The probability that BP fails is linked to the sign of $L_v(x)$ as follows.

Lemma 2. *Let $v \in \{1, \dots, t\}$.*

- (i) *If $L_v(x) > 0$ for every candidate $x \neq \mathbf{tresh}_\nu(k_v)$, then $\widehat{\mathbf{tresh}}_\nu(k_v) = \mathbf{tresh}_\nu(k_v)$.*
- (ii) *If $L_v(x) > 0$ for every x with $|x - \mathbf{tresh}_\nu(k_v)| \geq 2$, then $|\widehat{\mathbf{tresh}}_\nu(k_v) - \mathbf{tresh}_\nu(k_v)| \leq 1$.*

We can then upper-bound the following probabilities,

$$\Pr(\widehat{\mathbf{tresh}}_\nu(k_v) \neq \mathbf{tresh}_\nu(k_v)) \leq \sum_{x \neq \mathbf{tresh}_\nu(k_v)} \Pr(L_v(x) \leq 0), \quad (14)$$

$$\Pr(|\widehat{\mathbf{tresh}}_\nu(k_v) - \mathbf{tresh}_\nu(k_v)| \geq 2) \leq \sum_{|x - \mathbf{tresh}_\nu(k_v)| \geq 2} \Pr(L_v(x) \leq 0). \quad (15)$$

Proof. Items (i) and (ii) are a direct consequence of the definitions of the distance d and of L_v . The two bounds follow by a union bound over the events described respectively in (i) and (ii).

We now write $L_v(x)$ as a sum of independent contributions, one per check node, to estimate $\Pr(L_v(x) \leq 0)$. Let $v \in \mathcal{NC}(i)$. After one round, the message the check node i sends to the variable node v is obtained, as in Section 4, by the piling-up lemma:

$$\mu_{c_i \rightarrow v_v}^{(1)}(x) \propto \frac{1}{2} \left(1 + (-1)^{Y^{(i)}} \tilde{\varepsilon}_{i,v}(x) \prod_{w \in \mathcal{NC}(i) \setminus \{v\}} \varepsilon_{i,w} \right), \quad x \in \{0, \dots, \nu\} \quad (16)$$

with $\tilde{\varepsilon}_{i,v}(x) = (-1)^{\hat{c}_i(x)} \varepsilon_{i,v}(x, X_v^{(i)})$ and $\hat{c}_i(x) = \text{GT}(\text{ind}_\nu(X_v^{(i)}), x)$. Following Equation (3), the contribution of the check node i to $L_v(x)$ is then

$$\Lambda_i(x) := \log \frac{1 + (-1)^{Y^{(i)}} \tilde{\varepsilon}_{i,v}(\mathbf{tresh}_\nu(k_v)) \prod_{w \neq v} \varepsilon_{i,w}}{1 + (-1)^{Y^{(i)}} \tilde{\varepsilon}_{i,v}(x) \prod_{w \neq v} \varepsilon_{i,w}}, \quad (17)$$

the products running over $w \in \mathcal{NC}(i) \setminus \{v\}$. It is defined for the check nodes i incident to v , that is $i \in \mathcal{NV}(v)$.

Lemma 3. Let $v \in \{1, \dots, t\}$ and $x \neq \mathbf{tresh}_\nu(k_v)$. After one round of belief propagation,

$$L_v(x) = \kappa_v(x) + \sum_{i \in \mathcal{NV}(v)} \Lambda_i(x),$$

where p_v is the prior at the variable node v , used as the likelihood in Equation (6), $\kappa_v(x) = \log \frac{p_v(\mathbf{tresh}_\nu(k_v))}{p_v(x)}$ is a constant, and the sum runs over the $d_v := |\mathcal{NV}(v)|$ check nodes incident to v . The variables $\Lambda_i(x)$ are independant.

Proof. By Equation (6), b_v is the prior p_v multiplied by the messages (given in Equation (16)) of the check nodes incident to v , and normalised. The normalisation does not depend on x and cancels in the ratio defining $L_v(x)$. What remains is the ratio of the priors times the product, over the check nodes incident to v , of the ratio of Equation (16) at $\mathbf{tresh}_\nu(k_v)$ and at x , whose logarithm is the sum of the $\Lambda_i(x)$. The independence of the $\Lambda_i(x)$ follows from their dependency on the $X_w^{(i)}$ alone.

Remark 5. The $\Lambda_i(x)$ are not independent and identically distributed since they depend on their own fixed key. However, we will average their behaviour later so as to use the limit theorem.

Expectation and Variance of Λ_i . Now that $L_v(x)$ is a sum of variables $\Lambda_i(x)$, $i \in \mathcal{NV}(v)$, our goal is to compute their expectation and their variance. By first order approximation of log, we have, for an incident check node $i \in \mathcal{NV}(v)$,

$$\Lambda_i(x) \approx (-1)^{Y^{(i)}} \left(\tilde{\varepsilon}_{i,v}(\mathbf{tresh}_\nu(k_v)) - \tilde{\varepsilon}_{i,v}(x) \right) \prod_{w \neq v} \varepsilon_{i,w}. \quad (18)$$

Since $Y^{(i)} = \bigoplus_{w \in \mathcal{NC}(i)} G_{i,w}$ with $G_{i,w} = \text{GT}(X_w^{(i)}, k_w)$, we have

$$(-1)^{Y^{(i)}} \prod_{w \neq v} \varepsilon_{i,w} = (-1)^{G_{i,v}} \prod_{w \neq v} (-1)^{G_{i,w}} \varepsilon_{i,w}.$$

Using the remark of Section 4.3, the piling-up lemma turns the expectation of this product into a product of expectations:

$$\mathbb{E} \left[(-1)^{Y^{(i)}} \prod_{w \neq v} \varepsilon_{i,w} \mid X_v^{(i)} \right] = (-1)^{G_{i,v}} \prod_{w \neq v} \mathbb{E} [(-1)^{G_{i,w}} \varepsilon_{i,w}].$$

Then, let us denote

$$\begin{aligned} \Delta_i(x) &:= (-1)^{G_{i,v}} \left(\tilde{\varepsilon}_{i,v}(\mathbf{tresh}_\nu(k_v)) - \tilde{\varepsilon}_{i,v}(x) \right), \quad \text{and} \\ \Sigma_v(x) &:= \prod_{w \in \mathcal{NC}(i) \setminus \{v\}} (-1)^{G_{i,w}} \varepsilon_{i,w}. \end{aligned}$$

To first focus on a single edge, recall that $\varepsilon_{i,w} = 1 - 2h(\mathbf{ind}_\nu(X_w^{(i)}))$ at the first round, since it is a function of the query alone, with h being the first round

check value of Section 4.3. In what follows, we say that a query X is in the bucket m if $\text{ind}_\nu(X) = m$, so that the set $\{x \mid \text{ind}_\nu(X) = m\}$ is the query bucket m . Modelling the key as a uniform K , independent of the query, we define $h(\text{ind}_\nu(X_w^{(i)}))$ by:

$$h(\text{ind}_\nu(X_w^{(i)})) = \Pr(\text{GT}(X_w^{(i)}, K) = 1 \mid \text{ind}_\nu(X_w^{(i)})). \quad (19)$$

Here, K follows the first round prior, not the fixed key k_w of the edge: the message is built before anything is known about k_w , so h , and hence $\varepsilon_{i,w}$, depends on the query through its bucket $\text{ind}_\nu(X_w^{(i)})$ alone. The fixed key k_w impacts only the observed bit $G_{i,w} = \text{GT}(X_w^{(i)}, k_w)$. Our goal is now to compute the mean behaviour of $\Sigma_i(x)$ over the keys, starting with the terms $(-1)^{G_{i,w}} \varepsilon_{i,w}$. We prove the following technical proposition in Appendix B.1

Proposition 2. *Fix an edge (i, w) with key k_w , and let the query $X_w^{(i)}$ be uniform over $\{1, \dots, \omega\}$ and assume $2\nu \mid \omega$. The mean value π of $\mathbb{E}_{X_w^{(i)}}[(-1)^{G_{i,w}} \varepsilon_{i,w}]$ over uniform keys satisfies:*

$$\pi := \frac{1}{\omega} \sum_{k_w=1}^{\omega} \mathbb{E}_{X_w^{(i)}}[(-1)^{G_{i,w}} \varepsilon_{i,w}] = \mathbb{E}_{X_w^{(i)}}[\varepsilon_{i,w}^2]. \quad (20)$$

Furthermore,

$$\pi = \frac{1}{3} - \frac{1}{3\nu^2} + \frac{1}{4\nu^3} + \frac{\nu-1}{\nu\omega^2}. \quad (21)$$

Using Lemma 4 that we prove in Appendix B.2, we conclude the study of the expectancy and variance, by following proposition.

Proposition 3 (Expectancy and Variance of $\Lambda_i(x)$). *Assume the approximation of Equation (18), so that $\Lambda_i(x) = \Delta_i(x) \Sigma_v(x)$. For a fixed key k and a candidate x interior at distance d , $\mu_v(x) = \frac{2d}{\nu} \prod_{w \neq v} \mathbb{E}_{X_w^{(i)}}[(-1)^{G_{i,w}} \varepsilon_{i,w}]$, $\sigma_v(x)^2 = \frac{4d}{\nu} \mathbb{E}[\varepsilon_{i,w}^2]^{\ell-1} - \mu_v(x)^2$. Further, assuming mean behaviour of the $\ell-1$ keys of the other edges, i.e., $\mathbb{E}_{X_w^{(i)}}[(-1)^{G_{i,w}} \varepsilon_{i,w}] \approx \pi$, we have*

$$\mu \approx \frac{2d}{\nu} \pi^{\ell-1} \quad \text{and} \quad \sigma^2 \approx \frac{4d}{\nu} \pi^{\ell-1}.$$

Estimating $\Pr(L_v(x) \leq 0)$. Assuming independence of sample and the mean behavior of Proposition 3, we now use limits theorems on the sum of $\Lambda_i(x)$ considered as i.i.d variables to bound $\Pr(L_v(x) \leq 0)$, and to estimate the number of samples needed for BP to succeed in this ideal setting. We prove the following in Appendix B.3

Proposition 4. *Let $v \in \{1, \dots, t\}$ and $x \neq \text{tresh}_\nu(k_v)$, the key k being fixed, and assume $\sigma_v(x) > 0$. Let Φ be the distribution function of the standard normal law, let $\varrho := \mathbb{E}[|\Lambda_i(x) - \mu|^3]$ and*

$$z_v(x) := \frac{\mathbb{E}[L_v(x)]}{\sqrt{\text{Var}(L_v(x))}} = \frac{\kappa_v(x) + d_v \mu_v(x)}{\sigma_v(x) \sqrt{d_v}} \approx \sqrt{\frac{d \pi^{\ell-1}}{\nu d_v}}.$$

Under the assumptions of Proposition 3, the following holds:

(i) by Berry-Esseen's theorem, we have:

$$|\Pr(L_v(x) \leq 0) - \Phi(-z_v(x))| \leq 0.4748 \frac{\varrho}{\sigma^3 \sqrt{d_v}}, \text{ and}$$

(ii) by Bernstein's bound, if $z_v(x) > 0$ and $|A_v(x)| \leq B$, then

$$\Pr(L_v(x) \leq 0) \leq \exp\left(-\frac{z_v^2(x)}{2} \cdot \frac{1}{1 + \frac{2B \cdot z_v(x)}{3\sigma_v(x)\sqrt{d_v}}}\right) \lesssim \exp\left(-\frac{z_v^2(x)}{2}\right).$$

Remark 6. We use points (i) and (ii) independently. The first is used to justify a Gaussian fit in the Section 5.3, the second to estimate a lower bound on the number of samples needed to have $\Pr(L_v(x) \leq 0) = \delta$ for a fixed probability δ .

Estimation of the data complexity Using a union bound over the t variable nodes and the results above, we derive the number of samples one round of belief propagation needs in this ideal setting. We assume that each key has a mean behaviour, that is, $z_v(x) = \sqrt{\frac{d \pi^{\ell-1}}{\nu d_v}}$, and that the edges are uniformly distributed, i.e. $d_v = \frac{\ell N}{t}$. Using (ii) from Proposition 7, we get

$$\Pr(\widehat{\mathbf{tresh}_\nu(k_v)} \neq \mathbf{tresh}_\nu(k_v)) \lesssim 2 \sum_{d \geq 1} e^{-\frac{d \mathbb{E}[\varepsilon_{i,w}^2]^{\ell-1}}{2\nu d_v}} \approx 2 e^{-\frac{\pi^{\ell-1}}{2\nu d_v}},$$

the dominant term being for $d = 1$, i.e., the nearest candidates. A union bound over the t variable nodes gives $\Pr(\exists v : \widehat{\mathbf{tresh}_\nu(k_v)} \neq \mathbf{tresh}_\nu(k_v)) \leq 2t e^{-\frac{\pi^{\ell-1}}{2\nu d_v}}$, which is at most δ provided that $-\frac{\pi^{\ell-1}}{2\nu d_v} \geq 2 \ln(2t/\delta)$. We then have

$$N \geq \frac{2t\nu}{\ell} \pi^{-(\ell-1)} \ln \frac{2t}{\delta} \approx \frac{2t\nu}{\ell} 3^{\ell-1} \ln \frac{2t}{\delta}, \quad (22)$$

using $\mathbb{E}[\varepsilon_{i,w}^2] \approx \frac{1}{3}$. This is the number of samples that recovers the entire key exactly after a single round, with probability at least $1 - \delta$. If we want to get the probability to recover thresholds at distance $d = 2$, we just sum over values satisfying $|x - \mathbf{tresh}_\nu(k_v)| \geq 2$ in Equation (5.1). We then get a factor 2 in the exponent as we miss out on the terms at $d = 1$, that translates into a factor 2 in the estimation of the number of samples.

5.2 Iteration under ideal conditions

Proposition 3 describes the behavior of the “signal” over a single round. Iterating belief propagation sharpens the messages, but making rigorous estimations becomes harder in non-tree-like Tanner graphs. Assuming ideal conditions, we still propose a recursion formula that produces estimations of d_v that match our experimental results.

Proposition 5 (Signal after r rounds). *Assume that each round of BP is independent and assume mean behavior of the keys. Let $\varepsilon_{i,w}^{(r)}$ be the round- r variable-to-check message on the edge (i, w) . We have*

$$\pi_r := \mathbb{E}[(-1)^{G_{i,w}} \varepsilon_{i,w}^{(r)}] = \mathbb{E}[(\varepsilon_{i,w}^{(r)})^2] .$$

Assume that $\pi_1 = \pi$ and that, at first order, we have $\pi_{r+1} \approx (d_v - 1) \pi_r^{\ell-1}$. This recursion has a repelling fixed point $\pi^* = (d_v - 1)^{-\frac{1}{\ell-2}}$, which means that, if $\pi > \pi^*$, then π_r increases with r and BP converges, whereas if $\pi < \pi^*$ it decreases to 0.

Proof. The recursion formula stems from direct application of the proof strategy of the results above starting from π_r under the assumptions of the proposition. Then, monotony results from the fact that π^* is the fixed point of the recursion, i.e, it verifies $(d_v - 1)(\pi^*)^{\ell-1} = \pi^*$.

Remark 7. With $\pi \approx \frac{1}{3}$, at the limit in ν , the fixed point bound gives us $d_v \gtrsim 3^{\ell-2}$, i.e., comparable to 3^ℓ as the one-round estimate (22). Take the aggressive setting $\ell = 11$, $\nu = 100$, $t = 660$. By Lemma 2, $\pi = 0.3333$, so the converging condition $d_v \gtrsim \pi^{-(\ell-2)} = \pi^{-9}$ gives $d_v \gtrsim 1.97 \times 10^4 \approx 3^9$ (the threshold is essentially independent of ν , since $\pi \approx \frac{1}{3}$ for every ν). With uniformly distributed edges, $d_v = \frac{\ell N}{t}$, and so we deduce the following lower bound on the amount of data needed to build a successful BP instance:

$$N \gtrsim \frac{t}{\ell} \pi^{-9} = 60 \times 1.97 \times 10^4 \approx 1.18 \times 10^6 \approx 2^{20.2},$$

which is very close to the experimental results we present in Section 5.3.

5.3 Practical data complexity

As reported in the introduction, our attack runs in a few minutes on a regular desktop. This means that finding the amount necessary to break the aggressive instance of EALPN is feasible empirically. However, in light of our theoretical analysis, we rigorously justify the extrapolations of complexities. In what follows, we fix ω and t and analyse the impact of ℓ on the complexities.

Gaussian fit. Following the one round ideal estimation of (i) in Proposition 7, a normal law's cumulative distribution function is a good approximation of the probability $\Pr(L_v(x) \leq 0)$. This suggests that, experimentally, a Gaussian fit is a reasonable method to extrapolate the number of samples needed for belief propagation to succeed. Further, our small dimension experiments and other studies of the BP algorithm support this behavior. Typically, our small dimension experiments follow the shape of Figure 3: a stagnation at low sample count followed by a “sigmoid” growing to 1.

However, as again suggested by our theoretical study, finding the candidates key threshold at distance $d = |\widehat{\text{tresh}}_\nu(k_i) - \text{tresh}_\nu(k_i)| = 1$ is the hardest

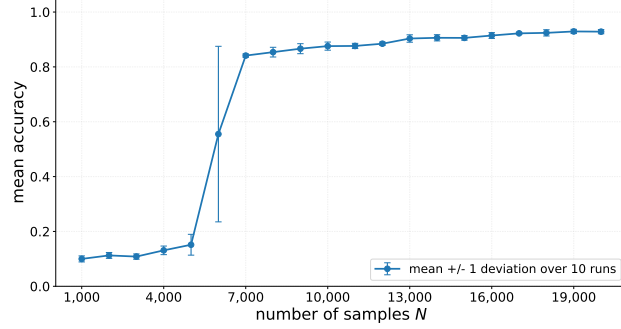


Fig. 3: An example of the experimental accuracy $acc = \frac{\#\{\widehat{\text{tresh}}_\nu(k_i) = \text{tresh}_\nu(k_i)\}}{t}$ as a function of the number of samples N for a toy example with $l = 6$, $t = 660$

part: it needs a larger sample set that allows to discriminate that region. This is confirmed by our experiments. Further, if a ν -approximation is a good approximation of EALPN, then an instance with thresholds at distance $d = 1$ is a comparable approximation: in the case where $d = 1$, a comparison error on one term of the XOR-sum is made roughly with probability $\mathcal{O}(\frac{1}{\nu})$, which is dampened by the pilling-up lemma to a power ℓ . We report later the experimental accuracy for these instances. We then fit the mean experimental accuracy:

$$acc_\nu^{\pm 1}(N) = \frac{\#\{i \mid |\widehat{\text{tresh}}_\nu(k_i) - \text{tresh}_\nu(k_i)| \leq 1\}}{t}$$

with a Gaussian fit of the shape

$$acc_\nu^{\pm 1}(N) = f + (1 - f) \Phi\left(\frac{N - N^{(0.5)}}{s}\right), \quad f = \frac{3q - 2}{q^2}, \quad q = \nu + 1,$$

with $N^{(0.5)}$, s and f being parameters to recover. We chose the floor and the ceiling of the curve to have better fitting. For the floor, represented by f , we assume a random behavior of BP when the number of samples is not large enough, i.e., that f is equal to the mean of the accuracy of a random vector of thresholds. The ceiling was chosen to be 1 since $acc_\nu^{\pm 1}(N)$ reached 1 on a large number of samples in many of our experiments. Note that, by definition, $N^{(0.5)}$ is the inflexion point, i.e., the number of samples needed to reach $acc_\nu^{\pm 1}(N) = 0.5$. The values $N^{(0.5)}$ and s are recovered by the fitting algorithm. We denote $N^{(0.95)}$ the smallest number of samples such that $acc_\nu^{\pm 1}(N^{(0.95)}) = 0.95$.

Extrapolating $N^{(0.95)}(\ell)$. Our goal is now to derive a formula for $N^{(0.95)}(\ell)$ that is more precise than the theoretical estimation proportional to $3^{\ell-1}$ from Equation (22). To do so, we ran experiments over large samples sets at $t = 660$,

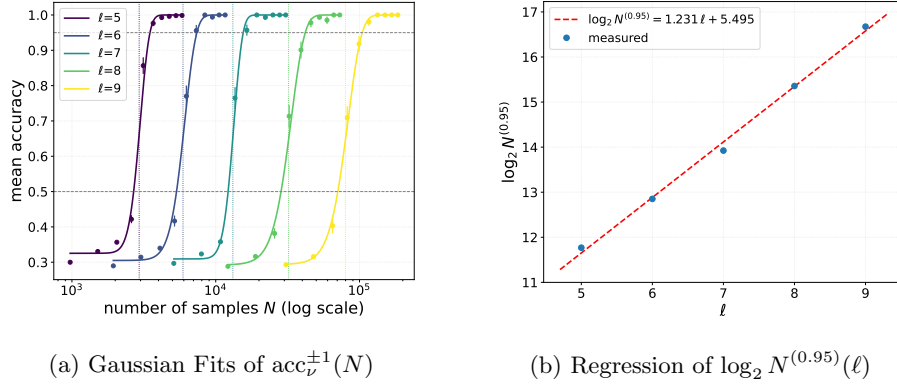


Fig. 4: Experimental results at $t = 660$, $\nu = 10$, over $\ell = 5$ to 9, 100 repeats.

$\nu = 10$ and ℓ from 5 to 9, with a bound of $n_r = 100$ round of BP and a precision threshold of 10^{-5} . Each experiment was repeated 100 times to avoid statistical anomalies while not requiring too much computational power. We then performed a Gaussian regression for each data set and read off the high-accuracy points $N^{(0.95)}$ (Figure 4(a)). Finally, we performed a linear regression on the logarithm in base 2 of $N^{(0.95)}(\ell)$ (Figure 4(b)). We then extrapolate:

$$N^{(0.95)}(\ell) = 2^{1.23\ell + 5.5}. \quad (23)$$

Remark 8. Note that we used $\nu = 10$ for this estimation. As π grows with ν , this means that Equation (23) is a lower bound for the number of samples needed for greater values of ν . We then use this estimation in what follows.

6 Attack on aggressive EALPN

This section closes the complexity analysis of our attack against EALPN that we started in Section 5.1. In particular, we show that $\ell > 32$ should be chosen for the designers to keep their security claim.

Our attack strategy. With the technical details above, we refine the description of our attack. As we explained in the theoretical analysis and confirmed in the experimental analysis, obtaining the exact key thresholds requires more samples than would be needed to get a good approximation of EALPN, e.g. with the relaxed condition of having key thresholds at distance $d = 1$. Our attack then focuses on maximizing the accuracy $\text{acc}_\nu^{\pm 1}$ at a minimal data complexity.

To do so, we fix $n_s = N^{(0.95)}$ and draw $2n_s$ samples. We then run 4 instances of BP with n_s samples drawn at random. We then decide the key thresholds by making a majority vote among the instances. We use this strategy to avoid too much influence from a “bad” sample set having too many cycles in the Tanner graph, which could cause BP to fail.

Time and memory complexity. To avoid hiding multiplicative coefficients in our complexity analysis, we report the exact operations performed by the experimental code depending on the parameters of an EALPN instance, i.e. ℓ, t, ν and the experimental parameters n_s and n_r , the upper-bound we allow for BP before returning an answer. Each value (thresholds and probability vectors) can be stored on 32 bits, as is done in the code in the supplementary material.

A run of the BP algorithm described in the supplementary material makes about 30 multiplications and additions of arrays of size either $n_r \cdot n_s \cdot \ell \cdot \nu$ or $n_r \cdot t \cdot \nu$. Similarly, it keeps in memory 4 arrays of size either $n_s \cdot \ell \cdot \nu$ or $n_s \cdot t \cdot \nu$. We then upper-bound the time and memory complexity of running BP by:

Time: $30 \cdot n_r \cdot (n_s \cdot \ell + t) \cdot \nu$, Memory: $4 \cdot 32 \cdot (n_s \cdot \ell + t) \cdot \nu = 128 \cdot (n_s \cdot \ell + t) \cdot \nu$, the factor 4 accounting for the tables and the factor 32 the bits per stored value. Table 1 instantiates these two bounds at $t = 660$ and $n_r = 100$, and compares them to the theoretical estimation.

Estimate	ℓ	$\nu = 10$				$\nu = 50$				$\nu = 100$			
		P_{app}	$\log_2 n_s$	$\log_2 T$	$\log_2 M$	P_{app}	$\log_2 n_s$	$\log_2 T$	$\log_2 M$	P_{app}	$\log_2 n_s$	$\log_2 T$	$\log_2 M$
Experimental	11	0.784	19.0	37.4	32.8	0.948	19.0	39.7	35.1	0.973	19.0	40.7	36.1
Experimental	32	0.597	44.9	64.8	60.2	0.862	44.9	67.1	62.5	0.926	44.9	68.1	63.5
Recursion	11	0.784	20.3	38.6	34.1	0.948	20.2	40.8	36.3	0.973	20.2	41.8	37.3
Recursion	27	0.625	44.6	64.2	59.6	0.881	44.2	66.2	61.6	0.937	44.2	67.2	62.6
One round	11	0.784	29.6	41.2	43.3	0.948	31.8	45.8	47.9	0.973	32.7	47.8	49.9

Table 1: Time complexity (T), in 32-bit operations and memory complexity (M) in bits, at $t = 660$, with $n_r = 100$ rounds. The last row instead reports the single-round data bound of Equation (22) at $\delta = 0.05$. The column P_{app} is the probability that the ν -approximation agrees with EALPN of Proposition 1.

The first conclusion is that we need only about 2^{19} samples to break the aggressive instance of EALPN. In practice, the whole attack runs in 10 minutes on a regular desktop with enough memory (i.e. at least 22 Go of RAM). The code needed to launch this attack is available in the supplementary material. The second point is that, using the extrapolation of $N^{(0.9)}$, we can state that at fixed $t = 660$ and $\nu = 100$, our attack is within the query budget allowed by the designers with ℓ up to 32. As our attack is only exponential in ℓ , as suggested by our theoretical estimation and as is confirmed by the experiments, users should take ℓ at least such that $\min(\frac{t}{\ell} 3^{\ell-2}, 2^{1.23\ell+5.5})$ is greater than the query budget. The table 1 seems to confirm that the estimation of $N^{(0.95)} \approx \frac{t}{\ell} \pi^{\ell-2}$ of Proposition 5 is the better theoretical estimation.

The BP attack on the countermeasure. In the hardened variant of Section 2.4, each term in the sum is $c_{\tau_j} \oplus \beta_{\tau_j} \cdot \text{GT}(X, k_{\tau_j})$, and since c_{τ_j} follows a Bernoulli(1/2)

distribution, then each term is itself Bernoulli($1/2$) and the biases $\varepsilon_{i,j}$ are nullified. The “signal” that we defined as $\pi = \mathbb{E}_{X_w^{(i)}}[\varepsilon_{i,w}^2]$ in Equation (20) is then $\pi = 0$, and the BP algorithm cannot move from its starting point. One possible direction for an attacker to bypass this would be to sum equations with the same τ to remove the c_{τ_j} entirely. But then, the problem becomes obtaining enough such equations from the oracle, which happens with a probability that is too small with the 2^{45} samples given.

Note that the masks c_{τ_j} are the most important part of this countermeasure. Indeed, if the terms in the sum where only $\beta_{\tau_j} \cdot \text{GT}(X, k_{\tau_j})$, our algorithm would also work as the “signal” of each edge would only be slightly dampened. By adding more samples, our attack would still be below the allowed query amount.

While the c_{τ_j} and β_{τ_j} both come from the translation into our *greater than* framework of a single countermeasure on the code, we think it is important to highlight that, from a key-size perspective, the additive mask given by the c_{τ_j} have more impact than the multiplication by the β_{τ_j} .

7 Conclusion

In order for weak shallow PRFs to be usable in practice, it is necessary to estimate their security level for fixed parameters, not just asymptotically. We have contributed to this general goal in different ways. First, we clarified and rephrased the distinguisher provided by the authors of [16] to fit the alternative representation of [8]. We hope that this will foster a deeper study of their contributions: while their distinguisher has an overall complexity similar to our more ambitious key recovery, we could also show that the countermeasure they suggested effectively thwarts our attack. The connections that seem to exist between their approach and ours, despite their very different nature, is a topic we believe to need further investigation.

Second, we showed the surprising relevance of the belief propagation algorithm in this context. This sheds a new light on how to analyse ws-PRFs. Indeed, due to the non-chosen plaintext security constraint, most tools of symmetric cryptanalysts cannot be used: that we added a general algorithm to this toolbox is good news. Further, our attack shows that a high total key size can lose its relevance if there is a much lower number of key classes where all the keys in the same class are *quasi-equivalent*. It is the case for EALPN: our attack targets an approximation of the primitive, but the ws-PRF instance it yields is correct with a probability higher than 0.97.

Finally, we highlighted the importance of a practical, experimental study of wsPRFs. Our attack and our investigations that lead to it were greatly informed by experimental observations. We believe that this area of cryptography would benefit from a greater emphasis on this aspect. More generally, relying only on asymptotic formulas to assess security of ws-PRFs can still lead to significant gaps in their actual security.

Acknowledgements

The work of the authors is supported by the European Research Council (ERC) under grant number 101041545 (ReSCALE). AI tools were used for the review and debug of the code in the supplementary material. We thank Christina Boura, Geoffroy Couteau, Yann Rotella for insightful discussions and Jean-Pierre Tillich for pointing us towards the direction of belief propagation.

References

1. Agarwal, A., Baum, C., Braun, L., Scholl, P.: Low-bandwidth mixed arithmetic in VOLE-based ZK from low-degree PRGs. pp. 396–426 (2025). https://doi.org/10.1007/978-3-031-91134-7_14
2. Albrecht, M.R., Davidson, A., Deo, A., Gardham, D.: Crypto dark matter on the torus - oblivious PRFs from shallow PRFs and TFHE. pp. 447–476 (2024). https://doi.org/10.1007/978-3-031-58751-1_16
3. Ayala, I.M., Raddum, H.: Zeroed out: Cryptanalysis of weak PRFs in alternating moduli **2025**(2), 1–15 (2025). <https://doi.org/10.46586/tosc.v2025.i2.1-15>
4. Bariant, A., Boura, C., Germon, B., Boissier, R.H., Meyer-Hilfiger, C., Tiessen, T.: Cryptanalysis of the alternative mod-2/mod-3 weak PRF. Cryptology ePrint Archive, Paper 2026/1982 (2026), <https://eprint.iacr.org/2026/1982>
5. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems **4**(1), 3–72 (Jan 1991). <https://doi.org/10.1007/BF00630563>
6. Boneh, D., Ishai, Y., Passelègue, A., Sahai, A., Wu, D.J.: Exploring crypto dark matter: New simple PRF candidates and their applications. pp. 699–729 (2018). https://doi.org/10.1007/978-3-030-03810-6_25
7. Boucheron, S., Lugosi, G., Massart, P.: Concentration Inequalities: A Nonasymptotic Theory of Independence. Oxford University Press (2013)
8. Boura, C., Couteau, G., Perrin, L., Rotella, Y.: SoK: On shallow weak PRFs A common symmetric building block for MPC protocols **2025**(3), 289–336 (2025). <https://doi.org/10.46586/tosc.v2025.i3.289-336>
9. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Resch, N., Scholl, P.: Correlated pseudorandomness from expand-accumulate codes. pp. 603–633 (2022). https://doi.org/10.1007/978-3-031-15979-4_21
10. Canteaut, A., Trabbia, M.: Improved fast correlation attacks using parity-check equations of weight 4 and 5. In: Advances in Cryptology — EUROCRYPT 2000. LNCS, vol. 1807, pp. 573–588. Springer (2000)
11. Cheon, J.H., Cho, W., Kim, J.H., Kim, J.: Adventures in crypto dark matter: attacks, fixes and analysis for weak pseudorandom functions **90**(8), 1735–1760 (2022). <https://doi.org/10.1007/s10623-022-01071-x>
12. Chose, P., Joux, A., Mitton, M.: Fast correlation attacks: An algorithmic point of view. In: Advances in Cryptology — EUROCRYPT 2002. LNCS, vol. 2332, pp. 209–221. Springer (2002)
13. Forney, G.D.: 6.451 Principles of Digital Communication II, Spring 2005. Massachusetts Institute of Technology: MIT OpenCourseWare (2005), <https://ocw.mit.edu/courses/6-451-principles-of-digital-communication-ii-spring-2005/pages/readings-and-lecture-notes/>, accessed: 2026-09-01. License: Creative Commons BY-NC-SA

14. Freedman, M.J., Ishai, Y., Pinkas, B., Reingold, O.: Keyword search and oblivious pseudorandom functions. pp. 303–324 (2005). https://doi.org/10.1007/978-3-540-30576-7_17
15. Gallager, R.G.: Low-Density Parity-Check Codes. MIT Press (1963)
16. Guo, X., Liu, H., Huang, Z., Cui, H., Zhang, W., Hong, C., Wang, X., Yang, K., Yu, Y.: Dory: Streaming PCG with small memory. In: IEEE Symposium on Security and Privacy, SP 2026, San Francisco, CA, USA, May 18–21, 2026. pp. 1902–1920. IEEE (2026). <https://doi.org/10.1109/SP63933.2026.00079>, <https://doi.org/10.1109/SP63933.2026.00079>
17. Hu, K., Leander, G., Raddum, H., Sandrib, A., Udovenko, A.: Cryptanalysis of two alternating moduli weak PRFs. Cryptology ePrint Archive, Paper 2026/482 (2026), <https://eprint.iacr.org/2026/482>
18. Johansson, T., Meier, W., Nguyen, V.: Differential cryptanalysis of mod-2/mod-3 constructions of binary weak prfs. In: 2023 IEEE International Symposium on Information Theory (ISIT). pp. 477–482 (2023). <https://doi.org/10.1109/ISIT54713.2023.10206853>
19. Mihaljević, M.J., Fossorier, M.P.C., Imai, H.: A low-complexity and high-performance algorithm for the fast correlation attack. In: Fast Software Encryption (FSE 2000). LNCS, vol. 1978, pp. 196–212. Springer (2001)
20. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988)
21. Raghuraman, S., Rindal, P., Tanguy, T.: Expand-convolute codes for pseudorandom correlation generators from LPN. pp. 602–632 (2023). https://doi.org/10.1007/978-3-031-38551-3_19
22. Shevtsova, I.: On the absolute constants in the Berry-Esseen type inequalities for identically distributed summands. arXiv:1111.6554 (2011)
23. Tanner, R.M.: A recursive approach to low complexity codes. IEEE Trans. Inf. Theory **27**(5), 533–547 (1981). <https://doi.org/10.1109/TIT.1981.1056404>, <https://doi.org/10.1109/TIT.1981.1056404>

A Description of the supplementary material

The code of the attack is written in Python. The code uses Sage only for the generation of samples as it uses the reference implementation of [8]. The goal of the supplementary material is to help reproduce the extrapolation of the number of samples $N^{(0.95)}$ and the results of the attack on the aggressive instance of EALPN. Follow the file `experiment_notice.md` for more details.

A.1 Structure of the Folder

The supplementary material contains:

- The reference implementation for EALPN from the article “SoK: On Shallow Weak PRFs” by Boura et al., in the folder `wsPRF`.
- The Python code of our BP, in the script `1_nu_N_bp_benchmark.py`.
- The folder `fixed_1_nu_behavior`, which contains the files to study our algorithm on different sample sizes.
- The folder `attack`, which allows us to reproduce the simulated attacks on EALPN.

A.2 Extrapolating the number of samples for large ℓ

The Python script `experiment_range_N.py` allows to run BP with a range of samples N and store the output data in a JSON file. The script `plot_range_N.py` can be used with `python3 plot_range_N.py <filename>` to get two graphs: one with the accuracy of the output of BP to the indexes of the key, one with the ± 1 accuracy to these indexes, similar to figure 3.

The bash script `run_experiments_15_9.sh` runs `experiment_range_N.py` for $\ell = 5$ to 9 with N varying. With the data output from `run_experiments_15_9.sh`, you can run

```
python3 inflexion_vs_l_probit.py experiment_range_N*_nu10*.json
```

to get the Gaussian regression curves for each ℓ and the linear regression of $\log(N^{0.95})$ and reproduce our results.

A.3 Attack on aggressive EALPN

The code to replicate our attack on the aggressive instance of EALPN is in the `attack/` folder. It is divided in two Python scripts. The script generating an instance of EALPN using is `attack_sample_generation.py` and stores input-output pairs into a JSON file. This script needs Sage, as the reference implementation of `wsPRF` uses it. Then the script `attack_script.py` launches the attack described in the paper and compares the recovered key thresholds to the ones of the instance. To replicate our results and launch the attack, simply run the bash script `run_gen.sh` to generate the corresponding JSON file. Then, run `run_attack.sh` to recover the key thresholds. As reported, this might take up to 10 minutes and needs about 21GB of RAM to properly run.

B Technical Proofs

B.1 Estimation of π

Proposition 6. *Fix an edge (i, w) with key k_w , and let the query $X_w^{(i)}$ be uniform over $\{1, \dots, \omega\}$ and assume $2\nu \mid \omega$. The mean value π of $\mathbb{E}_{X_w^{(i)}}[(-1)^{G_{i,w}} \varepsilon_{i,w}]$ over uniform keys satisfies:*

$$\pi := \frac{1}{\omega} \sum_{k_w=1}^{\omega} \mathbb{E}_{X_w^{(i)}}[(-1)^{G_{i,w}} \varepsilon_{i,w}] = \mathbb{E}_{X_w^{(i)}}[\varepsilon_{i,w}^2]. \quad (24)$$

Furthermore,

$$\pi = \frac{1}{3} - \frac{1}{3\nu^2} + \frac{1}{4\nu^3} + \frac{\nu-1}{\nu\omega^2}. \quad (25)$$

This proposition implies that $\pi \approx \frac{1}{3}$ as ν grows, which is coherent with the intuition that a good approximation is “easier” to obtain when ν is larger.

Proof. Throughout the proof, we write $X := X_w^{(i)}$ for the query, uniform over $\{1, \dots, \omega\}$. Recall from Section 4 that $G_{i,w} = \text{GT}(X, k_w)$ is a Bernoulli variable, equal to 1 exactly when $X > k_w$, so that

$$(-1)^{G_{i,w}} = 1 - 2 \text{GT}(X, k_w), \quad (26)$$

and that, at the first round, the check-to-variable value $\varepsilon_{i,w} = 1 - 2h(\text{ind}_\nu(X))$ is a function of the query bucket $\text{ind}_\nu(X)$ alone. Following Lemma 1 and Section 4.1, we assume $2\nu \mid \omega$, so that $\text{step} = \omega/\nu$ is even and every bucket $\{\text{ind}_\nu(X) = m\}$ carries an integer number of points. For a fixed X , $(-1)^{\text{GT}(X,k)} = -1$ for exactly the $X - 1$ values $k \in \{1, \dots, X - 1\}$ and $(-1)^{\text{GT}(X,k)} = +1$ for the remaining $\omega - (X - 1)$ values, so

$$\frac{1}{\omega} \sum_{k=1}^{\omega} (-1)^{\text{GT}(X,k)} = \frac{1}{\omega} \left((\omega - (X - 1)) - (X - 1) \right) = 1 - \frac{2(X - 1)}{\omega}. \quad (27)$$

Since the prior K in Equation (19) is uniform over $\{1, \dots, \omega\}$ and independent of X , we have $\mathbb{E}[(-1)^{\text{GT}(X,k)} \mid \text{ind}_\nu(X)] = 1 - 2\Pr(\text{GT}(X,k) = 1 \mid \text{ind}_\nu(X))$ and averaging over the uniform key $k = K$ returns $1 - 2h(\text{ind}_\nu(X))$. Thus, $\varepsilon_{i,w}$ is the key-mean of the bucket-conditioned signs, and by Equation (27),

$$\varepsilon_{i,w} = 1 - 2h(\text{ind}_\nu(X)) = \frac{1}{\omega} \sum_{k=1}^{\omega} \mathbb{E}[(-1)^{\text{GT}(X,k)} \mid \text{ind}_\nu(X)] \quad (28)$$

$$= \mathbb{E}\left[1 - \frac{2(X-1)}{\omega} \mid \text{ind}_\nu(X)\right]. \quad (29)$$

Then, expanding the expression of π from Equation (26), and using that $\varepsilon_{i,w}$ does not depend on the summation key k_w , we get

$$\pi = \frac{1}{\omega} \sum_{k_w=1}^{\omega} \mathbb{E}_X[(-1)^{\text{GT}(X,k)} \varepsilon_{i,w}].$$

We further condition each term on $\text{ind}_\nu(X)$, on which $\varepsilon_{i,w}$ is constant, and then swap the finite key-sum with the expectation to obtain:

$$\begin{aligned} \pi &= \frac{1}{\omega} \sum_{k_w=1}^{\omega} \mathbb{E}[\varepsilon_{i,w} \mathbb{E}[(-1)^{\text{GT}(X,k)} \mid \text{ind}_\nu(X)]] \\ &= \mathbb{E}\left[\varepsilon_{i,w} \cdot \frac{1}{\omega} \sum_{k_w=1}^{\omega} \mathbb{E}[(-1)^{\text{GT}(X,k)} \mid \text{ind}_\nu(X)]\right] \\ &= \mathbb{E}_X[\varepsilon_{i,w}^2], \end{aligned}$$

the inner key-mean being $\varepsilon_{i,w}$ itself by Equation (29). This is the middle equality of Equation (20). With $2\nu \mid \omega$, rounding to the nearest grid index gives the following buckets and their means :

- $m \in \{1, \dots, \nu - 1\}$: $\text{ind}_\nu(X) = m$ if and only if X is among step consecutive points of mean $m\text{step} - \frac{1}{2}$, i.e., in $\{(m - \frac{1}{2})\text{step}, \dots, (m + \frac{1}{2})\text{step} - 1\}$,
- $\text{ind}_\nu(X) = 0$ if and only if $X \in \{1, \dots, \frac{\text{step}}{2} - 1\}$, i.e. $\frac{\text{step}}{2} - 1$ points;
- $\text{ind}_\nu(X) = \nu$ if and only if $X \in \{(\nu - \frac{1}{2})\text{step}, \dots, \omega\}$, i.e. $\frac{\text{step}}{2} + 1$ points.

The three sizes sum to $(\frac{\text{step}}{2} - 1) + (\frac{\text{step}}{2} + 1) + (\nu - 1)\text{step} = \nu\text{step} = \omega$. By Equation (29), $h(m) = (\bar{X}_m - 1)/\omega$ with $\bar{X}_m$ the bucket mean, so that, using $\text{step}/\omega = 1/\nu$, we obtain $\varepsilon_{i,w}(m) = 1 - \frac{2m}{\nu} + \frac{3}{\omega}$ when $1 \leq m \leq \nu - 1$, $\varepsilon_{i,w}(0) = 1 - \frac{1}{2\nu} + \frac{2}{\omega}$, and $\varepsilon_{i,w}(\nu) = -1 + \frac{1}{2\nu} + \frac{2}{\omega}$. Weighting each square by its bucket size, we obtain the following expression for π :

$$\mathbb{E}_X[\varepsilon_{i,w}^2] = \frac{1}{\omega} \left[\left(\frac{\text{step}}{2} - 1 \right) \varepsilon_{i,w}(0)^2 + \left(\frac{\text{step}}{2} + 1 \right) \varepsilon_{i,w}(\nu)^2 + \text{step} \sum_{m=1}^{\nu-1} \varepsilon_{i,w}(m)^2 \right].$$

Expanding the interior square and using $\sum_{m=1}^{\nu-1} m = \frac{(\nu-1)\nu}{2}$ and $\sum_{m=1}^{\nu-1} m^2 = \frac{(\nu-1)\nu(2\nu-1)}{6}$, together with $\text{step} = \omega/\nu$, all half-integer boundary terms combine and the sum simplifies to

$$\mathbb{E}_X[\varepsilon_{i,w}^2] = \frac{1}{3} - \frac{1}{3\nu^2} + \frac{1}{4\nu^3} + \frac{\nu-1}{\nu\omega^2} = \frac{4\nu^3 - 4\nu + 3}{12\nu^3} + \frac{\nu-1}{\nu\omega^2},$$

which concludes the proof.

B.2 Expectancy and variance of $\Sigma_i(x)$ and $\Delta_i(x)$ π

We can now combine the previous results to derive the expectancy and variance of $\Sigma_i(x)$ and $\Delta_i(x)$.

Lemma 4 (Expectancy and Variance of Σ_i and Δ_i). *Consider x at distance $d = |x - \text{tresh}_\nu(k_v)|$ and assume mean behaviour for the keys k . Assuming the queries of distinct edges being independent, we have:*

$$\begin{cases} \mathbb{E}[\Sigma_v(x)] = \prod_{w \neq v} \mathbb{E}_{X_w^{(i)}} [(-1)^{G_{i,w}} \varepsilon_{i,w}] , & \text{Var}(\Sigma_v(x)) \approx \mathbb{E}[\varepsilon_{i,w}^2]^{\ell-1} \\ \mathbb{E}[\Delta_i(x)] \approx \frac{2d}{\nu} , & \text{Var}(\Delta_i(x)) \approx \frac{4d}{\nu} . \end{cases}$$

Proof. Since distinct edges carry independent queries and $\varepsilon_{i,w}$ is a function of the query $X_w^{(i)}$ alone, we use throughout that $((-1)^{G_{i,w}})^2 = 1$ and that, by Lemma 2, $\mathbb{E}_{X_w^{(i)}}[\varepsilon_{i,w}^2] = \pi$ is the same for every edge.

By definition of $\Sigma_v(x)$ as a product of $\ell - 1$ independent variables, we have

$$\mathbb{E}[\Sigma_v(x)^2] = \prod_{w \neq v} \mathbb{E}_{X_w^{(i)}} [(-1)^{G_{i,w}} \varepsilon_{i,w}]^2 = \prod_{w \neq v} \mathbb{E}[\varepsilon_{i,w}^2] = \mathbb{E}[\varepsilon_{i,w}^2]^{\ell-1},$$

since we assumed mean behaviour for the keys. Consequently,

$$\text{Var}(\Sigma_v(x)) = \mathbb{E}[\varepsilon_{i,w}^2]^{\ell-1} - \mathbb{E}[\Sigma_v(x)]^2.$$

The subtracted term is $\mathbb{E}[\Sigma_v(x)]^2 \approx \pi^{2(\ell-1)}$, that is, a factor $\pi^{\ell-1} \approx 3^{-(\ell-1)}$ smaller than the leading term $\pi^{\ell-1}$ which proves the first approximations.

For $\Delta_i(x)$, write $m := \text{ind}_\nu(X_v^{(i)})$ and $T := \text{tresh}_\nu(k_v)$. Recall $\tilde{\varepsilon}_{i,v}(y) = (-1)^{\hat{c}_i(y)} \varepsilon_{i,v}(y, X_v^{(i)})$ with $\hat{c}_i(y) = \text{GT}(m, y)$ and reliability $0 \leq \varepsilon_{i,v}(y, \cdot) \leq 1$, which equals 1 unless the query lies in the threshold-bucket of y , i.e. unless $m = y$. As a function of the candidate y , $\text{GT}(m, y)$ is non-increasing and switches from 1 to 0 as y crosses m . We first take the hard-decision value $\varepsilon_{i,v}(y, \cdot) = 1$ and bound the correction afterwards. Then $\tilde{\varepsilon}_{i,v}(y) = (-1)^{\text{GT}(m, y)}$, and

$$\tilde{\varepsilon}_{i,v}(T) - \tilde{\varepsilon}_{i,v}(x) = (-1)^{\text{GT}(m, T)} - (-1)^{\text{GT}(m, x)}.$$

By the monotonicity above, $\text{GT}(m, T)$ and $\text{GT}(m, x)$ disagree exactly when m lies strictly between x and T , that is $m \in \{\min(x, T), \dots, \max(x, T) - 1\}$, a set of $d = |x - T|$ buckets; elsewhere the difference is 0. On that set the difference equals $2 \text{sign}(T - x)$. Using the grid approximation $G_{i,v} = \text{GT}(X_v^{(i)}, k_v) = \text{GT}(m, T)$, valid up to the error probability $p_{\text{err}} = \frac{1}{4\nu} + \frac{1}{2\omega}$ of Lemma 1, we have on this set $(-1)^{G_{i,v}} = +1$ when $x < T$ and -1 when $x > T$, which in both cases cancels $\text{sign}(T - x)$. Hence, we deduce

$$\Delta_i(x) = 2 \cdot [\min(x, T) \leq m < \max(x, T)], \quad \Delta_i(x)^2 = 2 \Delta_i(x). \quad (30)$$

For a candidate for which the d buckets of this set are interior, each of probability $\Pr(\text{ind}_\nu(X_v^{(i)}) = m) = 1/\nu$, we have

$$\mathbb{E}[\Delta_i(x)] = \frac{2d}{\nu}, \quad \text{and} \quad \mathbb{E}[\Delta_i(x)^2] = \frac{4d}{\nu}.$$

The approximation on the variance follows by using $1 - \frac{d}{\nu} \approx 1$. For a candidate for which there are not d values, i.e. for which some values at distance d are over ω or under 0, we argue that the approximation still stands as an upper-bound. Indeed, in this case the number of values where $\Delta_i(x)$ is non-zero is simply reduced.

B.3 Estimating $\Pr(L_v(x) \leq 0)$.

Assuming independence of sample and the mean behavior of Proposition 3, we now use limits theorems on the sum of $\Delta_i(x)$ considered as i.i.d variables to bound $\Pr(L_v(x) \leq 0)$, and to estimate the number of samples needed for BP to succeed in this ideal setting.

Proposition 7. *Let $v \in \{1, \dots, t\}$ and $x \neq \text{tresh}_\nu(k_v)$, the key k being fixed, and assume $\sigma_v(x) > 0$. Let Φ be the distribution function of the standard normal law, let $\varrho := \mathbb{E}[|\Delta_i(x) - \mu|^3]$ and*

$$z_v(x) := \frac{\mathbb{E}[L_v(x)]}{\sqrt{\text{Var}(L_v(x))}} = \frac{\kappa_v(x) + d_v \mu_v(x)}{\sigma_v(x) \sqrt{d_v}} \approx \sqrt{\frac{d \pi^{\ell-1}}{\nu d_v}}.$$

Under the assumptions of Proposition 3, the following holds:

(i) by Berry-Esseen's theorem, we have:

$$|\Pr(L_v(x) \leq 0) - \Phi(-z_v(x))| \leq 0.4748 \frac{\varrho}{\sigma^3 \sqrt{d_v}} , \text{ and}$$

(ii) by Bernstein's bound, if $z_v(x) > 0$ and $|A_v(x)| \leq B$, then

$$\Pr(L_v(x) \leq 0) \leq \exp\left(-\frac{z_v^2(x)}{2} \cdot \frac{1}{1 + \frac{2B \cdot z_v(x)}{3\sigma_v(x)\sqrt{d_v}}}\right) \lesssim \exp\left(-\frac{z_v^2(x)}{2}\right).$$

Proof. The approximation of $z_v(x)$ follows from its definition, previous lemmas and the fact that $\kappa_v(x)$ is negligible before $d_v \mu_v(x)$. Fact (i) is a direct application of the Berry-Esseen's theorem [22] to the sum of variables $A_i - \mu$ that are centered, with a moment of order 3 (namely, ρ) that is finite. It establishes the following bound:

$$\sup_{a \in \mathbb{R}} \left| \Pr\left(\frac{S - d_v \mu}{\sigma \sqrt{d_v}} \leq a\right) - \Phi(a) \right| \leq \frac{0.4748 \varrho}{\sigma^3 \sqrt{d_v}}. \quad (31)$$

Since, L_v is the sum of κ and the A_i by Lemma 3, $\{L_v(x) \leq 0\}$ is equivalent to $\{\sum_i A_i(x) \leq -\kappa\}$ which in turn means $\{\frac{\sum_i A_i - d_v \mu}{\sigma \sqrt{d_v}} \leq -z\}$.

For (iii) The variables $A_i(x) - \mu$, $i \in \mathcal{N}\mathcal{V}(v)$, are independent, centered, of variance σ^2 , and bounded by $|\mu| + B \leq 2B$ (according to the hypothesis). The event $L_v(x) \leq 0$ is $\sum_i (\mu - A_i(x)) \geq \kappa + d_v \mu$. The result follows by Bernstein's inequality [7] after dividing the numerator and the denominator by $d_v \sigma^2$. The last approximation follows by approximating the right factor in the exponent.