Algorithms for Sparse Random 3XOR: The Low-Density Case

Charles Bouillaguet, Claire Delaplace

Algorithms for Sparse Random 3XOR: The Low-Density Case

Charles Bouillaguet
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
charles.bouillaguet@lip6.fr

Claire Delaplace
MIS Laboratory, Université de Picardie Jules Verne, 14 quai de la Somme, 80080 Amiens, France
claire.delaplace@u-picardie.fr

Abstract

We present an algorithm for a variant of the 3XOR problem with lists consisting of n-bit vectors whose coefficients are drawn independently at random according to a Bernoulli distribution of parameter $p < 1/2$. We show that in this particular context the problem can be solved much more efficiently than in the general setting. This leads to a linear algorithm with overwhelming success probability for $p < 0.0957$. With slightly different parameters, this method succeeds deterministically. The expected runtime is also linear for $p < 0.02155$ and always sub-quadratic.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation

Keywords and phrases Algorithms, 3-xor problem, random sparse 3-xor

1 Introduction

Given three lists A, B and C of n-bit vectors, the 3XOR problem consists in deciding the existence of (or even finding) a triplet $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in A \times B \times C$ such that $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z}$ is equal to a given target, often assumed to be zero (here the $\oplus$ symbol represent the exclusive-OR or XOR).

In the general setting where the input lists have size N and can be arbitrary, a simple algorithm decides the existence of a 3XOR triplet in time $\mathcal{O}\left(N^2\right)$, but there is no known way of doing it in time $\mathcal{O}\left(N^{2-\epsilon}\right)$ for any $\epsilon > 0$. Dietzfelbinger, Schlag and Walzer found an algorithm that gain a poly-logarithmic factor over the quadratic algorithm [4].

3XOR can be seen as a variant of the celebrated 3SUM problem, where this time the input list items are seen as integers and we must have $x + y + z = 0$. Many geometric problems can be reduced to 3SUM in sub-quadratic time, and those problems are said to be 3SUM hard [6]. Although the 3XOR problem has enjoyed less interest in the complexity theory field, there exists a few such reductions. For instance, it is a fact that any $\mathcal{O}\left(N^{1+\epsilon}\right)$ algorithm for the 3XOR problem with input lists of size N would imply faster-than-expected algorithms for listing triangles in a graph [7]. In the other direction, some conditional lower-bounds have been established based on the hypothesis that 3XOR is inherently quadratic: [4] shows that the offline SetDisjointness and SetIntersection problems cannot be solved in time $\mathcal{O}\left(N^{2-\epsilon}\right)$ unless 3XOR can.

Besides being a natural extension of 3SUM, the 3XOR problem has some cryptographic applications, in which the input lists consist of uniformly *random* vectors. In particular, we can mention Nandi's attack [10] against the COPA mode of authenticated encryption, or the more recent attack against the two-round single-key Even-Mansour cipher by Leurent and Sibleyras [9]. May and Both have been considering a variant of the 3XOR problem, the *approximate 3-list birthday problem*: given three lists of N uniformly random elements of $\{0,1\}^n$ the goal consist in finding triplets $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ in the list such that the hamming weight of $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z}$ is small [2].

Studying the 3XOR problem with uniformly random inputs seemed natural to the cryptography community (which makes this assumption "by default"), and this lead to several algorithms gaining a polylogarithmic factor, predating [4], and often tailored to specific input sizes [8, 11, 3]. The assumption that the input is *random* makes the problem simpler, but it has not yet been possible to obtain a $\mathcal{O}\left(N^{2-\epsilon}\right)$ algorithm even in this simpler case.

Contributions. In this paper, we consider an even more favorable setting, where the input lists are both *random* and *sparse*, namely where the input bitstrings are biased towards zero in some way. More precisely, we assume that each input bit is drawn independently at random according to a Bernoulli distribution of parameter $0 < p < 1/2$ — the "dense" random case corresponds to $p = 1/2$.

We first give the probability that the input actually contains a 3XOR triplet for a given list size N and density p. To the best of our knowledge, this result was not readily available from the existing literature, not even in the simple "cryptographically relevant" case where $p = 1/2$. We then describe a new algorithm to solve the random low-density 3XOR problem (Section 3). The main idea is to discard useless input vectors (i.e. whose Hamming weight is above a well-chosen threshold), then search a solution using the simple quadratic algorithm. This algorithm returns a solution with overwhelming probability in time linear in N for small density (i.e. $p < 0.0957$). A slight variation of the same algorithm can also be used to deterministically return a solution if there is one in the input. In this case, the expected running time of the procedure is also linear in N when p is smaller than 0.02155. The algorithm has a time complexity of $\mathcal{O}\left(N + N^e\right)$, for some parameter $e < 2$ when $p < 1/2$. The evolution of this parameter e in function of p is shown in Figure 1.

Another relevant sparse distribution would be the "low-weight" distribution, where bitstrings from $\{0,1\}^n$ of a given small weight are drawn uniformly at random. For lack of space, we do not consider this (quite different) problem in this paper. Our main contribution, the algorithm described in section 4, cannot be adapted to this other setting.

The algorithm presented in this paper has no concrete application that we are aware of, either in cryptography or elsewhere. We nevertheless demonstrate its practical efficiency by obtaining the —non-trivial and "sensible"— result shown in Section 5.

2 Preliminaries

Let $\mathbf{x} = x_0 x_1 \dots x_{n-1}$ be an n-bit string (we use "bit string" and "vector" as synonyms). We denote by $\mathrm{wt}(\mathbf{x})$ its Hamming weight. Let A be a list ; $|A|$ denotes the number of elements in A and $A[i]$ denotes the i-th element. We say that $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a *3XOR triplet* if $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0$.

We consider random instances of the problem defined as follows. Let Ψ be some probability distribution over $\{0,1\}^n$. Let A, B and C be three lists of N elements drawn independently at random according to Ψ. A *solution* to the instance of the 3XOR problem given by (A, B, C) is a 3XOR triplet $(\mathbf{x}, \mathbf{y}, \mathbf{z}) \in A \times B \times C$.

Let $0 < p < 1/2$ be fixed and let Ber_p the Bernoulli distribution of parameter p. In the sequel, we focus on the sparse "low-density" distributions over $\{0,1\}^n$, denoted as $\mathcal{D}$, where each input bit is drawn independently from Ber_p.

Bounds for Binomial Distributions. Let $\mathcal{B}(n,p)$ denote the binomial distribution with parameters n, p. Let $X \sim \mathcal{B}(n,p)$ be a binomial random variable. We make heavy use of tail bounds, notably the Chernoff bound (1) and the tighter classical inequality (2), a proof of

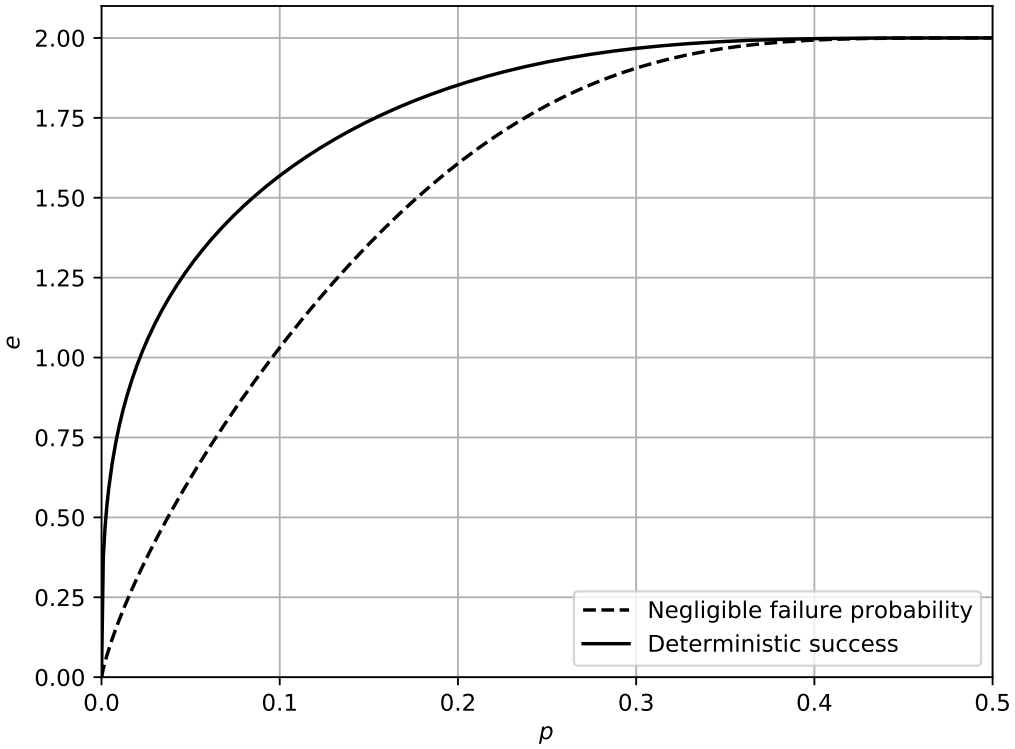

Figure 1 The IncrementalXOR algorithm of section 4 run in time $\mathcal{O}(N + N^e)$ where e is shown here. With $w_{\max} = 2nu$ (cf. theorem 3), this yields the "negligible failure probability" curve. With $w_{\max} = n$ (theorem 6), this yields the "deterministic success" curve.

which can be found in [1] amongst others.

$$\Pr(X \leq an) \leq \exp -\frac{n}{2p}(p-a)^2, \quad \text{if } \frac{k}{n} < p \tag{1}$$

$$\Pr(X \leq an) \leq \exp -nD(a,p) \quad \text{if } a < p, \tag{2}$$

$$\Pr(X \geq an) \leq \exp -nD(a,p) \quad \text{if } a > p,$$

where $D(a,p) = a \ln \frac{a}{p} + (1-a) \ln \frac{1-a}{1-p}$ is the Kullback-Leibler divergence between an a-coin and a p-coin.

Computational Model. We consider a transdichotomous word Random Access Machine (word-RAM) model. In this model, we have access to a machine in which each "memory cell" contains a n-bit word. We assume that the usual arithmetic and bit-wise operations on n-bit words, as well as the comparison of two n-bit integers and memory access with n-bit addresses can be done in constant time. We also assume that obtaining the Hamming weight of an n-bit word is an elementary operation. In other terms, we assume that the machine is large enough to accommodate the instance of the problem at hand.

The Quadratic Algorithm. The simplest possible way to solve the 3XOR problem is the quadratic algorithm (shown as algorithm 1). For all pairs $(x,y) \in A \times B$, check if $x \oplus y$ belongs to C. Each test can be done in constant time if all entries of C have been stored in a

suitable data structure beforehand — for instance one could use the optimal static dictionary of [5], or simply cuckoo hashing [12].

Algorithm 1 The simple quadratic algorithm for 3XOR.

```
function QuadraticSetup(C)
    Initialize a static dictionary 𝒞 containing all the bit strings in C
    return 𝒞
function Quadratic3XOR(A, B, 𝒞)
    for (x, y) ∈ A × B do
        if x ⊕ y ∈ 𝒞 then
            return (x, y, x ⊕ y)
    return ⊥
```

The initialization of the dictionary holding C is separated from the rest, because we will subsequently invoke Quadratic3XOR many times with the same C. We assume (using [5] for instance) that QuadraticSetup takes time $\mathcal{O}(|C|)$. Then Quadratic3XOR$(A, B, \mathcal{C})$ takes time $\mathcal{O}(|A| \times |B|)$. The quadratic algorithm works regardless of the sparsity of the input, and as such it does not take advantage of it. It is the only solution, up to logarithmic factors, when the input is dense. In our case, this is the baseline that must be improved upon.

3 Bounds on the Existence of 3XOR Triplets

Suppose that $(G, +)$ is a group (in additive notation) and assume that three lists A, B and C, each of size N, are made of group elements sampled independently at random according to some distribution Ψ. Let Y be the random variable that counts the number of triplets $(x, y, z) \in A \times B \times C$ such that $x + y + z = 0$ (taken over the random choice of A, B and C). The expected value of Y is easy to compute, and because the average value is bounded away from zero, a concentration bound would yield the probability that the input lists contain at least a single "good triplet" (that sums to zero).

The problem is that the N^3 triplets in $A \times B \times C$ are not independent and thus classical techniques such as the Chernoff bound do not apply. This dependence has to be accounted for. Let x, y, z, u, v denote five independent random variable distributed according to Ψ, and set

$$\rho = \Pr(x + y + z = 0)$$
$$\sigma = \Pr(x + y + z = 0 \mid u + v + z = 0)$$
$$\tau = \Pr(x + y + z = 0 \mid u + y + z = 0)$$

The following result can be specialized for any group G and any distribution Ψ by computing ρ, σ and τ.

▶ **Lemma 1.** $\mathrm{E}\, Y = N^3\rho$ *and* $1 - N^3\rho \leq \Pr(Y = 0) \leq \frac{1}{\rho}\left(\frac{3\sigma}{N} + \frac{3\tau}{N^2} + \frac{1}{N^3}\right)$.

Proof. Let $X(i, j, k)$ denote the binary random variable that takes the value 1 if and only if $A[i] + B[j] + C[k] = 0$ (and zero otherwise), so that $Y = \sum X(i, j, k)$. Unless mentioned otherwise, in this section all sums are taken over $0 \leq i, j, k < N$; we omit the indices to alleviate notations.

The expected value of Y is easy to determine. Because the elements of the lists are identically distributed, $\Pr(A[i] + B[j] + C[k] = 0)$ is independent of i, j and k and its value is ρ. We get:

$$\mathrm{E}\, Y = \mathrm{E} \sum X(i,j,k) = \sum \mathrm{E}\, X(i,j,k) = \sum \Pr(A[i] + B[j] + C[k] = 0) = N^3\rho.$$

The Markov bound yields $1 - \mathrm{E}\, Y \leq \Pr(Y = 0)$; the less immediate part consists in estimating $\Pr(Y > 0)$. Because Y is the sum of binary random variables, we are entitled to use Ross's conditional expectation inequality [13]:

$$\Pr(Y > 0) \geq \sum \frac{\mathrm{E}\,(X(i,j,k))}{\mathrm{E}\,(Y \mid X(i,j,k) = 1)}.$$

As argued above, the value of the term under the sum is independent of i, j and k, so this boils down to: $\Pr(Y > 0) \geq EY/E(Y \mid X(0,0,0) = 1)$. It remains to compute the expected number of solutions knowing that there is at least one. This yields:

$$E(Y \mid X(0,0,0) = 1) = \sum \Pr(A[i] + B[j] + C[k] = 0 \mid A[0] + B[0] + C[0] = 0)$$

We split this sum in 8 parts by considering separately the situation where $i = 0$, $j = 0$ and $k = 0$ (resp $\neq 0$ for each summation index). We introduce the shorthand $p_{ijk} = \Pr\,(A[i] + B[j] + C[k] = 0 \mid A[0] + B[0] + C[0] = 0)$ and we assume that $i, j, k > 0$. Then the two events are in fact independent and $p_{ijk} = \rho$. But when at least one index is zero, this is no longer the case ; the extreme situation is $p_{000} = 1$. By symmetry between the three input lists, we find that $p_{ij0} = p_{i0k} = p_{0jk}$ (this is the value we denote by σ) and $p_{i00} = p_{0j0} = p_{00k}$ (this is the value we denote by τ). We can now write:

$$\begin{aligned}
\mathrm{E}\,(Y \mid X(0,0,0) = 1) &= (N-1)^3\rho + 3(N-1)^2\sigma + 3(N-1)\tau + 1 \\
&= N^3\rho + 3N^2\sigma + 3N\tau + 1 - \Delta \\
\text{with} \qquad \Delta &= (3N^2 - 3N + 1)\rho + 3(2N-1)\sigma + 3\tau
\end{aligned}$$

The "error term" Δ is always positive for $N \geq 1$. Going back to the beginning, we have:

$$\Pr\,(Y > 0) \geq \frac{N^3\rho}{N^3\rho + 3N^2\sigma + 3N\tau + 1 - \Delta} \quad \geq \frac{1}{1 + 3N^{-1}\sigma/\rho + 3N^{-2}\tau/\rho + N^{-3}/\rho}$$

Using the convexity of $1/(1+x)$, we obtain $\Pr\,(Y = 0) \leq 3N^{-1}\sigma/\rho + 3N^{-2}\tau/\rho + N^{-3}/\rho$. ◀

Low-Density 3XOR. We now specialize the result of lemma 1 to the group $(\{0,1\}^n, \oplus)$ with the low-density distribution $\mathcal{D}$ (each bit is drawn independently at random according to the Bernoulli distribution Ber_p of parameter $p < 1/2$).

▶ **Theorem 2.** *Let Y denote the random variable counting the number of 3XOR triplets in $A \times B \times C$. The probability that a random triplet from $A \times B \times C$ XORs to zero is $\rho = (1-p)^n(1-2p+4p^2)^n$, and their expected number is $\mathrm{E}\, Y = N^3\rho$. Furthermore:*

$$1 - \mathrm{E}\, Y \leq \Pr\,(Y = 0) \leq \frac{3}{(\mathrm{E}\, Y)^{1/3}} + \frac{3}{(\mathrm{E}\, Y)^{2/3}} + \frac{1}{\mathrm{E}\, Y}. \tag{3}$$

Proof. If a, b and c are random bits drawn according to Ber_p, then the probability that they XOR to zero is $(1-p)\left(1-2p+4p^2\right)$. It follows that if $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ are drawn according to $\mathcal{D}$, then $\rho = \Pr(\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0) = (1-p)^n(1-2p+4p^2)^n$.

Let us compute $\sigma = \Pr(\mathbf{x} \oplus \mathbf{y} = \mathbf{z} \mid \mathbf{u} \oplus \mathbf{v} = \mathbf{z})$. What happens essentially depends on the hamming weight of $\mathbf{z}$. Both $\mathbf{x}, \mathbf{y}$ and $\mathbf{u}, \mathbf{v}$ have to XOR to $\mathbf{z}$. Two random bits drawn according to Ber_p XOR to zero with probability $p^2 + (1-p)^2$ and their XOR to one with probability $2p(1-p)$. This yields (using the binomial theorem):

$$\begin{aligned}
\rho\sigma &= \Pr(\mathbf{x} \oplus \mathbf{y} = \mathbf{z} \wedge \mathbf{u} \oplus \mathbf{v} = \mathbf{z}) \\
&= \sum_{k=0}^{n} \Pr(wt(\mathbf{z}) = k) \Pr(\mathbf{x} \oplus \mathbf{y} = \mathbf{z} \wedge \mathbf{u} \oplus \mathbf{v} = \mathbf{z} \mid wt(\mathbf{z}) = k) \\
&= \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} \left[2p(1-p)\right]^{2k} \left[p^2 + (1-p)^2\right]^{2(n-k)} \\
&= \left[(1-p)(1-4p+8p^2-4p^3)\right]^n
\end{aligned}$$

We move on to $\tau = \Pr(\mathbf{x} = \mathbf{u} \oplus \mathbf{v} \mid \mathbf{z} = \mathbf{u} \oplus \mathbf{v})$. In this context, this mostly depends on the hamming weight of $\mathbf{u} \oplus \mathbf{v}$. This yields (again using the binomial theorem):

$$\begin{aligned}
\rho\tau &= \Pr(\mathbf{x} = \mathbf{u} \oplus \mathbf{v} \wedge \mathbf{z} = \mathbf{u} \oplus \mathbf{v}) \\
&= \sum_{k=0}^{n} \Pr(wt(\mathbf{u} \oplus \mathbf{v}) = k) \Pr(\mathbf{x} = \mathbf{u} \oplus \mathbf{v} \wedge \mathbf{z} = \mathbf{u} \oplus \mathbf{v} \mid wt(\mathbf{u} \oplus \mathbf{v}) = k) \\
&= \sum_{k=0}^{n} \binom{n}{k} \left[2p(1-p)\right]^k \left[p^2 + (1-p)^2\right]^{n-k} \left[p^k (1-p)^{n-k}\right]^2 \\
&= \left[(1-p)(1-3p+4p^2)\right]^n
\end{aligned}$$

We now move on to establish (3). Because $N = (\mathrm{E}\, Y)^{1/3}/\rho$, the bound of lemma 1 can be rewritten as:

$$\Pr(Y = 0) \leq \frac{3}{(\mathrm{E}\, Y)^{1/3}} \frac{\sigma}{\rho^{2/3}} + \frac{3}{(\mathrm{E}\, Y)^{2/3}} \frac{\tau}{\rho^{1/3}} + \frac{1}{\mathrm{E}\, Y}.$$

We now claim that $1/2 \leq \sigma^{3/n}/\rho^{2/n} \leq 1$ and $1/4 \leq \tau^{3/n}/\rho^{1/n} \leq 1$ when $0 \leq p \leq 1/2$; this yields the desired result. This claim follows from the facts that both values are decreasing functions of p. This can be seen by computing their derivatives (all factors are easily seen to be positive when $0 \leq p \leq 1/2$):

$$\begin{aligned}
\frac{\partial}{\partial p} \frac{\sigma^{3/n}}{\rho^{2/n}} &= -6 \frac{(4p^3 - 8p^2 + 4p - 1)^2 (2p^2 - 6p + 3)(1-2p)^2 p}{(4p^2 - 2p + 1)^6 (1-p)^3} \\
\frac{\partial}{\partial p} \frac{\tau^{3/n}}{\rho^{1/n}} &= -6 \frac{(4p^2 - 3p + 1)^2 (4p^2 - 6p + 3)(1-2p) p}{(4p^2 - 2p + 1)^5 (1-p)^2}
\end{aligned}$$

◀

4 An Algorithm for Random Low-Density 3XOR

In this section, we present a simple algorithm that solves the low-density 3XOR problem with interesting theoretical guarantees when p is small. It is always subquadratic but its most striking feature is that it succeeds with overwhelming probability in linear time when p is small enough.

First of all, because the input is random, the problem may potentially be easy to solve with low error probability without even observing the input lists, depending on n, the size and

the distribution of the input. In light of theorem 2, we see that if N (the size of input lists) is exponentially smaller than $\rho^{-1/3}$, then "**return** $\perp$" is an algorithm that has an exponentially small probability of yielding false negatives. Alternatively, if N exponentially larger than $(1-p)^{-n}$, then we expect the string $000\ldots0$ to be present in A, B and C with overwhelming probability ; it follows that "**return** $(0,0,0)$" is a fast algorithm with exponentially low false positive probability. To avoid these pitfalls, our main focus is on the hard case where $N \approx \rho^{-1/3}$, and where the expected number of solutions in the random input lists is close to one.

Let a, b and c be random bits drawn according to Ber_p conditioned to $a \oplus b \oplus c = 0$. The possible combinations are 000, 011, 101 and 110. We find that

$$u := \Pr(abc = 110 \mid a \oplus b \oplus c = 0) = p^2/(1 - 2p + 4p^2)$$

The same result is attained for 101 and 011. Two out of the three non-zero options result in a 1 bit, and therefore $\Pr(a = 1 \mid a \oplus b \oplus c = 0) = 2u$. It follows that if the $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a triplet drawn from $\mathcal{D}$ such that $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0$, then the expected "density" of $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ is $2u$. This is always smaller than p when $0 < p < 1/2$. In other terms: *random triplets drawn from $\mathcal{D}$ have density p, but random* 3XOR *triplets drawn from $\mathcal{D}$ have smaller density.*

This observation can be exploited in a simple way: we iteratively search for 3XOR triplets of increasing maximum weight. This yields the Incremental3XOR function (shown as algorithm 2). It takes an additional argument $w_{\max}$ controlling the maximum allowed weight. $w_{\max} = 2nu$ yields an algorithm that reveals a 3XOR triplet present in the input with probability greater than 1/4, because the median weight of both $\mathbf{x}$ and $\mathbf{y}$ is $2nu$ if $\mathbf{x} \oplus \mathbf{y} \oplus \mathbf{z} = 0$. Setting $w_{\max} = 2nu(1+\epsilon)$ is enough to miss an existing solution with exponentially small probability. With $w_{\max} = n$, the algorithm deterministically finds a solution if it is present in the input, but the complexity is much higher.

Algorithm 2 An "incremental" algorithm for the sparse 3XOR problem.

```
function Incremental3XOR-One-Sided(A, B, C, w_max)
    for 0 ≤ j ≤ w_max do
        for 0 ≤ i ≤ j do
            ζ ← Quadratic3XOR(A_i, B_j, C)
            if ζ ≠ ⊥ then return ζ
    return ⊥
function Incremental3XOR(A, B, C, w_max)
    for 0 ≤ i ≤ w_max do                    ▷ Bucket sort by Hamming weight
        A_i ← {x ∈ A | wt(x) = i}
        B_i ← {y ∈ B | wt(y) = i}
    ζ_1 ← Incremental3XOR-One-Sided(A, B, C, w_max)
    ζ_2 ← Incremental3XOR-One-Sided(B, A, C, w_max)
    if both ζ_1 = ⊥ and ζ_2 = ⊥ then return ⊥ ; otherwise return the solution found.
```

It is fairly obvious that `Incremental3XOR-One-Sided` only finds a 3XOR triplet $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ in the input lists if $wt(\mathbf{x}) \leq wt(\mathbf{y}) \leq w_{\max}$. By calling it twice, we lift the restriction that $\mathbf{x}$ must be sparser than $\mathbf{y}$.

4.1 Overwhelming Success Probability

Choosing a small value of $w_{\max}$ leads to smaller "filtered" instances and thus to a smaller running time for the actual computation using the quadratic algorithm. However, if $w_{\max}$ is

too small, then a potential solution present in the input might be discarded.

A sensible choice consists in picking $w_{\max}$ in the range $]2nu, np[$ — above the expected density of random 3XOR triplets so that we do not discard them, and below the density of the input lists in order to actually discard input vectors that are too heavy. In this case the algorithm succeeds with overwhelming probability as long as the original input contains at least one solution. The main contribution of this paper is the following

▶ **Theorem 3.** *Let* $e := 2 + 6D\left(2p^2/(1-2p+4p^2), p\right) / \ln(1-p)(1-2p+4p^2)$. *For all* $d > e$ *there is an algorithm for the random low-density 3XOR problem that runs in time* $\mathcal{O}\left(N + N^d\right)$, *where* N *denotes the size of the input list and fails with negligible probability (in* n*).*

We first discuss some aspects of the result. The graph of the best possible exponent (e) is shown in Fig. 1. The exponent e increases from zero to 2 as p goes from zero to $1/2$. Using the bisection algorithm, we find that $e \leq 1$ when $p \leq 0.0957$. It follows that IncrementalXOR is *linear* in the size of the input for small p.

It seems that $e \leq 2p(1 - 2\ln p)$; establishing this is a fascinating endeavour that we must regrettably leave for future work. It is worth noting that the not-so-friendly expression of e comes from the use of the tight binomial bound (2). It would be greatly simplified had we instead used the simpler Chernoff bound (1). However, doing so makes the result much worse for small p: it results in $\lim e = 1$ when p goes to zero (instead of $\lim e = 0$).

▶ **Lemma 4.** *With* $w_{\max} = 2nu(1+\epsilon)$, *if the input contains a 3XOR triplet, then* Incremental3XOR *returns* $\perp$ *with probability less than* $2\exp(-nu\epsilon^2)$.

Proof. Assume that the input lists contain a 3XOR triplet $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*)$. It will be discarded if and only if the weight of either $\mathbf{x}^*, \mathbf{y}^*$ is greater than $w_{\max}$. We know that the expected weight of $\mathbf{x}^*$ and $\mathbf{y}^*$ (and $\mathbf{z}^*$ as well but this is irrelevant) is $2un$, therefore the Chernoff bound (1) shows that either has weight greater than $2un(1+\epsilon)$ with probability less than $\exp(-nu\epsilon^2)$. A union bound (for $\mathbf{x}^*$ and $\mathbf{y}^*$) then ensures that the solution is discarded with probability less than $2\exp(-nu\epsilon^2)$. ◀

▶ **Lemma 5.** *Let* T *denote the running time of* Incremental3XOR *with* $w_{\max} = 2nu(1+\epsilon)$. *Then* $\mathrm{E}\, T \leq N + N^2 \exp(-2nD\left(2u(1+\epsilon), p\right))$.

Proof. In the sequel, all the stated complexities must be understood "up to a constant factor". Bucket-sorting the input lists by Hamming weight takes time N. Up to a constant factor, it is sufficient to upper-bound the running time of Incremental3XOR-One-Sided. The running time of Quadratic3XOR on input $(A_i, B_j, \mathcal{C})$ is upper-bounded by $|A_i| \cdot |B_j|$. Therefore, the total running time of Incremental3XOR is then:

$$T = N + \sum_{j=0}^{w_{\max}} \sum_{i=0}^{j} |A_i| \cdot |B_j| \leq N + \left(\sum_{j=0}^{w_{\max}} |B_j|\right)\left(\sum_{i=0}^{w_{\max}} |A_i|\right)$$

Let $\mathfrak{A} = \bigcup_{i=0}^{w_{\max}} A_i$ and $\mathfrak{B} = \bigcup_{j=0}^{w_{\max}} B_i$ denote the lists of all input vectors of weight less than of equal to $w_{\max}$. The above inequality shows that the running-time of the algorithm is less than that of running Quadratic3XOR directly on $\mathfrak{A}$ and $\mathfrak{B}$.

Let $X \sim \mathcal{B}(n, p)$ be a (binomial) random variable modeling the weight of an input vector of density p. Such a vector belongs to $\mathfrak{A}$ or $\mathfrak{B}$ if its is less than or equal to $w_{\max}$, and this happens with probability $s := \Pr(X \leq w_{\max})$. Because $w_{\max} < np$, the binomial tail bound (2) yields the tight upper-bound $s \leq e^{-nD(2u(1+\epsilon),p)}$.

The sizes of $\mathfrak{A}$ and $\mathfrak{B}$ are stochastically independent random variables following a binomial distribution of parameters N, s and their expected size is Ns. The expected running time of the quadratic algorithm on $\mathfrak{A}$ and $\mathfrak{B}$ (which is an upper-bound on that of IncrementalXOR) is therefore $\mathrm{E}\,|\mathfrak{A}| \times \mathrm{E}\,|\mathfrak{B}| = \mathrm{E}\,|\mathfrak{A}| \times \mathrm{E}\,|\mathfrak{B}| = N^2 s^2$. Combining this with the upper-bound on s gives the announced result. ◀

We are now ready to prove theorem 3.

proof of theorem 3. Let d be a complexity exponent greater than the bound e given in theorem 3. Let $\epsilon > 0$ be such that

$$d = 2 + 6\frac{D\left(2u(1+\epsilon), p\right)}{\ln(1-p)(1-2p+4p^2) - 3\epsilon \ln 2}.$$

(such an ϵ always exist). Note that setting $\epsilon = 0$ in this expression yields the lower-bound exponent e of the theorem.

Let $N_0 := \rho^{-1/3}$ (input lists of this size contain a single 3XOR triplet in average). Suppose that $N \leq N_0 2^{\epsilon n}$, where N denotes the size of the input lists ; in this case run IncrementalXOR with $w_{\max} = 2un(1+\epsilon)$. Lemma 4 guarantees the exponentially small failure probability while lemma 5 tells us that the expected running time T is less than $N + N^2 \exp -nD(2u(1+\epsilon), p)$.

Set $d' := \log_N(\mathrm{E}\,T - N)$, so that the algorithm runs in time $\mathcal{O}\left(N + N^{d'}\right)$. A quick calculation shows that $d' = d$, and the theorem is proved in this case.

If $N > N_0 2^{n\epsilon}$, then do the following : slice the input lists in chunks of size $4N_0$ and run IncrementalXOR with $w_{\max} = 2un(1+\epsilon)$ on each successive chunk until a solution is found. Theorem 2 tells us that each chunk contain 64 3XOR triplets on average, and contain at least one 3XOR triplet with probability greater than $3/64$. IncrementalXOR will reveal an existing 3XOR triplet present in a chunk with probability greater than $1/4$ (this is true regardless of the value of $\epsilon > 0$). Therefore, a 3XOR triplet will be found in each chunk with probability greater than $3/256$; because the the chunks are completely independent parts of the random input, the events "IncrementalXOR finds a 3XOR triplet in chunk i" are independent. Therefore, the whole process fails to reveal a 3XOR triplet with probability less than $\left(1 - \frac{3}{256}\right)^{n\epsilon}$. The running time is $4N \cdot N_0^{d-1}$, which is less than N^d. ◀

4.2 Deterministic Success

We now study the expected behavior of IncrementalXOR algorithm with $w_{\max} = n$. When $w \geq np$, then discarding vectors of weight greater than w does not reduce significantly the size of the lists. Thus, all iterations with "threshold weight" $w \geq np$ cost essentially $\Omega\left(N^2\right)$. It follows that if there is no 3XOR triplet in input lists (for instance, if they are too short), then the algorithm is essentially quadratic. On the other hand, if the input lists are so long that they contain many 3XOR triplets, then the probability that they contain a low-weight triplet (and thus that the algorithm stops early) increases. We therefore focus on the hardest relevant case, namely input lists of size $N = P(n)\rho^{-n/3}$, where P is a non-constant positive polynomial. The input lists contain on average $P(n)^3$ 3XOR triplets and thanks to theorem 2, they contain a 3XOR triplet with probability greater than $1 - 1/P(n)(1 + o(1))$. The crux of the analysis is that the w-th iteration is done if and only if the solution has Hamming weight greater than or equal to w. The expected weight of the solution is $2un < 2np$, and therefore the probability that the most expensive iterations take place is exponentially small.

▶ **Theorem 6.** *Consider input lists of size $N = P(n)\rho^{-n/3}$ and assume that they contain a 3XOR triplet. Then* INCREMENTAL3XOR *with $w_{\max} = n$ returns a valid solution and terminates in time $\mathcal{O}(N + N^e)$, where*

$$e = 2 + 3\frac{2D\left(\frac{1}{1+\sqrt[3]{\left(\frac{1}{p}-1\right)^2\left(\frac{1}{2u}-1\right)}}, p\right) + D\left(\frac{1}{1+\sqrt[3]{\left(\frac{1}{p}-1\right)^2\left(\frac{1}{2u}-1\right)}}, 2u\right)}{\ln(1-p)(1-2p+4p^2)}.$$

The graph of e is also shown in Fig. 1. We find again that $e \leq 1$ when $p \leq 0.02155$. Thus, this unfailing algorithm is also linear for small p. It is worthwhile noting that the complexity is significantly higher than when $w_{\max} = 2un$. The proof is potentially less tight, but it seems plausible that there is a "heavy tail" effect. Theorem 6 follows from the following

▶ **Lemma 7.** *Let T denote the running time of* INCREMENTAL3XOR *with $w_{\max} = n$. Then*

$$\mathrm{E}\,T \leq N + 4nN^2 e^{-n[2D(x_0,p)+D(x_0,2u)]} \qquad \textit{where} \qquad \frac{1}{x_0} - 1 = \sqrt[3]{\left(\frac{1}{p}-1\right)^2\left(\frac{1}{2u}-1\right)}.$$

Proof. Let T_j denote the running time of the j-th iteration of the outer **for** loop and S the number of iterations done when the algorithm stops (i.e. the value of j in the algorithm). As in the proof of lemma 5, let $\mathfrak{A}_k = \bigcup_{i=0}^{k} A_i$ and $\mathfrak{B}_k = \bigcup_{j=0}^{k} B_j$ and we find that $T_j \leq |\mathfrak{A}_j| \times |\mathfrak{B}_j|$.

The total running time is then given by $T = \sum_{j=0}^{n} [S \geq j] T_j$. The two random variables $[S \geq j]$ and T_j are not independent, however we claim that $\mathrm{E}\left([S \geq j] T_j\right) \leq \left(\mathrm{E}\,[S \geq j]\right) T_j$ — *i.e.* they are negatively correlated. The point is that is that the fact that input lists contain a "large weight" triplet t^* can only reduce the expected size of low-weight lists by at most one, and therefore reduce the expected time needed to process them. It follows that $\mathrm{E}\,T \leq \sum_{j=0}^{n} \Pr(S \geq j) \cdot (\mathrm{E}\,T_j)$.

Next, let us consider two random variables following binomial distributions $X_{2u} \sim \mathcal{B}(n, 2u)$ and $X_p \sim \mathcal{B}(n, p)$. From the proof of lemma 5, we know that $\mathrm{E}\,T_j \leq N^2 s^2$, where $s = Pr(X_p \leq j)$. In addition, following the same reasoning as in the proof of lemma 4, we have $\Pr(S \geq j) \leq 2\Pr(X_{2u} \geq j)$. This gives:

$$\mathrm{E}\,T \leq 2N^2 \sum_{j=0}^{n} \Pr(X_p \leq j)^2 Pr(X_{2u} \geq j)$$

Set $\mu_j := \Pr(X_p \leq j)^2 Pr(X_{2u} \geq j)$. Our goal is to upper-bound the sum of the μ_j's. To this end, we split the sum in three parts:

$$\sum_{j=0}^{n} \mu_j \leq \sum_{j=0}^{2nu} \mu_j + \sum_{j=2nu}^{np} \mu_j + \sum_{j=np}^{n} \mu_j$$

First, we focus on the case where $0 \leq j \leq 2nu$. In this range, $\Pr(X_p \leq j)$ is increasing with j and $Pr(X_{2u} \geq j)$ is greater than $\frac{1}{2}$, therefore we find that $\mu_j \leq 2\mu_{2nu}$ for $0 \leq j \leq 2nu$. A symmetric argument shows that $\mu_j \leq 2\mu_{np}$ for all $np \leq j \leq n$.

Let M denote the largest μ_j for $2nu \leq j \leq np$. It follows from the above discussion that this is the largest of all the μ_j, so that their is less than $2nM$. We use the binomial tail bound (2) to get an upper-bound on M. Set $f(x) = 2D(x,p) + D(x,2u)$, so that $\mu_j \leq e^{-nf\left(\frac{j}{n}\right)}$ when $2nu \leq j \leq np$. We next seek the maximum of f, and for this we compute its derivative:

$$f'(x) = 2\log\frac{x}{p} - 2\log\frac{1-x}{1-p} + \log\frac{x}{2u} - \log\frac{1-x}{1-2u}$$

Solving $f'(x_0) = 0$ reveals only one possible real solution, that satisfies:

$$\frac{1}{x_0} - 1 = \sqrt[3]{\left(\frac{1}{p} - 1\right)^2 \left(\frac{1}{2u} - 1\right)}$$

It appears that $1/x_0 - 1$ is the geometric mean of $1/p - 1, 1/p - 1$ and $1/(2u) - 1$. This implies in particular $2u \leq x_0 \leq p$ as expected. ◀

5 A "Sensible" Application

In order to put this algorithm to the test, we forged an artificial instance of the problem. We downloaded an XML dump of the DBLP database, and extracted all articles published in a few selected cryptography conferences (CRYPTO, EUROCRYPT, ASIACRYPT, FSE, PKC, CHES, SAC) as well as two journals (TOSC, TCHES). This made more than 7700 articles. For each article, we wrote down one line of text with the authors and title.

We considered the function (where & denotes the bitwise AND):

$$F(x_1, x_2, x_3, x_4) = \texttt{SHA-512}(x_1) \;\&\; \texttt{SHA-512}(x_2) \;\&\; \texttt{SHA-512}(x_3) \;\&\; \texttt{SHA-512}(x_4).$$

This yields 512-bit hashes with expected density $1/16$. `SHA-512` is a secure cryptographic hash function, so we assumed that there was no way to find inputs leading to correlated outputs besides brute force. We looked for three quadruplets of articles such that

$$F(x_1, x_2, x_3, x_4) \oplus F(y_1, y_2, y_3, y_4) \oplus F(z_1, z_2, z_3, z_4) = 0$$

With the additional constraint that all articles are distinct. There are 5775 ways to dispatch 12 items into 4 indistinguishable urns, so that with our 7700 articles, we can assemble $2^{138.5}$ bundles of 12 publications having a chance to satisfy the above equation (of course the inputs are correlated, and this deviates from the original problem formulation, but this is not a serious issue). Alternatively, we may form 2^{47} quadruplets of publication. With $p = 1/16$ and $n = 512$, we could then expect about 40 solutions from out data set. This made us confident that we there would be at least one, but finding it does not seem so easy at first glance.

Evaluating F on all the available quadruplets is not a problem (it takes 240 CPU-hours). Trouble starts when we considered writing the list of 2^{47} hashes to persistent storage: this would require more than 9 petabytes — this is a lot, but some computing centers have that much. However, finding the "golden" triplet of quadruplets using the quadratic algorithm would then require 2^{94} probes into a large hash table, and given mankind's present computing power, this does not seem feasible before the sun turns into a red giant.

Exploiting the sparsity of the input turns the problem into a walk in the park. The expected weight of 3XOR triplets of density $1/16$ is ≈ 2.25. We evaluated F on all quadruplets, but kept only the hashes with hamming weight less than or equal to 3. We thus kept 5091 candidate quadruplets, for a total storage size of 358KB. We then searched for solutions in this restricted data set using the quadratic algorithm. This required 25 millions probes in a hash table and was very fast.

We found six solutions, one of which is shown as algorithm 3. Amazingly, it contains the name of one of the authors of the present article.

6 Conclusion

We presented a simple algorithm for the random sparse "low-density" 3XOR problem, which is always subquadratic and can even be linear if the density is low enough. It works by reducing

Algorithm 3 Demonstrating a sensible application of sparse 3XOR algorithms.

```
from hashlib import sha512

# FSE 2011
a = "Simon Knellwolf and Willi Meier: Cryptanalysis of the Knapsack Generator. (2011)"

# ASIACRYPT 2017
b = "Ran Canetti, Oxana Poburinnaya and Mariana Raykova: Optimal-Rate Non-Committing Encryption. (2017)"

# CRYPTO 2019
c = "Muhammed F. Esgin, Ron Steinfeld, Joseph K. Liu and Dongxi Liu: Lattice-Based Zero-Knowledge \
Proofs: New Techniques for Shorter and Faster Constructions and Applications. (2019)"

# FSE 2009
d = "Martijn Stam: Blockcipher-Based Hashing Revisited. (2009)"

# EUROCRYPT 1990
e = "Cees J. A. Jansen: On the Construction of Run Permuted Sequences. (1990)"

# EUROCRYPT 2013
f = 'Charles Bouillaguet, Pierre-Alain Fouque and Amandine Véber: Graph-Theoretic Algorithms for the \
"Isomorphism of Polynomials" Problem. (2013)'

# CRYPTO 2017
g = "Prabhanjan Ananth, Arka Rai Choudhuri and Abhishek Jain: A New Approach to Round-Optimal Secure \
Multiparty Computation. (2017)"

# EUROCRYPT 2001
h = "William Aiello, Yuval Ishai and Omer Reingold: Priced Oblivious Transfer: How to Sell Digital \
Goods. (2001)"

# CRYPTO 2019
i = "Navid Alamati, Hart Montgomery and Sikhar Patranabis: Symmetric Primitives with Structured \
Secrets. (2019)"

# CRYPTO 2019
j = "Shweta Agrawal, Monosij Maitra and Shota Yamada: Attribute Based Encryption (and more) for \
Nondeterministic Finite Automata from LWE. (2019)"

# EUROCRYPT 1986
k = "Christoph G. Günther: On Some Properties of the Sum of Two Pseudorandom Sequences. (1986)"

# CRYPTO 2009
l = "Susan Hohenberger and Brent Waters: Short and Stateless Signatures from the RSA Assumption. (2009)"

def H(s : str) -> int:
    """Returns the hash (SHA-512) of the string s as a 512-bit integer."""
    return int.from_bytes(sha512(s.encode('utf8')).digest(), byteorder='big')

assert (H(a) & H(b) & H(c) & H(d)) ^ (H(e) & H(f) & H(g) & H(h)) ^ (H(i) & H(j) & H(k) & H(l)) == 0
```

an instance of the "low-density" problem to several smaller instances of a "low-weight" problem, whose sparsity is not exploited at this stage. This begs for finding new algorithms for the "low-weight" 3XOR problem, which requires completely different techniques. This other sparse problem is not only interesting in itself, but better algorithms would yield even faster algorithms for the "low-density" case. This is the subject of ongoing and future research.

Ackowledgements We thank Pierre-Alain Fouque, Antoine Joux and Anand Kumar Narayanan for useful discussions. We are very grateful to 3 out of 6 anonymous reviewers (so far) for rejecting two previous versions of this paper while providing extremely helpful feedback and suggesting new ideas.

References

1. R. Arratia and L. Gordon. Tutorial on large deviations for the binomial distribution. *Bulletin of Mathematical Biology*, 51(1):125 – 131, 1989. URL: http://www.sciencedirect.com/science/article/pii/S0092824089800527, doi:https://doi.org/10.1016/S0092-8240(89)80052-7.
2. Leif Both and Alexander May. The approximate k-list problem. *IACR Transactions on Symmetric Cryptology*, 2017(1):380–397, 2017.
3. Charles Bouillaguet, Claire Delaplace, and Pierre-Alain Fouque. Revisiting and improving algorithms for the 3xor problem. *IACR Transactions on Symmetric Cryptology*, 2018(1):254–276, 2018.
4. Martin Dietzfelbinger, Philipp Schlag, and Stefan Walzer. A subquadratic algorithm for 3xor. In Igor Potapov, Paul G. Spirakis, and James Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science, MFCS 2018, August 27-31, 2018, Liverpool, UK*, volume 117 of *LIPIcs*, pages 59:1–59:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.59.
5. Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with o(1) worst case access time. *J. ACM*, 31(3):538–544, June 1984. URL: http://doi.acm.org/10.1145/828.1884, doi:10.1145/828.1884.
6. Anka Gajentaan and Mark Overmars. On a class of $\mathcal{O}(n^2)$ problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.
7. Zahra Jafargholi and Emanuele Viola. 3sum, 3xor, triangles. *Algorithmica*, 74(1):326–343, 2016. doi:10.1007/s00453-014-9946-9.
8. Antoine Joux. *Algorithmic cryptanalysis*. CRC Press, 2009.
9. Gaëtan Leurent and Ferdinand Sibleyras. Low-memory attacks against two-round even-mansour using the 3XOR problem. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 210–235. Springer, 2019. doi:10.1007/978-3-030-26951-7_8.
10. Mridul Nandi. Revisiting Security Claims of XLS and COPA. *IACR Cryptology ePrint Archive*, 2015:444, 2015.
11. Ivica Nikolić and Yu Sasaki. Refinements of the k-tree Algorithm for the Generalized Birthday Problem. In *ASIACRYPT*, pages 683–703. Springer, 2015.
12. Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. In *European Symposium on Algorithms*, pages 121–133. Springer, 2001.
13. S.M. Ross. *Probability Models for Computer Science*. Elsevier Science, 2002. URL: https://books.google.fr/books?id=fG3iEZ8f3CcC.