Ultra-Resilient Superimposed Codes:
Near-Optimal Construction and Applications

Gianluca De Marco*   Dariusz R. Kowalski†

Abstract

A superimposed code is a collection of binary vectors (codewords) with the property that no vector is contained in the Boolean sum of any k others, enabling unique identification of codewords within any group of k. Superimposed codes are foundational combinatorial tools with applications in areas ranging from distributed computing and data retrieval to fault-tolerant communication. However, classical superimposed codes rely on strict alignment assumptions, limiting their effectiveness in asynchronous and fault-prone environments, which are common in modern systems and applications.

We introduce Ultra-Resilient Superimposed Codes (URSCs), a new class of codes that extends the classic superimposed framework by ensuring a stronger codewords' isolation property and resilience to two types of adversarial perturbations: arbitrary cyclic shifts and partial bitwise corruption (flips). Additionally, URSCs exhibit universality, adapting seamlessly to any number k of concurrent codewords without prior knowledge. This is a combination of properties not achieved in any previous construction.

We provide the first polynomial-time construction of URSCs with near-optimal length, significantly outperforming previous constructions with less general features, all without requiring prior knowledge of the number of concurrent codewords, k. We demonstrate that our URSCs significantly advance the state of the art in multiple applications, including uncoordinated beeping networks, where our codes reduce time complexity for local broadcast by nearly two orders of magnitude, and generalized contention resolution in multi-access channel communication.

Keywords: superimposed codes, ultra-resiliency, derandomization, deterministic algorithms, uncoordinated beeping networks, contention resolution.

1   Introduction

Superimposed codes, proposed by Kautz and Singleton in 1964 [21], are a well-known and widely used class of codes represented by a binary $t \times n$ matrix, where columns correspond to codewords. These codes have a distinctive property: for any subset of k columns from the matrix, and for any column **c** within this subset, there exists at least one row where column **c** has an entry of 1 while all other columns in the subset have entries of 0. This feature allows for the unique identification of any single column (codeword) within a subset of columns.

The significance of superimposed codes cannot be overstated. Over the past six decades, these codes have found applications in an impressively wide range of fields. For instance, they have proven instrumental in information retrieval (see e.g. [23, p. 570]), pattern matching [19, 30], learning problems [2, 22], wireless and multi-access communication [34, 20, 5], distributed coloring [26, 29].

Despite their broad applicability, classic superimposed codes have a significant limitation: they require perfectly aligned (i.e., fixed) codewords. For example, in wireless distributed communication, this alignment requires synchronization before the codes can be applied. Similarly, in distributed information retrieval, it necessitates the alignment of file descriptor codewords. However, achieving synchronization is often challenging or impractical in many distributed environments, where entities may start using

---

*University of Salerno, Salerno, Italy (gidemarco@unisa.it)

†Augusta University, Augusta, Georgia, USA (dkowalski@augusta.edu)

codewords at arbitrary times as they wake up and join the system. Dynamic data retrieval also presents difficulties, as it may involve data arriving out of order. The lack of synchronization creates a difficulty in defining a meaningful product of two codes. Other limitations arise from a limited capacity to handle data corruption. Finally, existing codes assume that the number of superimposed codewords, k, is known. However, some applications may not provide this in advance, and standard estimation techniques—such as doubling the estimate of k—are ineffective, as consecutive code estimates can overlap due to arbitrary misalignments of codewords.

The purpose of the present work is to introduce and efficiently construct new superimposed codes that simultaneously address all the above limitations. We provide a near-optimal solution that significantly outperforms prior codes, which were designed to ensure only some or weaker properties, and demonstrate usefulness of new code's properties in various applications.

1.1 Our contributions and comparison with previous works

Before presenting the formal description in Section 2 (Definition 2), we will provide a high-level overview of our codes. We begin with the classic definition of superimposed codes, followed by an outline of the properties of our codes, including both the newly introduced and previously established properties. We will then review the relevant literature related to our codes and compare their performance to prior results. Finally, we will conclude with a discussion on how these codes can enhance performance in pertinent distributed problems, particularly in beeping networks and multiple-access channels, as well as in other contexts.

Definition 1 (Classic definition). *A (k, n)-superimposed code is formally defined as a $t \times n$ binary matrix* $\mathbf{M}$ *such that the following property holds:*

> *($\mathcal{I}$) For any k-tuple of columns of* $\mathbf{M}$ *and for any column* $\mathbf{c}$ *of the given k-tuple, there is a row $0 \le i < t$ such that column* $\mathbf{c}$ *has 1 in row i and all remaining $k-1$ columns of the k-tuple have all 0 in row i.*

The columns of matrix $\mathbf{M}$ *are called* codewords *and their length t is called the* code length.

1.1.1 Ultra-Resilient Superimposed Codes (URSC)

Our newly proposed ultra-resilient superimposed codes represent a substantial generalization of the classic definition, designed to provide robust performance without requiring prior knowledge of the number k of superimposed codes. We start from the following enhanced version of property $\mathcal{I}$, which ensures a better isolation of codewords; consequently, as shown in Section 4 (Algorithm 2 and Lemma 15), it allows to implement a meaningful product of codes even in the presence of arbitrary codeword misalignment and bit corruption.

> **Isolation property:** ($\mathcal{I}'$) For any k-tuple of columns of $\mathbf{M}$ and any column $\mathbf{c}$ of this k-tuple, there exists a row $0 \le i < t$ such that column $\mathbf{c}$ has a 1 in row i, while all other $k-1$ columns of the k-tuple have 0 in rows $(i-1) \bmod t$, i, and $(i+1) \bmod t$.

At a high level, the ultra-resilient capability of our codes simultaneously ensures the following:

(a) **Shift resilience:** Even if each codeword undergoes independent and arbitrary cyclic shifts, property $\mathcal{I}'$ is still preserved. This resilience is particularly effective in uncoordinated settings where computational entities can join the system at arbitrary times (asynchronous activations).

(b) **Flip resilience:** Property $\mathcal{I}'$ remains intact even if a fraction $1 - \alpha$ of unique occurrences of ones in each codeword can be flipped (the classic setting is obtained for $\alpha = 1$). This ensures robust performance under adversarial jamming in dynamic, asynchronous environments.

(c) **Universality:** The construction and application of these codes do not require prior knowledge of the parameter k, the number of concurrently active codewords. This intrinsic feature enables their applicability across dynamic and uncertain settings where k may vary or remain unknown.

Note: Our codes ensure that flip resilience is fully integrated with shift resilience, so that these properties work together to reinforce robustness. Specifically, flip resilience is maintained under arbitrary cyclic shifts, meaning that for any shifted configuration, the isolation property $\mathcal{I}'$ holds even if a fraction $1 - \alpha$ of ones in each codeword is flipped. This combined ultra-resilience provides robust protection in dynamic and adversarial environments, where both positional misalignments and bitwise corruption may occur simultaneously.

In a classic superimposed code (where in particular shift resilience is not required) the isolation property $\mathcal{I}'$ can be ensured by interleaving each row of its matrix $\mathbf{M}$ with a row of 0's. Specifically, for each row i of $\mathbf{M}$, the interleaved matrix $\mathbf{M}'$ has row $2i$ as row i of $\mathbf{M}$ and row $2i + 1$ filled entirely with 0's. However, in our ultra-resilient superimposed codes, where shift resilience has to be ensured, this technique fails as the independent shifts of columns would disrupt this row-by-row alignment, resulting in overlaps that violate $\mathcal{I}'$. Thus, achieving the isolation property with shift resilience requires a much more robust construction.

In classic superimposed codes, the length t of the codewords (Definition 1) serves as a key measure of efficiency and performance. However, in the more complex scenario we address—where universality must be maintained in the presence of arbitrary misalignments of the codewords—this fixed measure becomes impractical (see Section 2 for a more detailed discussion of this challenge). To overcome this limitation, we introduce a more advanced metric, that we call ***code elongation***, that dynamically adjusts to the actual (but unknown) number of superimposed codewords in the system.

Comparative Efficiency and Performance. Our work builds on and extends foundational studies on superimposed codes, enhancing them to address the unique challenges posed by dynamic and asynchronous communication scenarios, fault tolerance, and the absence of knowledge about the number k of superimposed codewords. Our main result, stated in Theorem 1 of Section 3, is as follows:

> *We provide a Las Vegas algorithm that, in polynomial time, constructs an ultra-resilient superimposed code with a near-optimal performance (code elongation) of $O\left(\frac{k^2}{\alpha^{2+\epsilon}} \log n\right)$.*

For classic superimposed codes (i.e., those with aligned codewords and no flip resilience), the best-known existential bound is $O(k^2 \log(n/k))$, established by Erdös, Frankl, and Füredi [15]. Over the years, various proofs have been presented for the corresponding lower bound, including those by D'yachkov and Rykov [14], Ruszinkó [33], and Füredi [16], all converging on an $\Omega(k^2 \log_k n)$ bound. An elegant combinatorial proof of this lower bound has also been provided more recently by Alon and Asodi [2]. Porat and Rothschild [30] made a significant contribution by introducing the first polynomial-time algorithm to construct classic superimposed codes with a near-optimal length of $O(k^2 \log n)$.

Recently, Rescigno and Vaccaro [31] introduced a generalized fault-tolerant version of classic superimposed codes (with flip resilience, but without shift resilience). They presented a randomized construction that achieves an average code length of $O((\frac{k}{\alpha})^2 \log n)$ in polynomial time. Additionally, they demonstrated that any superimposed code supporting flip resilience requires a length of $\Omega\left(\left(\frac{k}{\alpha}\right)^2 \frac{\log n}{\log(\frac{k}{\alpha})}\right)$.

An early extension addressing non-aligned codewords for use in synchronization problems was introduced by Chu, Colbourn, and Syrotiuk [6, 7], who proposed a generalized version of superimposed codes with cyclic shifts and fault tolerance. In terms of our parameters, their construction achieves an efficiency of $O((k \frac{\log n}{\log k})^3)$, assuming a fixed constant α. Another significant contribution in the pursuit of generalizing classic superimposed codes to accommodate non-aligned codewords was achieved recently by Dufoulon, Burman, and Beauquier [13] in the context of asynchronous beeping models. They introduced polynomial-time constructible superimposed codes, called Uncoordinated Superimposed Codes (USI-codes), which effectively handle arbitrary shifts of codewords (without fault tolerance), exhibiting a code length of $O(k^2 n^2)$.

It is important to note that all of the above upper bounds were achieved with knowledge of k, which played a significant supportive role in the design of the respective algorithms. Notably, the lack of knowledge about k presents a challenge for construction, only when shifts are introduced. Otherwise, it

suffices to construct codes tailored to fixed values of k, as is typically assumed in the literature (cf. [30]), and concatenate them for exponentially growing values of k.

With a performance of $O\left(\frac{k^2}{\alpha^{2+\epsilon}}\log n\right)$, our codes significantly outperform all prior results for non-aligned codewords, and they do so in a much more general and challenging dynamic setting. Specifically, our codes provide *a stronger isolation property $\mathcal{I}'$ that accommodates shift resilience, flip resilience, and universality*. This means that each codeword retains isolated 1-bits in a local neighborhood of three adjacent rows, even under arbitrary cyclic shifts, up to a $1 - \alpha$ fraction of bit flips for any $0 < \alpha \le 1$, and without prior knowledge of the parameter k, the number of active codewords. This combination of properties is unprecedented in the field.

Additionally, our codes are the *first* to achieve *near-optimal* performance in the generalization of both shift resilience and flip resilience (fault tolerance), closely adhering to the lower bound established for *fixed, aligned* codewords [31]. Importantly, all prior constructions combining shift and flip resilience performed substantially worse, even without guaranteeing the isolation property or universality.

If we set aside fault tolerance (i.e., by considering $\alpha = 1$), our *polynomial-time constructible* codes achieve a code elongation of $O(k^2 \log n)$, which matches the best existential bound for *classic* superimposed codes by Erdös, Frank and Füredi [15] for all $k = n^{o(1)}$. Notably, our codes achieve this same bound while additionally exhibiting our isolation property with shift and flip resilience and without requiring prior knowledge of the parameter k. Moreover, it is important to note that we also almost match the fundamental lower bound $\Omega(k^2 \log_k n)$, as established in [14, 33, 16, 2], which is valid even for classic superimposed codes (Definition 1)). Also, our construction yields codes with asymptotically the same length $O(k^2 \log n)$ as the best-known polynomially constructible classic superimposed codes [30], *i.e.*, codes requiring codewords to be aligned and without fault tolerance ($\alpha = 1$).

Technical novelty. Rooted in the universality concept of code elongation, our construction is efficiently achieved through a novel de-randomization of a specifically designed stochastic matrix distribution (as defined in Definition 4). We prove that this distribution satisfies a crucial property that we term the Collision Bound Property (Definition 3) with high probability (cf. Lemma 2). This property is essential for translating conditions on subsets of k columns into conditions that apply to pairs of columns, allowing us to verify and construct the ultra-resilient superimposed codes in a computationally efficient manner (cf. Lemma 1). In this way, we dramatically reduce the size of the problem space from considering all k-subsets of columns to focusing on pairs of columns and their possible shifts, see Algorithm 1. This transformation reduces the complexity of the problem to polynomial time in terms of the number of codewords n, making the de-randomization feasible and scalable. Once the problem space is reduced to pairs of columns, we must ensure that all desired code properties (as outlined in Definition 3) are preserved during the de-randomization process. This involves analyzing intervals within codewords and accounting for three types of shifts and rearrangements. The proof of Lemma 2 addresses this in detail; see Section 3.3 for an overview of the challenges and methods involved, and Appendix C for the complete proof.

1.1.2 Applications

The two main applications studied in this paper are within the contexts of Beeping networks and Contention resolution, which are outlined below. We direct the reader to Appendix A for a discussion of many other potential applications. It is important to note that, although the code is obtained by a Las Vegas randomized algorithm, the resulting codewords are fixed (i.e., non-probabilistic) and can be used reliably in deterministic algorithms. Furthermore, the randomized algorithm only needs to be run once, and the generated code can be reused as many times as needed.

Deterministic Neighborhood learning and Local broadcast in beeping networks (Section 4). The beeping model, introduced by Cornejo and Kuhn [9], is a minimalist communication framework where nodes communicate in discrete time slots by either beeping or remaining silent, and the only feedback a node receives is whether there was a beep in the current time slot.

In the context of uncoordinated beeping networks, our novel coding tool significantly improves the time complexity for the local broadcast problem, previously addressed in [13], and related neighborhood learning. The solution in [13] achieved a time complexity of $O(\Delta^4 M)$, where Δ is the maximum node degree and M is the message size.

> *Our approach (see Section 4 and Theorem 3) nearly quadratically reduces the complexity of local broadcast to a deterministic $O(\Delta^2 \log n \cdot (M + \log n))$, where n is the number of nodes.*

This improvement is made possible through our ultra-resilient superimposed codes, which enable more efficient and resilient data transmission even under adversarial jamming. More specifically, shift resilience mitigates the impact of uncoordinated activation, while isolation allows to use a product of the code with specifically designed small pieces of code that carry desired information despite of shifts of the main code.

Deterministic generalized Contention resolution on multi-access channels (Appendix B). In Contention resolution (CR) on a multi-access channel, k stations out of a total ensemble of n may become active, each with a packet that can be transmitted in a single time slot. The objective is to enable each of these k contending stations to successfully transmit its packet, i.e., to transmit without causing a collision with others in the same time slot. The earliest theoretical work on contention resolution dates back over 50 years, primarily with the seminal papers by Abramson [1], Roberts [32], and Metcalfe and Boggs [27]. Since then, CR has developed a long and rich history, addressing areas such as communication tasks, scheduling, fault tolerance, security, energy efficiency, game theory, and more. However, for deterministic solutions, only recently has the problem been studied in the challenging setting of arbitrary activation times, with the best known *existential* upper bound provided by De Marco, Kowalski and Stachowiak [12]. We can apply URSC codes with suitable parameters to efficiently solve an even more general CR problem, in which **at least** s successful transmissions per station are required; in particular, in Appendix B we prove:

> *By simultaneously leveraging all three properties of our codes – shift resilience, flip resilience, and universality – we solve the generalized CR problem for any $k \leq n$ contenders (with k unknown), ensuring that each contender achieves at least s successful transmissions within $O((k + \frac{s}{\log n})^2 \log n)$ rounds after activation, for any $s \geq 1$.*

Our *constructive* upper bound matches the *existential* bound in [12] (case $s = 1$ in our generalized result) and gets very close to the lower bound of $\Omega(\frac{k^2}{\log k})$ proved in the same paper.

2 Formal definition and notation

Given a binary vector $\mathbf{x} = (x_1, x_2, \ldots, x_t)$, we denote by $S(\mathbf{x})$ the set of all *cyclic shifts* of $\mathbf{x}$, that is, $S(\mathbf{x})$ contains all different binary vectors of the form $(x_{1 \oplus i}, x_{2 \oplus i}, \ldots, x_{t \oplus i})$, where $\oplus$ denotes addition mod t and $i = 0, \ldots, t - 1$. It is clear that $1 \leq |S(\mathbf{x})| \leq t$.

For any binary vector $\mathbf{x}$, $|\mathbf{x}|$ represents the number of 1's in $\mathbf{x}$, also known as the *weight* of $\mathbf{x}$. The symbols $\vee$ and $\wedge$ denote bitwise OR and AND operators, respectively, applied to binary vectors.

Let $R = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_r\}$ be a set of binary vectors. Given R, we construct the set $S_\vee(R)$ as follows:

- First we consider all cyclic shifts for each vector in R.

- For each combination of cyclic shifts from the vectors in R, we perform a bitwise OR operation.

Formally, $S_\vee(R)$ is defined as:

$$S_\vee(R) = \left\{ \mathbf{z} = \bigvee_{i=1}^{r} \mathbf{z}_i \ \middle| \ \mathbf{z}_i \in S(\mathbf{y}_i), 1 \leq i \leq r \right\}.$$

In other words, $S_\vee(R)$ consists of all possible binary vectors obtained by taking the bitwise OR of one cyclic shift from each vector in R.
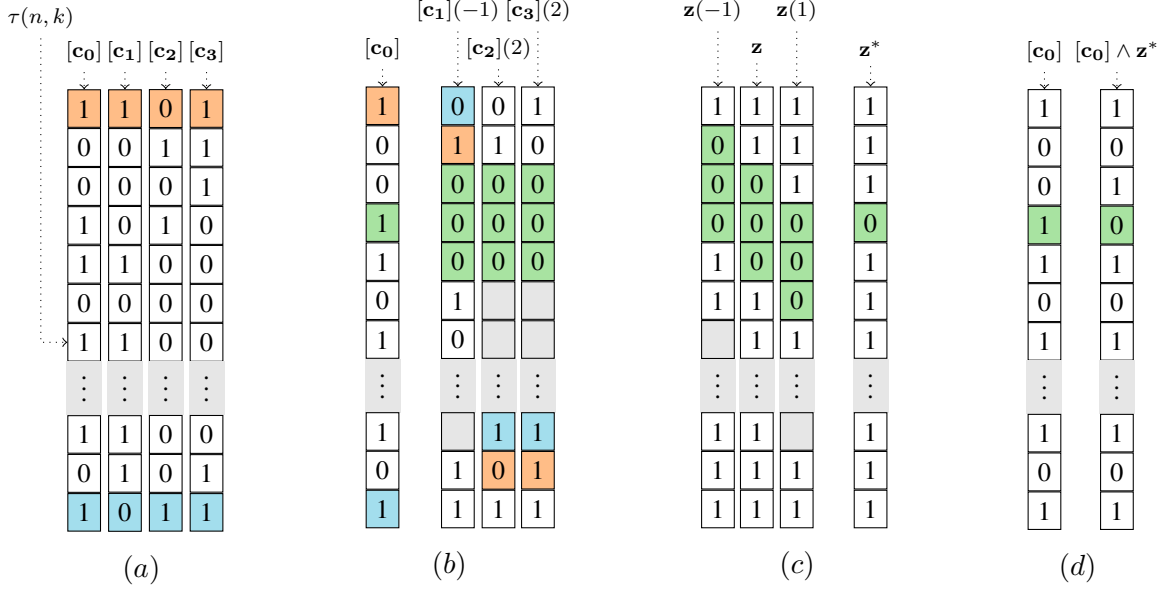
Figure 1: *An illustration of the ultra-resilient properties described in Definition 2 for parameters $k = 4 \leq n$ and $\alpha = 1$. In (a) an arbitrary subset $T = \{[c_0], [c_1], [c_2], [c_3]\}$ of column vectors in the matrix is depicted. (b) shows a designated column $[c_j] \in T$ (without loss of generality we assume that $[c_j] = [c_0]$), along with arbitrary shifts applied to the other columns in $T \setminus \{[c_0]\}$. The initial and final bits of each original column vector are highlighted orange and cyan, respectively. In (c) we have the superposition vector $z = [c_1](-1) \vee [c_2](2) \vee [c_3](2)$ surrounded by $z(-1)$ on its left and $z(1)$ on the right. The rightmost column corresponds to the slipped vector $z^* = z(-1) \vee z \vee z(1)$. In (d) the vectors $[c_j]$ and $[c_j] \wedge z^*$ are presented side by side. For $\alpha = 1$, the property ensures that $\left|([c_j] \wedge z^*)_{[0,\tau(n,k)]}\right| < \left|[c_j]_{[0,\tau(n,k)]}\right|$ indicating the existence of at least one row where column vector $[c_j]$ has a 1 while $[c_j] \wedge z^*$ has a 0, which in turn corresponds to $[c_j]$ having a 1 while $z(-1)$, z and $z(1)$ all having a 0 in the same position (see the rows highlighted green in (b) and (c) and (d)).*

Given a $t \times n$ binary matrix $\mathbf{M}$, we refer to the c_j-th column vector of $\mathbf{M}$ as $[c_j]$. For a column vector $[c_j]$, we denote $[c_j](i)$, where $i = 0, \ldots, t-1$, as the ith cyclic shift of $[c_j]$. Formally, if $[c_j] = (x_1, x_2, \ldots, x_t)$, then $[c_j](i) = (x_{1 \oplus i}, x_{2 \oplus i}, \ldots, x_{t \oplus i})$ for $i = 0, \ldots, t-1$. For simplicity, we extend this definition to any integer $i \geq 0$, with the understanding that i is taken modulo t throughout the paper. Finally, given any vector $\mathbf{x}$, its subvector from bit position β_1 to bit position β_2, where $0 \leq \beta_1 \leq \beta_2 \leq t$, is represented as $\mathbf{x}_{[\beta_1, \beta_2]}$.

We now present the formal definition of ultra-resilient superimposed codes. To do this, we first need to introduce two new concepts: code elongation, which generalizes the idea of code length for classic superimposed codes, and slipped vector, which is essential for preserving the isolation property.

Code elongation. In our general scenario, where codewords can undergo arbitrary adversarial shifts, the unknown parameter k introduces significant challenges that are not present in the classic case of fixed and aligned codewords. When codewords are fixed, a superimposed code for an unknown k can be constructed by concatenating superimposed codes for known, incrementally increasing parameters $k' \leq n$. However, it is well-known that when each codeword can experience an arbitrary adversarial shift, this concatenation technique, which attempts to 'guess' the unknown parameter by concatenation, becomes ineffective. As a result, the concept of *code length*, which for a known k (and a given n) was a fixed value corresponding to the number t of rows in the matrix, evolves into the more general notion of *code elongation* in our much broader definition of ultra-resilient superimposed codes for unknown k. Code elongation is characterized by a function $\tau : \mathbb{N} \times \mathbb{N} \to [0, t)$ that, in addition to the given n, also depends on the unknown parameter k. This dependence enables the code to maintain its ultra-resilience properties across different (unknown) column subset sizes k.

Slipped vector. For any vector $\mathbf{z}$, we define the corresponding *slipped* vector as $\mathbf{z}^* = \mathbf{z}(-1) \vee \mathbf{z} \vee \mathbf{z}(1)$. The slipped vector is crucial for ensuring the isolation property in the definition of ultra-resilient

superimposed codes given below (see Figure 1 for a graphical reference): any 1-bit in $[\mathbf{c_j}]$ that does not overlap with $\mathbf{z}^*$, i.e., in the surplus $[\mathbf{c_j}] \setminus ([\mathbf{c_j}] \wedge \mathbf{z}^*)$, indicates the existence of a row i such that $[\mathbf{c_j}]$ has a 1 at position i while $\mathbf{z}$ has 0's in positions $(i-1) \bmod t$, i, and $(i+1) \bmod t$.

Definition 2 (Ultra-resilient superimposed code). *Let n be any integer. Given a function $\tau : \mathbb{N} \times \mathbb{N} \to [0, t)$, where $t \geq n$, and a real number $0 < \alpha \leq 1$, we say that a $t \times n$ binary matrix $\mathbf{M}$ is a (n, α)-ultra-resilient superimposed code (denoted (n, α)-URSC) of elongation τ, if the following condition holds:*

For any $2 \leq k \leq n$ and any subset T of column indices of $\mathbf{M}$ with $|T| = k$, and for any column index $c_j \in T$, the inequality

$$\left| \left([\mathbf{c_j}] \wedge \mathbf{z}^* \right)_{[0, \tau(n,k)]} \right| < \alpha \cdot \left| [\mathbf{c_j}]_{[0, \tau(n,k)]} \right|$$

is satisfied for all $\mathbf{z} \in S_\vee(T \setminus \{c_j\})$.

3 Construction of Ultra-Resilient Superimposed Codes (URSC)

This section focuses on the construction of ultra-resilient superimposed codes without knowing the parameter k. The objective is to design a randomized algorithm that, given the input parameters n and α and for any $\epsilon > 0$, efficiently generates an ultra-resilient superimposed code with elongation $\tau(n, k) = c(k^2/\alpha^{2+\epsilon}) \ln n$, for any (unknown) $1 < k \leq n$. This is near-optimal in view of the $\Omega((k/\alpha)^2 (\log_{k/\alpha} n))$ lower bound proved in [31]. This lower bound also implies that to ensure the code elongation remains within practical limits, specifically polynomial in k, one can reasonably assume that $e^{-k} < \alpha \leq 1$.

Although the construction algorithm is randomized, the generated code can be used reliably in deterministic algorithms. Additionally, the code only needs to be generated once, as it can be reused in different contexts as long as the parameters n and α remain unchanged.

Our approach to efficiently constructing the codes revolves around two key concepts: the *Collision Bound Property* and the strategic selection of *assignment probabilities* for the 1's and 0's in the matrix. These concepts are closely intertwined: the Collision Bound Property streamlines the computational verification of code correctness, while the strategic assignment probabilities guarantee that the matrix satisfies the Collision Bound Property.

In Section 3.1, we introduce the Collision Bound Property and we demonstrate its sufficiency in ensuring that a given matrix qualifies as an ultra-resilient superimposed code. Section 3.2 introduces our random matrix construction method. Subsequently, Section 3.3 serves as the main technical segment where we establish that matrices generated using our random procedure have a high probability of satisfying both inequalities stipulated by the Collision Bound Property and consequently of qualifying as ultra-resilient superimposed codes. Finally, in Section 3.4, we outline the construction algorithm. This algorithm efficiently utilizes repeated applications of our random procedure of Section 3.2 to generate ultra-resilient superimposed codes.

Throughout this section, we assume that the two parameters n and α are fixed and given. Specifically, n is an integer such that $n \geq 2$, and α is a real number such that $e^{-k} < \alpha \leq 1$.

3.1 Collision Bound Property

The definition of URSC codes involves a condition on subsets of k columns, which can be computationally challenging to verify due to the super-polynomial number of such subsets. A crucial step towards an efficient construction is the introduction of the *Collision Bound Property*, a sufficient condition for ensuring the resilience properties of (n, α)-URSC, which applies to pairs of columns rather than subsets of k columns.

Before delving into the formal definition, let us summarize the Collision Bound Property. This property divides each column into upper and lower segments – specifically $[0, \tau_1(n, k)]$ and $[\tau_1(n, k), \tau_2(n, k)]$, with $\tau_2(n, k)$ corresponding to the elongation of the code – and ensures two key inequalities: the *Weight*

Inequality, which pertains to any individual column, and the *Collision Weight Inequality*, which applies to any pair of columns.

- **Weight Inequality**: The first inequality compares the weights of the upper and lower segments of any column. Specifically, it ensures that the weight of the upper segment is at most α times the weight of the lower segment. This establishes a specific dominance of the lower segment's weight over the upper segment's weight.

- **Collision Weight Inequality**: The second inequality bounds the "collision weight" of any pair of columns:

$$\left| \left([\mathbf{c_j}] \wedge \left([\mathbf{c_{j'}}](i-1) \vee [\mathbf{c_{j'}}](i) \vee [\mathbf{c_{j'}}](i+1) \right) \right)_{[\tau_1(n,k), \tau_2(n,k)]} \right|,$$

defined as the number of positions in the lower segment where a column intersects with the slipped vector of any cyclic shift of the other column of the pair. This inequality ensures that this collision weight does not exceed α times one $(k-1)$th of the lower segment's weight. This helps in controlling the overlap between columns, which is crucial for maintaining the superimposed code property.

Definition 3 (Collision Bound Property)**.** *Let* $\mathbf{M}$ *be a* $t \times n$ *binary matrix for some integer* $t \geq n$. *Let* $\tau_1, \tau_2 : \mathbb{N} \times \mathbb{N} \to [0, t)$ *be two integer functions.*

Collision Bound Property $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$: *For every* $1 < k \leq n$, *for each pair of column indices* c_j *and* $c_{j'}$ *and every cyclic shift* $[\mathbf{c_{j'}}](i)$, $0 \leq i \leq t-1$, *both of the following inequalities hold:*

- *Weight Inequality:*

$$\left| [\mathbf{c_j}]_{[0, \tau_1(n,k)]} \right| \leq \alpha \left| [\mathbf{c_j}]_{[\tau_1(n,k), \tau_2(n,k)]} \right|, \tag{1}$$

- *Collision Weight Inequality:*

$$\left| \left([\mathbf{c_j}] \wedge \left([\mathbf{c_{j'}}](i-1) \vee [\mathbf{c_{j'}}](i) \vee [\mathbf{c_{j'}}](i+1) \right) \right)_{[\tau_1(n,k), \tau_2(n,k)]} \right| \leq \left\lfloor \frac{\alpha \left| [\mathbf{c_j}]_{[\tau_1(n,k), \tau_2(n,k)]} \right| - 1}{k-1} \right\rfloor. \tag{2}$$

The following lemma establishes the sufficiency of the Collision Bound Property in guaranteeing that a matrix $\mathbf{M}$ is an $(n, 2\alpha)$-URSC.

Lemma 1. *Let* $\mathbf{M}$ *be a* $t \times n$ *binary matrix for some integers* $t \geq n$. *Assume that* $\tau_1, \tau_2 : \mathbb{N} \times \mathbb{N} \to [0, t)$ *are two integer functions such that the Collision Bound Property* $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$ *is satisfied. Then, the matrix* $\mathbf{M}$ $(n, 2\alpha)$-*URSC with elongation* $\tau_2(n, k)$.

Proof. Let us consider two functions τ_1, τ_2 and a $t \times n$ binary matrix $\mathbf{M}$ satisfying property $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$. For any $1 < k \leq n$, consider a fixed subset T of $|T| = k$ column indices, and a column index $c_j \in T$. Let $\mathbf{z} \in S_\vee(T \setminus \{c_j\})$ and $c_{m_1}, \ldots, c_{m_{k-1}}$ be the column indices in $T \setminus \{c_j\}$. By definition, $\mathbf{z} = [\mathbf{c_{m_1}}](i_1) \vee \cdots \vee [\mathbf{c_{m_{k-1}}}](i_{k-1})$, for some arbitrary shifts $0 \leq i_1, \ldots, i_{k-1} \leq t-1$. Hence, applying the distribution of conjunction over disjunction, we get

$$([\mathbf{c_j}] \wedge \mathbf{z})_{[\tau_1(n,k), \tau_2(n,k)]} = ([\mathbf{c_j}] \wedge [\mathbf{c_{m_1}}](i_1))_{[\tau_1(n,k), \tau_2(n,k)]} \vee \cdots \vee ([\mathbf{c_j}] \wedge [\mathbf{c_{m_{k-1}}}](i_{k-1}))_{[\tau_1(n,k), \tau_2(n,k)]}.$$

By inequality (2) of property $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$, it follows that

$$\left| ([\mathbf{c_j}] \wedge \mathbf{z})_{[\tau_1(n,k), \tau_2(n,k)]} \right| \leq (k-1) \left\lfloor \frac{\alpha \left| [\mathbf{c_j}]_{[\tau_1(n,k), \tau_2(n,k)]} \right| - 1}{k-1} \right\rfloor < \alpha \left| [\mathbf{c_j}]_{[\tau_1(n,k), \tau_2(n,k)]} \right|. \tag{3}$$

Considering also the second inequality of $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$, we have:

$$\begin{aligned}
\left| ([\mathbf{c_j}] \wedge \mathbf{z})_{[\tau_2(n,k)]} \right| &\leq \left| [\mathbf{c_j}]_{[0, \tau_1(n,k)]} \right| + \left| ([\mathbf{c_j}] \wedge \mathbf{z})_{[\tau_1(n,k), \tau_2(n,k)]} \right| \\
&< \alpha \left| [\mathbf{c_j}]_{[\tau_1(n,k), \tau_2(n,k)]} \right| + \alpha \left| [\mathbf{c_j}]_{[\tau_1(n,k), \tau_2(n,k)]} \right| \quad \text{by (1) and (3)} \\
&< 2\alpha \left| [\mathbf{c_j}]_{[\tau_2(n,k)]} \right|.
\end{aligned}$$

Since we have established this property for all $\mathbf{z} \in S_\vee(T \setminus c_j)$, and we have shown it holds for all possible subsets T and column indices c_j satisfying $|T| = k$, we conclude that $\mathbf{M}$ is a $(n, 2\alpha)$-URSC with elongation $\tau_2(n, k)$. □

3.2 Random construction

In this subsection, we present a random construction of a binary matrix. As demonstrated in the next subsection, this construction has a high probability of satisfying the Collision Bound Property $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$ and, hence, of being an $(n, 2\alpha)$-URSC in view of Lemma 1. In Theorem 1 we finally obtain an (n, α)-URSC.

In addition to n and α, the construction relies on two more parameters: an arbitrarily small constant $\epsilon > 0$ and a real constant $c > 0$. The role of ϵ is to bring the elongation of the code arbitrarily close to the asymptotic bound $O((k/\alpha)^2 \ln n)$. The constant c, if sufficiently large, ensures a high probability of successful construction, as will be demonstrated in the next section.

The goal of the random construction that we are going to describe is to assign the probability $p(r)$ of having a 1 in the rth bit of each column, for $0 \le r < t$. The ignorance of the parameter k presents new significant challenges (in addition to those related to the arbitrary shifts), as we cannot use k in assigning the probabilities $p(r)$, nor can we use a uniform distribution. Instead, we estimate k as we descend the positions of the columns, with probabilities in the upper part tailored for smaller values of k and those in the lower part for larger values of k. This is achieved by gradually decreasing the probabilities $p(r)$ as r increases, *i.e.*, as we move down the positions of the columns. Throughout this process, we must ensure that the two inequalities of the Collision Weight Property are satisfied, creating a subtle trade-off as explained below.

The Weight Inequality requires that the decrease in the frequency of 1's is controlled such that the weight of the lower segment (from $\tau_1(n, k)$ to $\tau_2(n, k)$) dominates the weight of the upper segment (up to position $\tau_1(n, k) - 1$). Conversely, the Collision Weight Inequality requires that this dominance does not lead to an excessive number of collisions in the lower segment; specifically, the number of collisions in the lower (dominating) segment must be significantly less than the weight of each column. Additionally, this inequality must be satisfied without extending $\tau_2(n, k)$ beyond the desired elongation.

As we will show, a carefully chosen probability function that decreases according to the square root of the inverse of the column's position r, strikes the right balance among all these conflicting targets.

Finally, it is worth noting, that although each column of the matrix will have length $t = c/(\alpha^{2+\epsilon})n^2 \ln n$ (since they cannot depend on k), it will be shown later that the code still guarantees an elongation of $\tau(n, k) = c/(\alpha^{2+\epsilon})k^2 \ln n$.

Definition 4 (Random Matrix Construction $\mathcal{M}(n, \alpha, \epsilon, c)$). *Let us define a random matrix $\mathcal{M}(n, \alpha, \epsilon, c)$ of n columns and $t = (c/\alpha^{2+\epsilon})n^2 \ln n$ rows, generated using the following procedure. The rth bit of each column, $0 \le r < t$, is independently set to 1 with a probability given by:*

$$p(r) = \sqrt{\frac{1}{\lfloor r/\ln n \rfloor + 1}},$$

and to 0 with the complementary probability. This corresponds to each column being partitioned into $(c/\alpha^{2+\epsilon})n^2$ blocks of equal length $\ln n$, with every bit of the bth block, for $0 \le b < (c/\alpha^{2+\epsilon})n^2$, independently set to 1 with a probability given by $1/\sqrt{b+1}$, and to 0 with the complementary probability.

3.3 Satisfying the Collision Bound Property

Our next objective is to show that for $\tau_1(n, k) = \frac{c}{64}k^2 \ln n$ and $\tau_2(n, k) = \frac{c}{\alpha^{2+\epsilon}}k^2 \ln n$, any random matrix constructed as illustrated in subsection 3.2, satisfies the Collision Bound Property $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$ with high probability. (It is important to clarify that, as we will see in Section 3.4, the functions τ_1 and τ_2, which are defined in terms of the unknown k (and n), do not need to be known by the construction algorithm.) Namely, we will be proving the following.

Lemma 2. *Fix any $\epsilon > 0$. Define $\tau_1(n, k) = (c/64)k^2 \ln n$ and $\tau_2(n, k) = (c/\alpha^{2+\epsilon})k^2 \ln n$, where $c > 0$ is a sufficiently large real constant. Let $\mathbf{M} = \mathcal{M}(n, \alpha, c)$ be a random matrix. For any given $1 < k \le n$, a pair of column indices c_j and $c_{j'}$, and a cyclic shift $[\mathbf{c_{j'}}](i)$ with $0 \le i \le t - 1$, the probability that the Collision Bound Property $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$ does not hold is less than $\frac{6}{c^2} \cdot n^{-8\ln(\frac{4}{\alpha})}$.*

The detailed technical proof of Lemma 2 is deferred to Appendix C. The proof of Lemma 2 is quite involved, as it requires bounding the probabilities associated with satisfying the Weight Inequality and the Collision Weight Inequality separately (subsections C.1 and C.2 of Appendix C, respectively).

The Weight Inequality requires us to control the expected weights of both the upper and lower segments of each column $[\mathbf{c_j}]$. We analyze these segments as expected values over the randomized matrix construction, allowing us to bound the probability that the Weight Inequality is satisfied (see Lemma 5).

For the Collision Weight Inequality, we define a random variable to represent the number of collisions between each pair of columns $[\mathbf{c_j}]$ and $[\mathbf{c_{j'}}]$ within the lower segment: $[\tau_1(n,k), \tau_2(n,k)]$. This random variable must account for any possible cyclic shift $0 \leq i < t$ (and corresponding slipped vector) of the second column $[\mathbf{c_{j'}}]$ relative to the first column $[\mathbf{c_j}]$. Specifically, for any fixed shift $0 \leq i < t$, this random variable is given by $\sum_{d=-1}^{1} |[\mathbf{c_j}] \wedge [\mathbf{c_{j'}}](i+d)|$. Bounding the probability that this variable meets the inequality's criteria is addressed in Lemma 11 and is particularly challenging for the following reasons.

The expected value of this variable is heavily influenced by the shift magnitude of the second column of the pair, which significantly affects, within the lower interval $[\tau_1(n,k), \tau_2(n,k)]$, the probabilities $p((r+i+d) \bmod t)$, representing the probability that the slipped vector of the shifted column hosts a 1 in its rth position after a shift i, for $0 \leq i < t$ and $d = -1, 0, 1$. To address this, Lemma 11 analyzes three separate cases based on the magnitude of probabilities $p((r+i+d) \bmod t)$ for all r in the interval $\tau_1(n,k) \leq r \leq \tau_2(n,k)$.

The first case handles collisions when the shift of the second column in the pair is minimal, causing all probabilities of the shifted column to be high within the positions of the lower segment. Here, we apply the *rearrangement inequality* [18] (see the appendix), which enables a precise estimation of the expected number of collisions for high-probability entries. This technique maximizes the sum of products of corresponding probabilities (i.e., the expected collision count) by aligning the largest probability values in both columns, yielding a tight upper bound on collisions under minimal shifts.

In the second case, we consider mid-range shifts where the probabilities in the shifted column remain within a constant factor of each other. To handle this, we develop a *pairwise bounding technique* that leverages the near-uniformity of probabilities across positions. This approach minimizes the dependency on exact probabilities, allowing us to obtain a balanced estimate of collisions despite moderate variations.

Finally, the third case considers the situations where the probabilities of the shifted column are very low from $\tau_1(n,k)$ up to some intermediate position τ', with $\tau_1(n,k) < \tau' < \tau_2(n,k)$, and then become large from the next position $\tau'+1$ until $\tau_2(n,k)$. To address this, we employ a *two-segment analysis* that treats the low-probability and high-probability segments separately. Specifically, we combine the techniques from Case 2 for the low-probability segment and Case 1 for the high-probability segment.

The proof of Lemma 2 is ultimately completed by combining the probability of satisfying the Weight Inequality (Lemma 5) and that of satisfying the Collision Weight Inequality (Lemma 11).

3.4 The construction algorithm

The construction of the ultra-resilient superimposed codes is accomplished by Algorithm 1, a randomized algorithm that, in addition to the parameter n and the real number $0 < \alpha \leq 1$, takes as input an arbitrarily small constant $\epsilon > 0$ and a constant $c > 0$. The next lemma proves that if the constant c is sufficiently large as established in Lemma 2, then Algorithm 1 outputs an $(n, 2\alpha)$-URSC with elongation $\tau(n,k) = c(k^2/\alpha^{2+\epsilon}) \ln n$, for any $1 < k \leq n$.

Lemma 3. *For $\alpha \in (0,1]$, Algorithm 1 generates with high probability in polynomial time an $(n, 2\alpha)$-URSC with elongation $\tau(n,k) \leq \frac{c \cdot k^2}{\alpha^{2+\epsilon}} \ln n$, for any $1 < k \leq n$. The same result can be obtained in expectation.*

Proof. The algorithm sets functions $\tau_1(x,y) = (c/64)x^2 \ln y$ and $\tau_2(x,y) = c(x^2/\alpha^{2+\epsilon}) \ln y$. Then it uses a `repeat-until` loop, which at each iteration generates a random matrix $\mathbf{M} = \mathcal{M}(n, \alpha, \epsilon, c)$,

---

Algorithm 1: Compute ultra-resilient superimposed codes (k is unknown)

---

1 Input: an integer $n \geq 2$, a real number $0 < \alpha \leq 1$, an arbitrarily small constant $\epsilon > 0$ and a constant
$c > 0$.

2 Output: a matrix $\mathbf{M}$ that is a $(n, 2\alpha)$-URSC with elongation $\tau(n, k) = c(k^2/\alpha^{2+\epsilon}) \ln n$, for any
$1 < k \leq n$.

3 Let τ_1 and τ_2 be functions defined as: $\tau_1(x, y) = (c/64)x^2 \ln y$, $\tau_2(x, y) = c(x^2/\alpha^{2+\epsilon}) \ln y$.

4 repeat

5 $\quad$ Generate a random matrix $\mathbf{M} = \mathcal{M}(n, \alpha, \epsilon, c)$ (see Definition 4);

6 until `Check_`$\mathcal{P}$ ($\mathbf{M}$, α, τ_1, τ_2) = TRUE;

7 return *the generated matrix* $\mathbf{M}$;

8 Function `Check_`$\mathcal{P}$ ($\mathbf{M}$, α, τ_1, τ_2) :

9 $\quad$ **for** $1 < k \leq n$ **do**

10 $\quad\quad$ **for** *each pair of column indices c_j and $c_{j'}$ of* $\mathbf{M}$ **do**

11 $\quad\quad\quad$ **for** *every shift* $[\mathbf{c_{j'}}](i)$ **do**

12 $\quad\quad\quad\quad$ **if** *either the Weight Inequality (1) or the Collision Weight Inequality, or both, are not*
satisfied **then**

13 $\quad\quad\quad\quad\quad$ **return** FALSE; // if the Collision Bound Property is not
$\quad\quad\quad\quad\quad\quad\quad$ satisfied, the function terminates and returns FALSE

14 $\quad$ **return** TRUE;

---

according to the random construction of Definition 4, and invokes a function `Check_`$\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$.
It continues iterating until this function returns TRUE.

The role of `Check_`$\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$ is simply to verify the satisfaction of the Collision Bound Property $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$. This verification is ensured by the three nested `for` loops in lines 9 - 13. Namely, the function returns TRUE if and only if, for every $1 < k \leq n$, for every pair of column indices c_j and $c_{j'}$, and every cyclic shift $[\mathbf{c_{j'}}](i)$ with $0 \leq i \leq t - 1$, both the Weight Inequality (1) and the Collision Weight Inequality (2) hold. This indeed corresponds to the Collision Bound Property $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$ being satisfied. Consequently, from Lemma 1 it follows that when the execution exits the `repeat-until` loop, the last generated matrix $\mathbf{M}$ is a valid (n, α)-URSC with elongation $\tau(n, k) \leq (c/\alpha^{2+\epsilon})k^2 \ln n$, for any $1 \leq k \leq n$.

To complete the proof, we must demonstrate that the total number of operations involved is polynomial with high probability and in expectation. Since the number of operations required to construct the random matrix is evidently polynomial — specifically, $n \cdot t = n \cdot \left(\frac{c}{\alpha^{2+\epsilon}} n^2 \ln n\right)$ random choices, one for each bit of the matrix — it suffices to show the following. First, we prove that each iteration requires polynomial time. Then, we establish that the number of iterations is constant both with high probability and in expectation.

Number of operations per iteration. In each iteration of the `repeat-until` loop, the function `Check_`$\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$ is invoked to verify whether the matrix satisfies property $\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$. In this function, the three nested loops cause the `if` condition on line 12 to be checked at most $n \cdot 2\binom{n}{2} \cdot t$ times. Checking the `if` condition requires verifying inequalities (1) and (2), which takes fewer than $3t$ operations.

Specifically, the first inequality involves counting the number of 1's in a vector $[\mathbf{c_j}]$ over no more than t positions and then comparing two values. The second inequality requires counting the number of 1's in a vector $\left([\mathbf{c_j}] \wedge ([\mathbf{c_{j'}}](i-1) \vee [\mathbf{c_{j'}}](i) \vee [\mathbf{c_{j'}}](i+1))\right)$ and in a vector $[\mathbf{c_j}]$, both over at most t positions, followed by comparing two values.

Overall, any iteration of the `repeat-until` loop takes time no more than $n \cdot 2\binom{n}{2} \cdot t \cdot 3t < 3n^3 t^2$.

Number of iterations. The remaining task is to count the total number of iterations of the algorithm until termination. Let's consider the probability that the function `Check_`$\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$, invoked on a

random matrix $\mathbf{M} = \mathcal{M}(n, c)$ with τ_1 and τ_2 set as specified in line 3, returns FALSE during a specific iteration of the innermost loop. This corresponds to the probability that, for any fixed value of k in the first loop, any pair of columns fixed in the second loop, and any shift $[\mathbf{c_{j'}}](i)$ fixed in the third loop, the inequalities (1) and (2) do not hold. Assuming that the input constant c used to build the matrix $\mathbf{M} = \mathcal{M}(n, c)$ is sufficiently large, according to Lemma 2, the probability that in any of the iterations of the three nested loops the inequalities do not hold, is at most $\frac{6}{c^2} \cdot n^{-8 \ln\left(\frac{4}{\alpha}\right)}$. Therefore, by applying the union bound over all three nested loops, we can calculate the probability that the function returns FALSE in any of its iterations as follows:

$$3n^3 t^2 \left(\frac{6}{c^2} \cdot n^{-8 \ln\left(\frac{4}{\alpha}\right)} \right) = \frac{3c^2 n^7 \ln^2 n}{\alpha^{4+2\epsilon}} \left(\frac{6}{c^2} \cdot n^{-8 \ln\left(\frac{4}{\alpha}\right)} \right) = \frac{18 \ln^2 n}{\alpha^6} \cdot n^{7 - 8 \ln(4/\alpha)} < \frac{1}{n^2} \, ,$$

where the last inequality holds for $n \geq 9$. Therefore, for sufficiently large n, a single iteration of the `repeat-until` loop is sufficient to obtain a valid (n, α)-URSC with high probability. Given that we have demonstrated that each iteration requires polynomial time, it follows that the algorithm concludes within polynomial time with high probability.

Analogously, to demonstrate that the result also holds in expectation, it suffices to show that the expected number of iterations of the `repeat-until` loop is constant. The probability that the algorithm terminates at the i-th iteration is the probability that, in the first $i - 1$ iterations, the function Check_$\mathcal{P}(\mathbf{M}, \alpha, \tau_1, \tau_2)$ returns FALSE, and only at the i-th iteration does it return TRUE. Thus, the number of iterations follows a geometric distribution with a success probability of $p = (1 - \frac{1}{n^2})$. The expected value for such a geometric distribution is known to be $1/p = \frac{n^2}{n^2-1}$, which is $O(1)$. $\qquad \square$

Now we can observe that if we want to construct codes guaranteeing flip resilience for $\alpha \in (0, 1]$, it is enough to run Algorithm 1 for $\alpha' = \alpha/2$. Since $\alpha' \in (0, 1/2]$, Lemma 3 implies the following.

Theorem 1. *For $\alpha \in (0, 1]$, Algorithm 1 can be used to generate with high probability in polynomial time an (n, α)-URSC having elongation $\tau(n, k) = O\left(\frac{k^2}{\alpha^{2+\epsilon}} \ln n \right)$, for any $1 < k \leq n$. The same result can also be obtained in expectation.*

3.5 Ultra-resilient superimposed codes parameterized by length

We could parameterize the ultra-resilient superimposed codes also by their length t, e.g., we could consider codes with length shorter than in Theorem 1. More precisely, in Definition 2, we could consider any length t of the code and vary the upper bound on the values of k for which the elongation guarantee holds, which originally was n, to some parameter Δ, where Δ is the maximum integer such that: $\Delta \leq n$ and $\tau(n, \Delta) \leq t$. This lowering of the upper bound on parameter k is natural, because shorter codes could not guarantee a successful position of any element in every configuration of the codewords, due to existing lower bounds on the length of the code even if k is known and there are no shifts, see e.g., [14].

In general, if we cut the ultra-resilient superimposed code at some smaller length $t' < t$, the resulting code may not satisfy the elongation property in Definition 2 for many values of parameter k, because the shifts in the original and the cut codes results in different configurations. However, our construction in Section 3, still works for shorter codes, with only few minor updates in the construction algorithm and in the analysis. Mainly, we need to replace the formula for the length $t = (c/\alpha^{2+\epsilon})n^2 \ln n$ by $t = (c/\alpha^{2+\epsilon})\Delta^2 \ln n$, and consider only parameters $2 \leq k \leq \Delta$. Note that throughout the analysis, all the probability bounds depend on the number of codewords n, the $\log n$ factor coming from the elongation function, and α; all of these stay the same in the t-parameterized codes, hence the probabilities stay analogous (only constants may change, which we could accomodate here by taking a larger constant c).

This way we can prove the following extension of Theorem 1 to codes with arbitrary length t.

Theorem 2. *Algorithm 1 can be modified to generate in polynomial time, with high probability, an (n, α)-URSC of a given length t, with elongation $\tau(n, k) = O\left(\frac{k^2}{\alpha^{2+\epsilon}} \ln n \right)$, for any $1 \leq k \leq \Delta$, where Δ is the maximum integer satisfying: $\Delta \leq n$ and $\tau(n, \Delta) \leq t$. The same result can also be obtained in expectation.*

4 Code Applications in Uncoordinated Beeping Networks

4.1 Model and problem

Let G be an underlying beeping network with n nodes, modeled as an undirected simple graph. Each node v in the graph has a unique identifier from the set $\{1, \ldots, n\}$. Without loss of generality, we will refer to both the node and its identifier as v. Each node is initially aware only of its own identifier, the total number of nodes n, and an upper bound Δ on its degree (the number of neighbors).

In the *uncoordinated setting*, nodes are activated in arbitrary rounds, as determined by a conceptual adversary.[1] The goal for each node v is to maintain a set of identifiers N_v^* that satisfies the following properties:

Inclusion: N_v^* contains all the identifiers of the neighbors of v in G.

Safety: N_v^* does not contain the identifier of any node that is not a neighbor of v in G.

This problem is referred to as *neighborhood learning*.

In a more general problem, each node has to maintain a set of input messages stored in its neighbors. This problem is often called *local broadcast*.

Time complexity of a given problem in uncoordinated beeping networks is typically measured as a worst-case (over graph topologies and adversarial wake-up schedules) number of rounds from the moment when all nodes become awaken until the task is achieved, see [13]. In the case of neighborhood learning and local broadcast, the task is achieved if all locally stored sets of neighbors (resp., messages in neighbors) contain all neighbors (resp., all neighbors' messages). Note that, due to the Safety condition, when one of these two tasks is achieved, the locally stored sets of neighbors (resp., messages) do not change in the future rounds. On the other hand, to guarantee the Inclusion property under arbitrarily long delays in adversarial wake-up schedule, algorithmic solutions have to be prepared for an arbitrarily long run – therefore, periodic algorithms seem to be most practical and as such have been considered in the literature, see [13].

It needs to be noted that the time complexity bound, denote it by $\mathcal{T}$, of our algorithms, proved in the analysis, satisfies even a stronger property: for any edge in the underlying network G, its both end nodes put each other (resp., each other's message) to the locally maintained set within time $\mathcal{T}$ after *both of them* become awaken. It means that we do not have to wait with time measurement until all nodes in the network are awaken, in case we are interested in specific point-to-point information exchange.

4.2 Neighborhood learning in uncoordinated beeping networks

Let G be an underlying beeping network of n nodes and node degree less than Δ. Each node knows only its id and the integers n and Δ.[2] All nodes have the same $(n, \frac{3}{4})$-URSC $\mathbf{M}$ from Theorem 2, of length set to $t = (c/\alpha^{2+\epsilon})\Delta^2 \ln n$ and elongation $\tau(n, k) \leq (c/\alpha^{2+\epsilon})k^2 \ln n$, for some constant $c > 0$, parameter $\alpha = \frac{1}{2} \cdot \frac{3}{4} = \frac{3}{8}$ and for any $k \leq \Delta$ (the latter is because $\tau(n, \Delta) = t$ and $\Delta \leq n$); in practice, it is enough that each node v knows only the corresponding column v of the code.[3] Additionally, each node computes in the beginning its unique block ID of length $7 + 2 \log n$, defined next.

Block ID. For a given node $v \in \{1, \ldots, n\}$, we define the *block ID of v*, denoted $\mathbf{b}_v$, as follows. It is a binary sequence of length $7 + 2 \log n$, it starts with three 1s followed by three 0's and another 1. To define the remaining $2 \log n$ positions of the block ID, we take the binary representation of number v, of logarithmic length, and simultaneously replace each 0 by bits 01 and each 1 by bits 10.

---

[1] Alternatively, the uncoordinated setting can be viewed as a temporal graph, where each node becomes "visible" after its adversarial wake-up time, and each edge becomes "visible" during the earliest time interval when both of its endpoints are awake.

[2] The algorithm and its analysis work also if each node knows some polynomial upper bound on n and a linear upper bound on Δ.

[3] Constant $\frac{3}{4}$ is set arbitrarily – it could be any constant in the interval $(\frac{1}{2}, 1)$ to allow the application of Theorem 3.

Main idea and intuitions. Here, we provide an overview of the key ideas and intuitions behind our approach, and we refer the reader to Appendix D for the full proofs. After awakening, a node periodically repeats a certain procedure – see the pseudo-code Algorithm 2. This periodicity is to assure that each node can properly pass its id, via sequence of beeps and idle rounds, to a neighbor who may be awaken at arbitrary time after the considered node. The main idea of the repeated part of Algorithm 2, see lines 6 - 16, which is in fact a form of another code, is as follows. A node v uses its corresponding codeword $\mathbf{c}_v$ from the ultra-resilient superimposed code $\mathbf{M}$ and substitutes each 1 in the codeword $\mathbf{c}_v$ by its block ID $\mathbf{b}_v$ of length $7 + 2\log n$, and each 0 in $\mathbf{c}_v$ by the block of zeros of the same length. Then, each node beeps at rounds corresponding (according to its local clock) to positions with 1's in the obtained sequence, and stays idle otherwise; see line 10 and preceding iterations in lines 7 and 8 corresponding to iterating over the length of the codewords $\mathbf{c}_v, \mathbf{b}_v$. The feedback, a beep or no beep, from the neighbors and the node itself is recorded in lines 12 and 14. Finally, a check is done (line 15) whether the recorded feedback in the last $7 + (2\log n)$ positions corresponds to any valid block ID, and if so, adding it to set N_v^* (unless it is already there), see line 16.

Intuitively, the codewords from $\mathbf{M}$ are to assure that some beeped block ID of any neighbor node will not be overlapped by any block ID beeped by its competing neighbors at the same time – therefore, it could guarantee the Inclusion property (see Lemma 15). More precisely, the shift property of the code guarantees that each neighbor has a unique 1 in its $\mathbf{c}_v$, corresponding to some block ID of v, (i.e., while other neighbors have only 0's in its overlapping blocks), regardless of the adversarial shift of the code. However, such single 1 does not guarantee passing id to the neighbor – this is why we need to substitute each 1 in the code by the block ID that allows decoding of the actual id if there are no beeping from other nodes at that time. This is challenging, because not only the original codewords from $\mathbf{M}$ could be adversarially shifted, but also the blocks themselves. Hence, to assure no beeping of other neighbors during the time when a block ID is beeped, we use the isolation property – at some point, all neighbors not only have one overlapping block of 0's, but also the preceding and the next blocks are the block of 0's. (This is because the isolation property guarantees that the preceding and the next position in the original codewords $\mathbf{c}_w$ of other at most $\Delta - 2$ neighbors have to be all 0's.)

The reason why we cannot use just a node id as its block ID is the following. The last $2\log n$ bits are to assure that any two aligned block IDs differ on at least two positions (which helps to assure that if a block ID is heard, it is not a "beeping superposition" of other block IDs but indeed a single node beeping its own block ID). The first 7 bits are to guarantee that no genuine block ID could be heard while there is no beeping block ID aligned. This assures Safety property, see Lemma 14 for details, and helps to fulfill Inclusion property too.

Theorem 3. *Algorithm 2 can be instantiated with some $(n, \frac{3}{4})$-URSC $\mathbf{M}$ of length $t = O(\Delta^2 \log n)$ and it guarantees learning neighborhoods deterministically by each node in $O(\Delta^2 \log^2 n)$ rounds after awakening of the node and the neighbors.*

4.3 From learning neighbors to local broadcast

Now suppose that each node v has a message of length $\mathcal{M}$. It splits it into a sequence of $\log n$ messages of size $\mathcal{M}/\log n$ each, say $M_v[1], \ldots, M_v[\mathcal{M}/\log n]$. Then, it creates extended messages $M_v^*[i]$, for any $1 \leq i \leq \mathcal{M}/\log n$, as follows: it puts a binary representation of v by $\log n$ bits first, then it puts a single bit equal to 1 for $i = 1$ and to 0 otherwise (this bit indicates whether it is the first extended message or not), and then appends $M_v[i]$. The length of $M_v^*[i]$ is $2\log n + 1 \leq O(\log n)$.

There are two changes in Algorithm 2. First, we now treat $M_v^*[i]$ as a set of new ids on node v. Hence, we need a $(2n^2, 3/4)$-URSC $\mathcal{M}$ from Theorem 2, but of asymptotically same length $t = O(\Delta^2 \log n)$ as for neighborhood learning (as obviously $\log(2n^2) = \Theta(\log n)$ and we want to use the same bound Δ). Second, node v in its ith periodic procedure (recall that such a procedure is in lines 6 - 16) uses $M_v^*[i \mod \mathcal{M}/\log n]$ as its id. Let's denote this modified algorithm as the *ultra-resilient Beeping Algorithm*.

The analysis is analogous, in particular, all nodes receive all messages $M_w^*[i]$ from their neighbors

---

Algorithm 2: Neighborhood learning algorithm in an uncoordinated beeping network; pseudo-code for node v after its (uncoordinated) wake-up

---

1 **Input:** Integers $n \geq \Delta \geq 1$, identifier $v \in \{1, \ldots, n\}$, a codeword $\mathbf{c}_v$ from $(n, \frac{3}{4})$-URSC $\mathbf{M}$ from
　Theorem 2, of length set to $t = (c/\alpha^{2+\epsilon})\Delta^2 \ln n$

2 **Maintained:** Set N_v^* of identifiers

3 $\mathbf{b}_v \leftarrow$ block ID of v ;

4 $N_v^* \leftarrow \emptyset$;

5 **while** `True` **do**

6 　　$\sigma_v \leftarrow$ sequence of $t \cdot (7 + 2 \log n)$ zeros ;

7 　　**for** $i = 1$ *to* t **do**

8 　　　　**for** $j = 1$ *to* $7 + 2 \log n$ **do**

9 　　　　　　**if** $\mathbf{c}_v[i] \wedge \mathbf{b}_v[j]$ **then**

10 　　　　　　　v beeps // `beeping block ID of` v `when` $\mathbf{c}_v[i] = 1$

11 　　　　　　**if** v *has beeped or heard a beep* **then**

12 　　　　　　　$\sigma_v[(i-1) \cdot (7 + 2 \log n) + j] \leftarrow 1$

13 　　　　　　**else**

14 　　　　　　　$\sigma_v[(i-1) \cdot (7 + 2 \log n) + j] \leftarrow 0$

15 　　　　　　**if** *sub-sequence* $\sigma_v[(i-2) \cdot (7 + 2 \log n) + j + 1 \mod t \cdot (7 + 2 \log n),$
　　　　　　$\ldots, (i-1) \cdot (7 + 2 \log n) + j \mod t \cdot (7 + 2 \log n)]$ *is equal to block ID of some*
　　　　　　$w \in \{1, \ldots, n\}$ **then**

16 　　　　　　　add w to set N_v^* (unless it is already there)

---

w, they are able to decode that they are from w by looking at the first $\log n$ of the decoded $M_w^*[i]$, to identify the starting message by looking at bit $\log n + 1$ (and then identifying the last message from w by looking for the last bit 0 at that position before getting 1 at that position), and getting the actual content by looking at the last $\log n$ bits of $M_v^*[i]$ (and concatenating contents starting from the first identified message from w up to the last piece). The only differences are that now both the code $\mathcal{M}$ and block IDs are a constant factor longer, and node v has to wait $\mathcal{M}/\log n$ periodic procedures to be able to decode all parts of the original message. Hence, the analog of Theorem 3 can be proved for local broadcast:

Theorem 4. *The Ultra-resilient Beeping Algorithm guarantees deterministic local broadcasting of an input message of length $\mathcal{M}$ by each node to each of its neighbors in $O(\Delta^2 \log n \cdot (\mathcal{M} + \log n))$ rounds after awakening of the node and the neighbor.*

5   Open Directions

There are several promising directions for future research. First, expanding the applications of URSCs to additional domains, such as genomic alignment and dynamic database search, could offer substantial advantages due to their inherent fault-tolerant and asynchronous properties. Exploring applications in other distributed and parallel computing contexts, as well as investigating further properties, like code weight and fairness in mechanism design, are intriguing directions for advancing URSC capabilities.

Closing the small gap between the code length of URSCs and the theoretical lower bound is a challenging yet valuable endeavor. This, along with deeper exploration of ultra-resilient properties in new settings, holds potential for further enhancing the impact of URSCs.

In summary, our contributions position URSCs as a robust, scalable solution to foundational challenges in distributed computing. We anticipate that URSCs will stimulate future research into resilient coding for asynchronous and dynamic environments, pushing the boundaries of fault-tolerant coding theory. More conclusions and future directions are deferred to Appendix A.

Acknowledgments

References

[1] Norman Abramson. The aloha system: Another alternative for computer communications. In *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, AFIPS '70 (Fall), pages 281–285. ACM, 1970.

[2] Noga Alon and Vera Asodi. Learning a hidden subgraph. *SIAM Journal on Discrete Mathematics*, 18(4):697–712, 2005.

[3] Giorgos Chionas, Bogdan S. Chlebus, Dariusz R. Kowalski, and Piotr Krysta. Adversarial contention resolution games. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 2598–2606. ijcai.org, 2023.

[4] Vicent Cholvi, Pawel Garncarek, Tomasz Jurdzinski, and Dariusz R. Kowalski. Stable routing scheduling algorithms in multi-hop wireless networks. *Theor. Comput. Sci.*, 921:20–35, 2022.

[5] Marek Chrobak, Leszek Gasieniec, and Wojciech Rytter. Fast broadcasting and gossiping in radio networks. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 575–581. IEEE Computer Society, 2000.

[6] W. Chu, C.J. Colbourn, and V.R. Syrotiuk. Slot synchronized topology-transparent scheduling for sensor networks. *Computer Communications*, 29(4):421–428, 2006. Current areas of interest in wireless sensor networks designs.

[7] Wensong Chu, Charles J. Colbourn, and Violet R. Syrotiuk. The effects of synchronization on topology-transparent scheduling. *Wireless Networks*, 12(6):681–690, 2006.

[8] A. E. F. Clementi, A. Monti, and R. Silvestri. Distributed broadcast in radio networks of unknown topology. *Theoretical Computer Science*, 302:337–364, 2003.

[9] Alejandro Cornejo and Fabian Kuhn. Deploying wireless networks with beeps. In *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, volume 6343 of *Lecture Notes in Computer Science*, pages 148–162. Springer, 2010.

[10] G. De Marco and D. Kowalski. Fast nonadaptive deterministic algorithm for conflict resolution in a dynamic multiple-access channel. *SIAM J. Comput*, 44(3):868–888, 2015.

[11] Gianluca De Marco, Dariusz R. Kowalski, and Grzegorz Stachowiak. Deterministic contention resolution without collision detection: Throughput vs energy. In *41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021*, pages 1009–1019. IEEE, 2021.

[12] Gianluca De Marco, Dariusz R. Kowalski, and Grzegorz Stachowiak. Deterministic non-adaptive contention resolution on a shared channel. *Journal of Computer and System Sciences*, 133:1–22, 2023.

[13] Fabien Dufoulon, Janna Burman, and Joffroy Beauquier. Can uncoordinated beeps tell stories? In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, page 408–417. Association for Computing Machinery, 2020.

[14] A.G. D'yachkov and V.V. Rykov. Bounds on the length of disjunct codes. *Problems of Information Transmission*, 18(3):7–13, 1982.

[15] P Erdös, P Frankl, and Z Füredi. Families of finite sets in which no set is covered by the union of two others. *Journal of Combinatorial Theory, Series A*, 33(2):158–166, 1982.

[16] Z. Füredi. On r-cover-free families. *Journal of Combinatorial Theory, Series A*, 73(1):172–173, 1996.

[17] I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products*. Elsevier/Academic Press, Amsterdam, seventh edition, 2007. Translated from the Russian, Translation edited and with a preface by Alan Jeffrey and Daniel Zwillinger, With one CD-ROM (Windows, Macintosh and UNIX).

[18] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1934.

[19] Piotr Indyk. Deterministic superimposed coding with applications to pattern matching. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997*, pages 127–136. IEEE Computer Society, 1997.

[20] Piotr Indyk. Explicit constructions of selectors and related combinatorial structures, with applications. In David Eppstein, editor, *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*, pages 697–704. ACM/SIAM, 2002.

[21] W. H. Kautz and R. C. Singleton. Nonrandom binary superimposed codes. *IEEE Transactions on Information Theory*, 10:363–377, 1964.

[22] Thomas Kesselheim, Robert D. Kleinberg, and Rad Niazadeh. Secretary problems with non-uniform arrival order. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 879–888. ACM, 2015.

[23] Donald E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., USA, 1998.

[24] J. Komlós and A. G. Greenberg. An asymptotically optimal nonadaptive algorithm for conflict resolution in multiple-access channels. *IEEE Trans. on Information Theory*, 31:302–306, 1985.

[25] D. Kowalski. On selection problem in radio networks. In *Proceedings, 24th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 158–166, Las Vegas, NV, USA, 2005. ACM.

[26] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[27] Robert M. Metcalfe and David R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Commun. ACM*, 19(7):395–404, jul 1976.

[28] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[29] R.A. Moser and G. Tardos. A constructive proof of the general lovász local lemma. *Journal of the ACM (JACM)*, 57(2):11, 2010.

[30] Ely Porat and Amir Rothschild. Explicit nonadaptive combinatorial group testing schemes. *IEEE Trans. Inf. Theory*, 57(12):7982–7989, 2011.

[31] Adele A. Rescigno and Ugo Vaccaro. Improved algorithms and bounds for list union-free families. *IEEE Trans. Inf. Theory*, 70(4):2456–2463, 2024.

[32] Lawrence G. Roberts. Aloha packet system with and without slots and capture. *SIGCOMM Comput. Commun. Rev.*, 5(2):28–42, April 1975.

[33] M. Ruszinkó. On the upper bound of the size of the r-cover-free families. *Journal of Combinatorial Theory, Series A*, 66(2):302–310, 1994.

[34] J. Wolf. Born again group testing: multiaccess communications. *IEEE Transactions on Information Theory*, 31(2):185–191, 1985.

Appendix

A Further Conclusions and Future Directions

In this work, we introduced Ultra-Resilient Superimposed Codes (URSCs), which extend classic superimposed codes with a stronger codewords' isolation property and unprecedented resilience to cyclic shifts and bitwise corruption, without requiring prior knowledge of the number of concurrent codewords, k. Our polynomial-time construction is the first efficient implementation of superimposed codes that can handle arbitrary shifts without assuming k, achieving near-optimal length that matches the best-known codes lacking these ultra-resilient properties.

URSCs provide significant improvements for several distributed problems where synchronization, essential for classic codes, is prohibitively expensive. For instance, in uncoordinated beeping networks, URSCs achieve nearly two orders of magnitude improvement in local broadcast efficiency. They also support deterministic contention resolution in multi-access channels, underscoring their potential to improve robustness and adaptability in real-world distributed systems.

Improvements in other distributed problems. Beyond the immediate applications to beeping networks and contention resolution, URSCs offer broader potential across diverse fields where synchronization constraints are challenging. For example, URSCs may improve scenarios where packets arrive dynamically at stations on a channel or in multi-hop wireless networks. In these cases, current efficient scheduling methods [4] rely on extended superimposed codes; our codes could replace them, potentially reducing their length to a near-optimal $O(k^2 \log n)$.

Additionally, our codes can benefit dynamic scheduling problems, such as topology-transparent and asynchronous schedules, where they efficiently manage activation times of participants in distributed systems. Prior studies, such as those by Chu, Colbourn, and Syrotiuk [6, 7], have highlighted the need for adaptable codes in such asynchronous contexts, and URSCs provide an efficient, near-optimal solution in these environments.

The broader applicability of URSCs extends to distributed database search, pattern recognition, and genomic data analysis. In distributed databases, traditional superimposed codes require aligned file descriptors, which can be impractical in asynchronous or adversarial retrieval systems. Here, URSCs overcome these limitations by tolerating out-of-order data arrivals and data corruption, thanks to their configurable parameter α, making them ideal for robust data handling. This approach is particularly relevant in the context of data-dependent superimposed codes by Indyk [19], which have applications in pattern matching but lack tolerance to misalignment and faults.

URSCs also hold promise for genomic sequence alignment in large, distributed datasets, where their resilience to misalignments and adversarial conditions can enhance fault tolerance in biological data processing.

$$\begin{array}{cccccccc}
v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & \cdots & v_n
\end{array}$$

Figure 2: *The one-to-one correspondence between stations and columns of* $\mathbf{M}$*: the transmission vector of station v_i will be the column at index v_i.*

B   Contention Resolution on a Multiple-Access Channel

In this section, we introduce a generalized version of the Contention Resolution (CR) problem on a multiple-access channel and demonstrate how to apply our codes to get an efficient solution.

B.1   Model and problem

We consider a group of n stations connected to a shared transmission medium called a *shared channel*. Each station v is uniquely identified by an ID in the range $\{1, \ldots, n\}$. For the sake of presentation and to simplify the notation, we treat v and its ID interchangeably, meaning that v not only represents the station but also serves as its ID, *i.e.*, $v \in \{1, \ldots, n\}$.

At most k stations out of the n, where k is *unknown*, may become active, potentially in different time rounds. The communication occurs in synchronous rounds, meaning that the clocks of all stations tick at the same rate. However, the most general assumption is without a *global clock* and *no system-based synchronization*. Each station measures time using its own local clock, which starts at the time round when it is activated, and is independent of other stations' clocks. An instance of the Contention Resolution problem is represented as a pair (T, δ), where $T = \{v_1, v_2, \ldots, v_k\}$ is a set of k stations, and $\delta : T \to \mathbb{N}$ is a function that maps each station v_i to its activation time $R_i = \delta(v_i)$. Without loss of generality $0 = \delta(v_1) \leq \delta(v_2) \leq \cdots \leq \delta(v_k)$.

We consider *non-adaptive* deterministic algorithms for contention resolution (CR), where each station's actions are predefined at the start of execution, based solely on the parameter n and the station's ID. More precisely, for each station v, a *transmission vector* $\mathcal{S}_v$ consists of a sequence of bits that correspond to the time slots of v's local clock. If the r-th bit of $\mathcal{S}_v$ is 1, the station transmits in the r-th slot of its local time; otherwise, it remains silent. A protocol $\mathcal{A}$ for the Contention Resolution problem consists of a collection of n transmission vectors, one for each station.

A protocol $\mathcal{A}$ solves the Contention Resolution problem with *latency* $\tau(\mathcal{A})$ if, for every possible instance (T, δ), it enables every station $v \in T$ to transmit successfully within $\tau(\mathcal{A})$ rounds from its activation time.

The classic version of the problem, as described above, aims at one successful transmission of each activated station. The generalized CR, studied in this work, assumes that each activated station i has a number s_i of packets to be successfully transmitted on the channel, i.e., it needs s_i successful transmissions. Let s be the upper bound on the values of s_i.

B.2   Description of the protocol

All stations are equipped with the same $t \times n$ binary matrix $\mathbf{M}$, which is a (n, α)-URSC, as specified in Definition 2, for an arbitrary constant $\alpha \in (0, 1)$, say $\alpha = 1/2$. We also need the code to satisfy Definition 3; hence, we could take our constructed code as in Theorem 1. We can establish a one-to-one

Communication rounds:	1	2	3	4	$\cdots$

Figure 3: *The 'out of sync' of transmission vectors for stations v_1, v_3, and v_5 following an instance where $\delta(v_1) = 0$, $\delta(v_3) = 2$, and $\delta(v_6) = 3$. The empty squares indicate times when the station was not yet activated.*

correspondence between the set of stations and the columns of matrix $\mathbf{M}$ (see Figure 2). Specifically, each station v is associated with the column at index v in $\mathbf{M}$ (recall that $v \in \{1, \ldots, n\}$).

We define our protocol $\mathcal{A}$ as follows. The transmission vector of any station v is represented by the column at index v, which we denote as $\mathbf{M}(v)$. In other words, for each station v, we set

$$\mathcal{S}_v = \mathbf{M}(v) \cdot \left\lceil \frac{s}{(1-\alpha) \cdot weight_n} \right\rceil ,$$

where $weight_n$ is the minimum of $|M(v)_{\tau_1(n,n), \tau_2(n,n)}|$ over stations v and $\mathbf{M}(v) \cdot x$ denotes concatenation of codeword $M(v)$ x times. Recall that $|M(v)_{\tau_1(n,n), \tau_2(n,n)}|$ denotes the number of 1's in the interval $[\tau_1(n,n), \tau_2(n,n)]$ of codeword $M(v)$, where $\tau_1(n,n), \tau_2(n,n)$ are as in Definition 3.

Example B.1. *Suppose the stations are provided with the matrix depicted in Figure 2. In this scenario, stations v_1, v_3 and v_6 have the following transmission vectors, respectively:*

$$(1, 1, 0, 0, \ldots, 1, 1) , \quad (1, 1, 1, 0, \ldots, 1, 0) , \quad (1, 0, 1, 1, \ldots, 0, 1) .$$

Consider an instance where $\delta(v_1) = 0$, $\delta(v_3) = 2$ and $\delta(v_6) = 3$. The r-th local round of station v_3 is synchronized with round $r + \delta(v_3) - \delta(v_1) = r + 2$ of v_1's local clock and with round $r + \delta(v_3) - \delta(v_6) = r - 1$ of v_6's local clock (see Figure 3).

In general, we can state the following fact.

Fact 1. *For any two stations v and v', the i-th round of station v is synchronized with round $i' = i + \delta(v) - \delta(v')$ of station v'. In particular, if i' is negative, it indicates that v' has not been activated yet.*

B.3 Correctness and complexity

Let us fix an arbitrary instance (T, δ) of the contention resolution problem. Let $k^* = k + c' \cdot \left\lceil \frac{s}{\log n} \right\rceil$ for some suitable constant c' to be specified later. We aim to show that every station $v \in T$ will successfully transmit at least s times within t rounds after activation, where t is $\tau(\mathcal{A}) = \Theta\left(\left(k + \frac{s}{\log n}\right)^2 \log n \right)$. More precisely, if $k^* \leq n$ then $\tau(\mathcal{A})$ is the elongation $\tau(k^*)$ of code M for parameter k^*, and otherwise $\tau(\mathcal{A})$ is the length of M multiplied by $\left\lceil \frac{s}{(1-\alpha) \cdot weight_n} \right\rceil$ (this is by observing that $\frac{k^*}{n} \geq \left\lceil \frac{s}{(1-\alpha) \cdot weight_n} \right\rceil$ for sufficiently large constant c' in the definition of k^*). Let us also generalize definition of $weight_n$ to $weight_\ell$, for any $\ell \leq n$, to be equal to the minimum number of 1's in any codeword of M in the interval $[\tau_1(n,\ell), \tau_2(n,\ell)]$ of positions. We will prove later in Lemma 8 that in our constructed code, $weight_\ell > \frac{3}{5}\sqrt{c}k \ln n = \Omega(k \log n)$.[4] We start with the following lemma.

---

[4]Formally, Lemma 8 proves the lower bound with high probability, but this probability is enough to carry on through the derandomization argument and also easy to check by the algorithm in linear time per codeword.

Lemma 4. *Let (T, δ) be an arbitrary instance of contention resolution problem, and consider a station $v \in T$ that is transmitting during the i-th round of its local clock. Suppose there exists another station $v' \in T \setminus \{v\}$ transmitting simultaneously. Let $\mathbf{c} = \mathbf{M}(v)$ and $\mathbf{y} = \mathbf{M}(v')$. There exists a vector $\mathbf{z} \in S(\mathbf{y})$ such that both $\mathbf{c}$ and $\mathbf{z}$ have a 1 in their i-th position.*

Proof. Let's explicitly represent the bit components of the 2 transmission vectors as $\mathbf{c} = (c_1, c_2, \ldots, c_t)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_t)$. Let $d = \delta(v) - \delta(v')$. As a consequence of Fact 1, the i-th bit component of $\mathbf{c}$ is processed at the same computation round as bit y_{i+d} of $\mathbf{y}$. Since the two stations transmit simultaneously in that computation round, we have $c_i = 1$ and $y_{i+d} = 1$.

It's worth noting that the value of d can be positive, zero, or negative, depending on whether v' is activated before, simultaneously with, or after v, respectively. Therefore, $y_{i+d} = 1$ implies $y_{i \oplus (t+d)} = 1$, where we need to recall that $\oplus$ denotes addition modulo t. Hence, if we set $\mathbf{z} = (y_{1 \oplus (t+d)}, y_{2 \oplus (t+d)}, \ldots, y_{t \oplus (t+d)})$, then both $\mathbf{c}$ and $\mathbf{z}$ have a 1 in their i-th position, and the lemma follows. $\square$

Theorem 5. *Let (T, δ) be an arbitrary instance of the generalized contention resolution problem. Protocol $\mathcal{A}$ solves the generalized contention resolution problem for any $k \leq n$ contenders (with k unknown) and guarantees each of them at least s successful transmissions with latency $\tau(\mathcal{A}) \leq t = O((k + \frac{s}{\log n})^2 \log n)$.*

Proof. Note first that if $k^* > n$ then we consider $\lceil k^*/n \rceil$ concatenations of M, and in each concatenation it is enough to prove the number of submission is $weight_n = \Omega(n \log n)$ (the asymptotics is by Lemma 8). In other words, this corresponds to case $k^* = n$ that occurs $\lceil k^*/n \rceil$ times in the code. Therefore, in the remainder it is sufficient to focus on case $k^* \leq n$.

Let v be an arbitrary station in T. Starting from computation time $\delta(v)$, station v transmits according to its transmission vector $\mathcal{S}_v = \mathbf{M}(v) = (c_1, \ldots, c_t)$: for $j = 1, 2, \ldots, t$ it transmits at time $\delta(v) + j$ if and only if $c_j = 1$.

Let $r = |[\mathbf{c}]_{[\tau_1(n,k^*), \tau_2(n,k^*)]}|$ and $\{j_1, j_2, \ldots, j_r\}$ be the collection of indices from the transmission vector $(c_1, \ldots, c_t)$ where the value is 1. Consequently, for station v, the transmission will happen exclusively during its local rounds $j_1, j_2, \ldots, j_r$. Note that $r \geq weight_{k^*}$, by the definition of $weight_{k^*}$, and by Lemma 8, we get $r \geq \frac{3}{5}\sqrt{c}(k + c'\frac{s}{\log n}) \log n > 2k \log n + 2s$ for sufficiently large constants c, c'. It follows that $r/2 > s$, and consequently, $r - s > r/2$.

Suppose, by contradiction, that less than s of these r transmissions are successful. In this scenario, it implies that for more than $r - s$ of v's local times j_i, for $i = 1, 2, \ldots, r$, there is (at least) another station $v^{j_i} \in T \setminus \{v\}$, associated with v's local time j_i, that transmits simultaneously at v's local time j_i.

Let $\mathbf{c} = \mathbf{M}(v)$ and $\mathbf{y}_i = \mathbf{M}(v^{j_i})$ for $i = 1, 2, \ldots, r$. Applying Lemma 4 repeatedly for $i = 1, 2, \ldots, r$, we get that for each $i = 1, 2, \ldots, r$, there exists a vector $\mathbf{z}_i \in S(\mathbf{y}_i)$ such that both $\mathbf{c}$ and $\mathbf{z}_i$ have a 1 in their i-th position. Consequently, the bitwise OR of all these vectors, that is $\mathbf{z} = \bigvee_{i=1}^{r} \mathbf{z}_i$, must have weight $|\mathbf{z}| > r - s$. By applying the previously proved bound $r - s > r/2$, we get $|\mathbf{z}| > r/2 = r\alpha$. Hence, we have found a vector $\mathbf{z} \in S_{\vee}(T \setminus \{\mathbf{c}\})$ such that $|\mathbf{c} \wedge \mathbf{z}| > |\mathbf{c}| - s > \alpha|\mathbf{c}|$. This contradicts the fact that $\mathbf{M}$ is an ultra-resilient superimposed code satisfying also Definition 3, for which it is required, instead, that for all $\mathbf{z} \in S_{\vee}(T \setminus \{\mathbf{c}\})$, $|\mathbf{c}_{[\tau_1(n,k^*), \tau_2(n,k^*)]} \wedge \mathbf{z}_{[\tau_1(n,k^*), \tau_2(n,k^*)]}| < \alpha|\mathbf{c}_{[\tau_1(n,k^*), \tau_2(n,k^*)]}|$, where $0 < \alpha \leq 1$. The last inequality comes from applying the Collision Weight Inequality (Equation (2) in Definition 3) $k - 1$ times – specifically, to the $k - 1$ superimposed codewords of the contending stations. $\square$

B.4 Other related work on contention resolution

Deterministic contention resolution has been studied in a slightly easier setting where all k stations arrive at the same time. In this scenario, the latency is significantly lower, $\Theta(k \log n)$, with the upper bound established in [24] and a construction given in [25], while the lower bound is proved in [8]. The dynamic case, assuming the availability of a global clock, was examined in [10].

Recently, fairness in contention resolution has emerged as a challenging objective [3]. In our work, we consider the setting with dynamic arrivals and without global clock, where another optimization criterion is the number of transmissions (corresponding to the weight of the codewords). Our approach also shows slight improvement in this aspect, cf. [11].

C Proof of Lemma 2

In proving Lemma 2, we proceed by separately bounding the probabilities of satisfying each of the two inequalities of the Collision Bound Property, and then we combine these bounds in the final proof. This is done in Appendix C.1 and C.2, respectively.

Some preliminary mathematical tools that will be pivotal in the proofs of this section are deferred to the Appendix C.3. This includes the *rearrangement inequality* [18], a fundamental result in combinatorial mathematics essential for bounding and optimizing sums involving products of sequences of probabilities, as well as specific results on bounding summations through integral approximation.

From this point onward, we assume the hypotheses of the lemma. Specifically, in addition to n and α, we fix $\epsilon > 0$ and set $\tau_1(n, k) = \frac{c}{64}k^2 \ln n$ and $\tau_2(n, k) = \frac{c}{\alpha^{2+\epsilon}}k^2 \ln n$, where $c > 0$ is a sufficiently large constant. Moreover, we consider $\mathbf{M} = \mathcal{M}(n, \alpha, \epsilon, c)$ to be any matrix generated according to the random construction of Definition 4, and $1 < k \leq n$ is any given (unknown) parameter.

C.1 Weight Inequality

We begin with inequality (1), which compares the weight of the upper segment to that of the lower segment of a column relative to τ_1 and τ_2. Specifically, our goal here is to prove the following lemma.

Lemma 5. *The probability that the Weight Inequality (1) does not hold is at most* $\frac{2}{c^2} \cdot n^{-8 \ln(4/\alpha)}$.

We need some preliminary result. Let $Y_j^\top$ and $Y_j^\perp$ be the random variables denoting the weight of the subcolumns $[\mathbf{c_j}]_{[0, \tau_1(n,k)]}$ and $[\mathbf{c_j}]_{[\tau_1(n,k), \tau_2(n,k)]}$, respectively.

Lemma 6. *We have*
$$\frac{1}{8}\sqrt{ck}\ln n < E[Y_j^\top] < \frac{1}{2}\sqrt{ck}\ln n$$

Proof. We have
$$E[Y_j^\top] = \sum_{b=0}^{\tau_1(n,k)} p(r) = \ln n \sum_{b=0}^{(c/64)k^2} \frac{1}{\sqrt{b+1}}.$$

We can find upper and lower bounds for $\frac{1}{\sqrt{b+1}}$ as follows.

$$\frac{\left(\sqrt{b+2} + \sqrt{b+1}\right)\left(\sqrt{b+2} - \sqrt{b+1}\right)}{\sqrt{b+1}} = \frac{1}{\sqrt{b+1}} = \frac{\left(\sqrt{b+1} + \sqrt{b}\right)\left(\sqrt{b+1} - \sqrt{b}\right)}{\sqrt{b+1}}.$$

From which we get

$$2\left(\sqrt{b+2} - \sqrt{b+1}\right) < \frac{1}{\sqrt{b+1}} < 2\left(\sqrt{b+1} - \sqrt{b}\right). \tag{4}$$

Hence,

$$
\begin{aligned}
2\ln n \left(\sqrt{(c/64)k^2 + 2} - 1\right) &< E[Y_j^\top] < 2\ln n \left(\sqrt{(c/64)k^2 + 1}\right), \\
\ln n \left(\sqrt{(c/64)k^2}\right) &< E[Y_j^\top] < 4\ln n \left(\sqrt{(c/64)k^2}\right), \\
\frac{1}{8}\sqrt{ck}\ln n &< E[Y_j^\top] < \frac{1}{2}\sqrt{ck}\ln n.
\end{aligned}
$$

$\square$

Analogously, we can now estimate the expected value of $Y_j^\perp$.

Lemma 7. *We have*

$$\left(\frac{1}{\alpha^{1+\epsilon/2}} - \frac{1}{8}\right)\sqrt{c}k\ln n < E[Y_j^\perp] < 4\left(\frac{1}{\alpha^{1+\epsilon/2}}\right)\sqrt{c}k\ln n.$$

Proof. We proceed as in the proof of Lemma 6. We have

$$E[Y_j^\perp] = \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r) = \ln n \sum_{b=(c/64)k^2}^{(c/\alpha^{2+\epsilon})k^2} \frac{1}{\sqrt{b+1}}.$$

Using the inequalities (4), we have

$$\ln n \sum_{b=(c/64)k^2}^{(c/\alpha^{2+\epsilon})k^2} 2\left(\sqrt{b+2} - \sqrt{b+1}\right) \; < E[Y_j^\perp] < \; \ln n \sum_{b=(c/64)k^2}^{(c/\alpha^{2+\epsilon})k^2} 2\left(\sqrt{b+1} - \sqrt{b}\right),$$

$$2\ln n\left(\sqrt{(c/\alpha^{2+\epsilon})k^2 + 2} - \sqrt{(c/64)k^2 + 1}\right) \; < E[Y_j^\perp] < \; 2\ln n\left(\sqrt{(c/\alpha^{2+\epsilon})k^2 + 1} - \sqrt{(c/64)k^2}\right),$$

$$\ln n\left(\sqrt{(c/\alpha^{2+\epsilon})k^2} - \sqrt{(c/64)k^2}\right) \; < E[Y_j^\perp] < \; 4\ln n\sqrt{(c/\alpha^{2+\epsilon})k^2},$$

$$\ln n\left(\frac{\sqrt{c}\,k}{\alpha^{1+\epsilon/2}} - \frac{\sqrt{c}\,k}{8}\right) \; < E[Y_j^\perp] < \; 4\ln n\frac{\sqrt{c}\,k}{\alpha^{1+\epsilon/2}},$$

$$\left(\frac{1}{\alpha^{1+\epsilon/2}} - \frac{1}{8}\right)\sqrt{c}k\ln n \; < E[Y_j^\perp] < \; 4\left(\frac{1}{\alpha^{1+\epsilon/2}}\right)\sqrt{c}k\ln n.$$

$\square$

Since the random construction of the matrix (Definition 4) sets each entry of the columns to 1 or 0 independently, both $E[Y_j^\top]$ and $E[Y_j^\perp]$ can be considered as sums of independent random variables. Consequently, we can apply the Chernoff bound to evaluate the deviation of $Y_j^\top$ and $Y_j^\perp$ respectively above the expected value $E[Y_j^\top]$ and below the expected value $E[Y_j^\perp]$.

Lemma 8. *We have,*

$$\Pr\left(Y_j^\top > \frac{3}{5}\sqrt{c}\,k\ln n\right) < \frac{1}{c^2}\cdot n^{-8\ln(4/\alpha)},$$

$$\Pr\left(Y_j^\perp < \frac{7}{10}\left(\frac{1}{\alpha^{1+\epsilon/2}} - \frac{1}{8}\right)\sqrt{c}\,k\ln n\right) < \frac{1}{c^2}\cdot n^{-8\ln(4/\alpha)}.$$

Proof. Letting $\delta = 1/5$, and recalling the upper and lower bounds for $E[Y_j^\top]$ established in Lemma 6,

we have:

$$
\begin{aligned}
\Pr\left(Y_j^\top > \frac{3}{5}\sqrt{c}\,k\ln n\right) \;&\le\; \Pr\left(Y_j^\top \ge \frac{6}{5}\cdot E[Y_j^\top]\right) \\
&=\; \Pr\left(Y_j^\top \ge (1+\delta)E[Y_j^\top]\right) \\
&\le\; \exp\left(-\frac{\delta^2 E[Y_j^\top]}{3}\right) \quad \text{(see, for example, Eq. (4.2) in [28])} \\
&\le\; \exp\left(-\frac{(1/5)^2\sqrt{c}\,k\ln n}{24}\right) \\
&\le\; \exp\left(-\frac{(1/5)^2\sqrt{c}\ln(1/\alpha)\ln n}{24}\right) \quad \text{(recalling that } e^{-k} < \alpha \le 1) \\
&\le\; \exp\left(-\frac{(1/5)^2\sqrt{c}\left(\frac{\ln(4/\alpha)}{2} - \frac{\ln 4}{2}\right)\ln n}{24}\right) \\
&<\; \frac{1}{c^2}\cdot n^{-8\ln(4/\alpha)},
\end{aligned}
$$

where the last inequality holds for a sufficiently large constant c. Analogously, we can now let $\delta = 3/10$ and use the lower bound for $E[Y_j^\perp]$ established in Lemma 7.

$$
\begin{aligned}
\Pr\left(Y_j^\perp < \frac{7}{10}\cdot\left(\frac{1}{\alpha^{1+\epsilon/2}} - \frac{1}{8}\right)\sqrt{c}k\ln n\right) \;&\le\; \Pr\left(Y_j^\perp \le \frac{7}{10}\cdot E[Y_j^\perp]\right) \\
&=\; \Pr\left(Y_j^\perp \le (1-\delta)\,E[Y_j^\perp]\right) \\
&\le\; \exp\left(-\frac{\delta^2\left(\frac{1}{\alpha} - \frac{1}{8}\right)\sqrt{c}k\ln n}{2}\right) \\
&\quad \text{(see, for example, Eq. (4.5) in [28])} \\
&\le\; \exp\left(-\frac{\delta^2\sqrt{c}\ln(1/\alpha)\ln n}{16}\right) \\
&\quad \text{(recalling that } e^{-k} < \alpha \le 1) \\
&\le\; \exp\left(-\frac{(3/10)^2\sqrt{c}\left(\frac{\ln(4/\alpha)}{2} - \ln 2\right)\ln n}{16}\right) \\
&<\; \frac{1}{c^2}\cdot n^{-8\ln(4/\alpha)},
\end{aligned}
$$

where the last inequality holds for a sufficiently large constant c. $\qquad\square$

We can now return to the Weight Inequality and prove Lemma 5.

Proof of Lemma 5. Let $w_1 = \frac{3}{5}\sqrt{c}\,k\ln n$ and $w_2 = \frac{7}{10}\left(\frac{1}{\alpha^{1+\epsilon/2}} - \frac{1}{8}\right)\sqrt{c}k\ln n$. Recalling that $\alpha \le 1$, we can observe that

$$
\begin{aligned}
w_1 \;&=\; \frac{3}{5}\sqrt{c}\,k\ln n \\
&<\; \frac{7}{10}\frac{7}{8}\sqrt{c}\,k\ln n \\
&\le\; \alpha\cdot\frac{7}{10}\left(\frac{1}{\alpha} - \frac{1}{8}\right)\sqrt{c}\,k\ln n \\
&\le\; \alpha\cdot\frac{7}{10}\left(\frac{1}{\alpha^{1+\epsilon/2}} - \frac{1}{8}\right)\sqrt{c}k\ln n \\
&=\; \alpha\,w_2.
\end{aligned}
$$

In view of this, the inequality (1) is guaranteed to be satisfied if the following events occur simultaneously: $(Y_j^\top \leq w_1)$ and $(Y_j^\perp \geq w_2)$. Hence, by applying Lemma 8, the inequality is not satisfied with a probability of at most

$$\Pr(Y_j^\top > w_1) + \Pr(Y_j^\perp < w_2) < \frac{2}{c^2} \cdot n^{-8\ln(4/\alpha)} \ .$$

$\square$

C.2 Collision Weight Inequality

Now it is the turn of the Collision Weight Inequality. For $d \in \{-1, 0, 1\}$ and $0 \leq i \leq t - 1$, let $X_{j,j'}^d(i)$ be the random variable denoting $|[\mathbf{c_j}] \wedge [\mathbf{c_{j'}}](i + d + t)|$. The left-hand side of inequality (2) is a random variable $X_{j,j'}(i)$ such that

$$X_{j,j'}(i) \leq X_{j,j'}^{-1}(i) + X_{j,j'}^0(i) + X_{j,j'}^1(i) \ .$$

Our goal is to show that the two random variables $X_{j,j'}(i)$ and $Y_j^\perp$ satisfy the Collision Weight Inequality. The first step is to determine an upper bound on the expectation of $Xj, j'(i)$. We can observe that

$$E[X_{j,j'}(i)] \leq \sum_{d=-1,0,1} E[X_{j,j'}^d(i)].$$

For $d \in \{-1, 0, 1\}$, we have

$$E[X_{j,j'}^d(i)] = \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r)\, p\left((r + i + d) \bmod t\right).$$

The value of $E[X_{j,j'}^d(i)]$ is significantly influenced by the size of $p\left((r + i + d) \mod t\right)$, which can vary considerably within the interval $[\tau_1(n, k), \tau_2(n, k)]$ depending on the shift $0 \leq i < t$. In this context, the following three cases can occur:

- **Case 1:** For all r in the interval $\tau_1(n, k) \leq r \leq \tau_2(n, k)$, we have $p((r + i + d) \mod t) > \frac{\alpha^{1+\epsilon/2}}{\sqrt{c}\, k} \ln(4/\alpha)$. In other words, all probabilities of the shifted column are large within the considered positions.

- **Case 2:** For $\lambda = \frac{\sqrt{c}}{\ln(4/\alpha)}, \frac{\sqrt{c}}{\ln(4/\alpha)} + 1, \ldots, \frac{\sqrt{c}n}{2k}$, and all r in the interval $\tau_1(n, k) \leq r \leq \tau_2(n, k)$, we have $\frac{\alpha^{1+\epsilon/2}}{2\lambda k} \leq p((r + i + d) \mod t) \leq \frac{\alpha^{1+\epsilon/2}}{\lambda k}$. In other words, all probabilities of the shifted column are relatively small within the considered positions, and thus, from the definition of blocks, which are longer than $\tau_2(n, k) - \tau_1(n, k)$ for the considered probability, all probabilities are within a constant factor from each other.

- **Case 3:** There is some $\tau' \in (\tau_1(n, k), \tau_2(n, k)]$ such that: for all r in the interval $\tau_1(n, k) \leq r \leq \tau'$, we have $p((r + i + d) \mod t) < \frac{\alpha^{1+\epsilon/2}}{2\sqrt{c}\, n}$, and for all $\tau' \leq r \leq \tau_2(n, k)$ we have $p((r + i + d) \mod t) > \frac{\alpha^{1+\epsilon/2}}{\sqrt{c}\, k} \ln(4/\alpha)$. In other words, all probabilities of the shifted column are large starting from some intermediate position τ', and smallest possible before that.

The following lemmas establish upper and lower bounds on the expectation of $X_{j,j'}^d(i)$ for each of the fist two cases; case 3 is a simple combination of case 2 in some prefix of $[\tau_1(n, k), \tau_2(n, k)]$ and case 1 in the remaining suffix.

Lemma 9. *In case 1, we have*

$$\frac{\ln n \ln(4/\alpha)}{2} < E[X_{j,j'}^d(i)] < \ln n \left(3 + \ln\left(\frac{64}{\alpha^{2+\epsilon}}\right)\right).$$

Proof. Let us begin with the lower bound. We have,

$$
\begin{aligned}
E[X_{j,j'}^d(i)] \;&>\; \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r)\, \frac{\alpha^{1+\epsilon/2}}{\sqrt{c}\,k}\ln(4/\alpha) \\[2mm]
&=\; \frac{\alpha^{1+\epsilon/2}}{\sqrt{c}\,k}\ln(4/\alpha) \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r) \\[2mm]
&>\; \frac{\alpha^{1+\epsilon/2}}{\sqrt{c}\,k}\ln(4/\alpha) \cdot E[Y_j^\perp] \\[2mm]
&>\; \frac{\alpha^{1+\epsilon/2}}{\sqrt{c}\,k}\ln(4/\alpha) \left(\frac{1}{\alpha^{1+\epsilon/2}} - \frac{1}{8}\right)\sqrt{c}k\ln n \quad \text{(by Lemma 7)} \\[2mm]
&>\; \frac{\ln n \ln(4/\alpha)}{2}.
\end{aligned}
$$

The upper bound is more complex because in this case we need to consider all possible shifts of the second column, j'. To address this, we will use the rearrangement inequality.

Let $P^{i+d} = \{p((r+i+d) \bmod t) \mid \tau_1(n,k) \le r \le \tau_2(n,k)\}$, and let $y_{\tau_1(n,k)}^{i+d}, \ldots, y_{\tau_2(n,k)}^{i+d}$ denote the elements of P^{i+d} arranged in decreasing order. Formally, for $d \in \{-1,0,1\}$, we have defined a set $\{y_{\tau_1(n,k)}^{i+d}, \ldots, y_{\tau_2(n,k)}^{i+d}\} = P^{i+d}$, where $y_{\tau_1(n,k)}^{i+d} \ge \cdots \ge y_{\tau_2(n,k)}^{i+d}$. By the rearrangement inequality (see Theorem 6) we have that

$$
\sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r)\, p((r+i+d)\bmod t) \;\le\; \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r)\, y_r^{i+d}.
$$

Moreover, since for $\tau_1(n,k) \le r \le \tau_2(n,k)$, $y_r^d \le p(r - \tau_1(n,k))$, then for $d = -1,0,1$:

$$
\begin{aligned}
E[X_{j,j'}^d(i)] \;&\le\; \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r)\, y_r^{i+d} \\[2mm]
&\le\; \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r)\, p(r - \tau_1(n,k)) \\[2mm]
&=\; \ln n \sum_{b=(c/64)k^2}^{(c/\alpha^{2+\epsilon})k^2} \frac{1}{\sqrt{b+1}} \cdot \frac{1}{\sqrt{b-(c/64)k^2+1}} \\[2mm]
&=\; \ln n \sum_{b=(c/64)k^2+1}^{(c/\alpha^{2+\epsilon})k^2+1} \frac{1}{\sqrt{b}} \cdot \frac{1}{\sqrt{b-(c/64)k^2}} \\[2mm]
&<\; \ln n \left(3 + \ln\left(\frac{ck^2}{\alpha^{2+\epsilon}}+1\right) - \ln\left(\frac{ck^2}{64}+1\right)\right) \quad \text{(by Lemma 13)} \\[2mm]
&<\; \ln n \left(3 + \ln\left(\frac{64}{\alpha^{2+\epsilon}}\right)\right).
\end{aligned}
$$

$\square$

Lemma 10. *In case 2, we have that for* $\lambda = \frac{\sqrt{c}}{\ln(4/\alpha)}, \frac{\sqrt{c}}{\ln(4/\alpha)}+1, \ldots, \frac{\sqrt{c}n}{2k}$:

$$
\frac{\sqrt{c}\ln n}{4\lambda} \;<\; E[X_{j,j'}^d(i)] \;<\; \frac{2\sqrt{c}\ln n}{\lambda}.
$$

26

Proof. In this case, we have,

$$
\begin{aligned}
E[X_{j,j'}^d(i)] &\geq \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r)\,\frac{\alpha^{1+\epsilon/2}}{2\lambda k} \\
&= \frac{\alpha^{1+\epsilon/2}}{2\lambda k} \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r) \\
&= \frac{\alpha^{1+\epsilon/2}}{2\lambda k} \cdot E[Y_j^\perp] \\
&> \frac{\alpha^{1+\epsilon/2}}{2\lambda k}\left(\frac{1}{\alpha^{1+\epsilon/2}}-\frac{1}{8}\right)\sqrt{ck}\ln n \quad \text{(by Lemma 7)} \\
&> \frac{\sqrt{c}\ln n}{4\lambda}.
\end{aligned}
$$

Analogously,

$$
\begin{aligned}
E[X_{j,j'}^d(i)] &\leq \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r)\,\frac{\alpha^{1+\epsilon/2}}{\lambda k} \\
&= \frac{\alpha^{1+\epsilon/2}}{\lambda k} \sum_{r=\tau_1(n,k)}^{\tau_2(n,k)} p(r) \\
&= \frac{\alpha^{1+\epsilon/2}}{2\lambda k} \cdot E[Y_j^\perp] \\
&< \frac{\alpha^{1+\epsilon/2}}{2\lambda k}\cdot 4\left(\frac{1}{\alpha^{1+\epsilon/2}}\right)\sqrt{ck}\ln n \\
&= \frac{2\sqrt{c}\ln n}{\lambda}.
\end{aligned}
$$

$\square$

The following lemma guarantees that the Collision Weight Inequality will be satisfied with high probability.

Lemma 11. *The probability that the Collision Weight Inequality does not hold is at most*

$$
\frac{4}{c^2}\cdot n^{-8\ln(4/\alpha)}.
$$

Proof. Let $w = \frac{7}{10}\left(\frac{1}{\alpha^{1+\epsilon/2}}-\frac{1}{8}\right)\sqrt{ck}\ln n$. The Collision Weight Inequality is guaranteed to be satisfied if the following two events occur simultaneously:

$$
X_{j,j'}(i) \leq \left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor,
$$

$$
\left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor \leq \left\lfloor \frac{\alpha\,Y_j^\perp-1}{k-1} \right\rfloor.
$$

Hence, it is not satisfied with probability at most

$$
\Pr\left(X_{j,j'}(i) > \left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor\right) + \Pr\left(\left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor < \left\lfloor \frac{\alpha\,Y_j^\perp-1}{k-1} \right\rfloor\right)
$$

$$
< \; 3\Pr\left(X_{j,j'}^d(i) > \frac{1}{3}\left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor\right) + \frac{1}{c^2}\cdot n^{-8\ln(4/\alpha)}, \tag{5}
$$

27

where the last inequality follows by the second inequality of Lemma 8.

To complete the proof, it remains to analyze the probability that $X_{j,j'}^d(i) > \frac{1}{3}\left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor$.

Substituting w to the right-hand-side of this inequality, we have

$$
\begin{aligned}
\frac{1}{3}\left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor
&> \frac{1}{3}\cdot\alpha\left(\frac{7}{10}\left(\frac{1}{\alpha^{1+\epsilon/2}}-\frac{1}{8}\right)\sqrt{c}\ln n\right)\\
&= \frac{1}{3}\frac{7}{10}\left(\frac{1}{\alpha^{\epsilon/2}}-\frac{\alpha}{8}\right)\sqrt{c}\ln n\\
&> \frac{1}{10}\left(\frac{1}{\alpha^{\epsilon/2}}\right)\sqrt{c}\ln n. \qquad\qquad\qquad (6)
\end{aligned}
$$

Because each entry of a column is independently set to either 1 or 0 in the random construction of the matrix, $X_{j,j'}^d(i)$ for $d=-1,0,1$ can be viewed as a sum of independent random variables. Therefore, we can utilize the Chernoff bound to assess the probability that $X_{j,j'}^d(i)$ exceeds its expected value. We need to analyze each of the three cases defined at the beginning of subsection C.2.

Case 1. Let $m=(1+\delta)\ln n\left(3+\ln\left(\frac{64}{\alpha^{2+\epsilon}}\right)\right)$, with $\delta=\ln c$. It holds that

$$
m \le \frac{1}{10}\left(\frac{1}{\alpha^{\epsilon/2}}\right)\sqrt{c}\ln n < \frac{1}{3}\left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor \text{ (by (6))}.
$$

Applying the upper bound of Lemma 9, we also have

$$
(1+\delta)E[X_{j,j'}^d(i)] < m \le \frac{1}{3}\left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor.
$$

Therefore,

$$
\begin{aligned}
\Pr\left(X_{j,j'}^d(i) > \frac{1}{3}\left\lfloor \frac{\alpha\,w-1}{k-1} \right\rfloor\right)
&\le \Pr\left(X_{j,j'}^d(i) \ge m\right)\\
&= \Pr\left(X_{j,j'}^d(i) \ge (1+\delta)E[X_{j,j'}^d(i)]\right)\\
&\le \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{E[X_{j,j'}^d(i)]} \quad\text{(see, for example, Eq. (4.1) in [28])}\\
&\le \exp\left(-\frac{\delta^2 E[X_{j,j'}^d(i)]}{2+\delta}\right)\quad\left(\text{because }\ln(1+\delta)\ge\frac{2\delta}{2+\delta}\right)\\
&\le \exp\left(-\frac{\ln c\,E[X_{j,j'}^d(i)]}{2}\right)\\
&\le \exp\left(-\frac{\ln c\ln n\ln(4/\alpha)}{4}\right)\quad\text{(by the lower bound of Lemma 9)}\\
&= \frac{1}{c^2}\cdot n^{-8\ln(4/\alpha)},
\end{aligned}
$$

where the last inequalities hold for sufficiently large c. Plugging this bound into (5) the lemma is proved for case 1.

Case 2. Let λ be any value in $\left\{\frac{\sqrt{c}}{\ln(4/\alpha)}, \frac{\sqrt{c}}{\ln(4/\alpha)}+1,\ldots,\frac{\sqrt{c}n}{2k}\right\}$. Let $m=(1+\delta)\frac{2\sqrt{c}\ln n}{\lambda}$, where $\delta=\ln c$. We can observe that for any $\epsilon>0$ the following inequality holds for every value of λ in $\left\{\frac{\sqrt{c}}{\ln(4/\alpha)}, \frac{\sqrt{c}}{\ln(4/\alpha)}+1,\ldots,\frac{\sqrt{c}n}{2k}\right\}$:

$$
m \le \frac{1}{10}\left(\frac{1}{\alpha^{\epsilon/2}}\right)\sqrt{c}\ln n.
$$

Applying the upper bound of Lemma 10 and (6) respectively on the left-hand-side and the right-hand-side of this inequality, we also get

$$(1 + \delta)E[X^d_{j,j'}(i)] < m \leq \frac{1}{3}\left\lfloor \frac{\alpha w - 1}{k - 1} \right\rfloor.$$

Therefore,

$$
\begin{aligned}
\Pr\left(X^d_{j,j'}(i) > \frac{1}{3}\left\lfloor \frac{\alpha w - 1}{k - 1} \right\rfloor\right) &\leq \Pr(X^d_{j,j'}(i) > m) \\
&\leq \Pr(X^d_{j,j'}(i) > (1 + \delta)E[X^d_{j,j'}(i)]) \\
&\leq \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}}\right)^{E[X^d_{j,j'}(i)]} \quad \text{(see, for example, Eq. (4.1) in [28])} \\
&\leq \exp\left(-\frac{\delta^2 E[X^d_{j,j'}(i)]}{2 + \delta}\right) \quad \left(\text{because } \ln(1 + \delta) \geq \frac{2\delta}{2 + \delta}\right) \\
&\leq \exp\left(-\frac{\ln c}{2} \frac{\sqrt{c}\ln n}{4\lambda}\right) \quad \text{(by the lower bound of Lemma 10)} \\
&\leq \exp\left(-\frac{\ln c \ln n \ln(4/\alpha)}{8}\right) \\
&\leq \frac{1}{c^2} \cdot n^{-8\ln(4/\alpha)},
\end{aligned}
$$

where the last steps hold for sufficiently large c. Plugging this bound into (5) the lemma is proved for case 2.

Case 3. It follows directly from applying the above analysis of case 2 for $\lambda = \frac{\sqrt{c}n}{2k}$ to the interval $[\tau_1(n, k), \tau']$ and the analysis of case 1 to the interval $[\tau', \tau_2(n, k)]$. Then, we take the union bound of the two events. □

Now, we are ready to complete the proof of Lemma 2.

Proof of Lemma 2. The lemma follows by applying the union bound to the results of Lemmas 5 and 11. □

C.3 Auxiliary results

The primary advantage of the Collision Bound Property is its ability to reduce a property concerning subsets of k matrix columns to one involving only pairs of columns. The following result from combinatorial mathematics plays a crucial role in computing the expected number of positions where two arbitrary columns both contain a 1 (collision). Specifically, it helps to address the challenge posed by arbitrary shifts of the columns during these computations. In our context, the sequences of real numbers $x_1 \leq \cdots \leq x_n$ and $y_1 \leq \cdots \leq y_n$ represent the probabilities of encountering a 1 at corresponding positions in two columns. The rearrangement inequality allows us to handle the probability of collisions independently of the misalignment caused by arbitrary shifts between the two sequences.

Theorem 6 (Theorem 368 (Rearrangement Inequality, [18])). *For any choice of real numbers*

$$x_1 \leq \cdots \leq x_n \quad and \quad y_1 \leq \cdots \leq y_n$$

and every permutation $y_{\sigma(1)} \leq \cdots \leq y_{\sigma(n)}$ of $y_1 \leq \cdots \leq y_n$,

$$x_1 y_n + \cdots + x_n y_1 \leq x_1 y_{\sigma(1)} + \cdots + x_1 y_{\sigma(n)} \leq x_1 y_1 + \cdots + x_n y_n. \tag{7}$$

The analysis of column-wise collisions is further facilitated by Lemma 13, which aims to bound the sum of collisions between two shifted columns within specific rows of the matrix. The following lemma is preparatory to Lemma 13.

Lemma 12. *For $x > a$, it holds that*

$$\int \frac{1}{\sqrt{x(x-a)}}\, dx = 2\ln\left|\sqrt{x-a} + \sqrt{x}\right| + C.$$

Proof. Letting $u = \sqrt{x}$, we get $x = u^2$ and $dx = 2u\, du$. Substituting these into the integral, we get:

$$\int \frac{1}{\sqrt{x(x-a)}}\, dx = \int \frac{1}{u}\cdot\frac{1}{\sqrt{u^2-a}}\cdot 2u\, du = 2\int \frac{1}{\sqrt{u^2-a}}\, du. \tag{8}$$

This is a standard integral (see *e.g.* entry 2.271 (4) in [17]), and it is known that:

$$\int \frac{1}{\sqrt{u^2-a}}\, du = \ln\left|\sqrt{u^2-a}+u\right| + C.$$

Substituting back $u = \sqrt{x}$ and plugging the result into equation (8), we obtain the lemma.

$\square$

Lemma 13. *For $1 < h_1 < h_2$, it holds that*

$$\sum_{i=h_1+1}^{h_2+1} \frac{1}{\sqrt{i(i-h_1)}} < 3 + \ln(h_2+1) - \ln(h_1+1).$$

Proof. For $x > h_1$, the function $f(x) = \frac{1}{\sqrt{x(x-h_1)}}$ is monotonic decreasing. Hence, we can apply the Riemann sum approximation as follows:

$$
\begin{aligned}
\sum_{i=h_1+1}^{h_2+1} \frac{1}{\sqrt{i(i-h_1)}} &= \frac{1}{\sqrt{(h_1+1)}} + \sum_{i=h_1+2}^{h_2+1} \frac{1}{\sqrt{i(i-h_1)}} \\
&< 1 + \sum_{i=h_1+2}^{h_2+1} \frac{1}{\sqrt{i(i-h_1)}} \\
&\leq 1 + \int_{h_1+1}^{h_2+1} \frac{1}{\sqrt{x(x-h_1)}}\, dx \\
&= 1 + \left[2\ln\left(\sqrt{x-h_1} + \sqrt{x}\right)\right]_{h_1+1}^{h_2+1} \quad \text{(by Lemma 12)} \\
&= 1 + 2\ln\left(\sqrt{h_2+1-h_1} + \sqrt{h_2+1}\right) - 2\ln\left(1 + \sqrt{h_1+1}\right) \\
&< 1 + 2\ln\left(2\sqrt{h_2+1}\right) - 2\ln\left(\sqrt{h_1+1}\right) \\
&= 1 + 2\ln 2 + \ln(h_2+1) - \ln(h_1+1).
\end{aligned}
$$

$\square$

D Analysis of the Neighborhood-Learning Algorithm (Algorithm 2) and the Proof of Theorem 3

Lemma 14 (Safety). *If a node v adds node w to set N_v^*, then w is a neighbor of v in the underlying network G.*

Proof. The proof is by contradiction – suppose that in some network and adversarial wake-up schedule, v is not a neighbor of w but v puts w to its set N_v^*. It can happen only if v heard at some point a binary sub-sequence equal to the block ID of node w, denoted $\mathbf{b}_w$. To focus our attention, we denote that particular sub-sequence of σ_v by γ; it is identical to $\mathbf{b}_w$, and it occurs in particular time. Since w is not a neighbor of v, this heard sequence γ comes from activities (beeping or not beeping) of actual neighbors of v after awakening.

Let $w_1, \ldots, w_\ell$ be the minimal, in the sense of inclusion, subset of neighbors of v that contribute to the considered sequence γ heard by node v, for some integer $\ell \geq 1$. More precisely, each bit 1 in γ comes from beep of some (at least one) of the nodes $w_1, \ldots, w_\ell$ and none of them beeps in round corresponding to bit 0 in γ. By minimality of set $w_1, \ldots, w_\ell$, each of them has some of its block IDs overlapping sequence γ, as otherwise it would only have blocks of zeros overlapping and we could have removed it from set $w_1, \ldots, w_\ell$.

Claim 1. *Each such overlapping block ID of nodes $w_1, \ldots, w_\ell$ must actually be aligned (in time) with γ, unless it finishes before round 7 of γ or starts in the last round of γ.*

Proof of Claim 1: Suppose, to the contrary, that some overlapping block ID of some w_i, for $1 \leq i \leq \ell$, does not finish before round 7 of γ and does not start at the very last round of γ. We consider two complementary cases and obtain contradiction in each of them.

If this block ID starts before round 1 of γ, we observe that during the rounds $4, 5, 6$ of γ, node w_i has to beep at least once. This is because these rounds align with rounds $j, j+1, j+2$ of the block ID of w_i, for some $j \geq 5$, and in every three of such rounds there is at least one beep (by the definition of the block ID). We obtain contradiction with the fact that there is no beep in γ, which value is the same as the block ID $\mathbf{b}_w$ of w, in the considered rounds $4, 5, 6$ (see again the definition of block ID).

In the remaining case, i.e., when the considered block ID of w_i overlapping γ starts after round 1 and before the last round of γ, the first three consecutive 1's of that block would cause a beep at some position of γ that stores 0. This is because one of the following arguments, and obviously yields contradiction: (i) in the subsequence of γ, starting after position 1, there are no three consecutive 1's, or (ii) if the block actually starts in the second-last round of γ, in the last two positions of γ there is at least one 0. $\square$

It follows from Claim 1 that at least one of the overlapping block IDs of $w_1, \ldots, w_\ell$ must be aligned with γ, because otherwise none of them would cause the beeps in some positions $7, \ldots, 7 + 2 \log n - 1$ of γ. W.l.o.g. we could assume that the block ID of w_1 is aligned with γ. Recall that $w_1 \neq w$, because w_1 is a neighbor of v and w is not. Hence, there is a position j in the ids of w_1 and w on which they differ. It implies that their block IDs are different at positions $7 + 2(j-1) + 1$ and $7 + 2(j-1) + 2$. In particular, in one of these position there is 0 in γ and 1 in the considered block ID of w_1, which contradicts the definition of γ and the set $w_1, \ldots, w_\ell$; more precisely, by definition, if γ stores 0 at some position then all $w_1, \ldots, w_\ell$ did not beep at the corresponding round. $\square$

Lemma 15 (Inclusion). *If a node w is a neighbor of node v, then the block ID of w can be found in some $\sigma_v[i, \ldots, i + (7 + 2 \log n) - 1]$, for some $1 \leq i \leq (t-1) \cdot (7 + 2 \log n) + 1$, checked by v at most $2t \cdot (7 + 2 \log n)$ rounds after both v and w are awaken.*

Proof. Consider a node v and its neighbors $w_1, \ldots, w_\ell$, for some integer $0 \leq \ell < \Delta$, whose beeping overlap the execution of the algorithm by node v. Consider one of the neighbors of v, w.l.o.g. w_1. Consider the first round when both v and w_1 are already awaken and w_1 starts its periodic repetition (line 6). In what follows, for the sake of simplifying notation, we will be numbering rounds, starting from the considered round, $1, 2, \ldots$. (Note that such round occurs less than $t \cdot (7 + 2 \log n)$ rounds after both these nodes are awaken.)

We define checkpoints at rounds $1 + (i-1) \cdot (7 + 2 \log n)$, for $i = 1, \ldots, t$. (Recall that t is the length of the ultra-resilient superimposed code $\mathbf{M}$.) For every node $v, w_1, \ldots, w_\ell$, at each checkpoint i we define the bit i of its sequence to be equal to 1 if the node executes its block ID, and 0 otherwise (i.e., if it beeps its block of zeros). Observe that the resulting sequence is a shifted codeword of that node, and

the sequence for w_1 is its original codeword $\mathbf{c}_v$ (i.e., not shifted, because we assumed that it starts its periodic procedure at round we number 1). Here by "shifted" we consider both: shifted cyclically, as in Definition 2, or "shifted not cyclically" in which we pad positions before the shift value with zeros. The latter corresponds to the case when a neighbor has been awaken during the considered interval. Note that if the Definition 2 holds for a configuration of codewords in some set T, shifted cyclically, it also holds if instead of cyclic shifts we pad positions before the shift values with zeros instead. This is because it may only create more zeros in the definition of vector $\mathbf{z}^*$, which is even better from perspective of the guaranteed equation in Definition 2.

By the fact that code $\mathbf{M}$ from Theorem 2 satisfies Definition 2 of $(n, 3/4)$-URSC with elongation guarantees for any $k \leq \Delta$ (this is because $\tau(n, \Delta) = t$ and $\Delta \leq n$), there is a position i such that: the codeword $\mathbf{c}_{w_1}$ has 1 at this position, while the other considered shifted codewords of $v, w_2, \ldots, w_\ell$ have zeros at position i and $i + 1$. Here we use the isolation property together with shift resilience. It means that in the interval $[1 + (i-1) \cdot (7 + 2\log n), i \cdot (7 + 2\log n)]$ node w_1 beeps its block ID, while nodes $v, w_2, \ldots, w_\ell$ stay idle. The latter holds because by the definition of checkpoint i, taken in round $1 + (i-1) \cdot (7+2\log n)$, all these nodes have their $\mathbf{c}_v[i] = \mathbf{c}_{w_2}[i] = \ldots = 0$ and thus execute their blocks of zeros in that round; even if they finish their blocks early after that round, because they all have also $\mathbf{c}_v[i+1] = \mathbf{c}_{w_2}[i+1] = \ldots = 0$, they start executing another block of zeros (each of length $7 + 2\log n$), hence none of them ever beeps in the considered interval $[1 + (i - 1) \cdot (7 + 2\log n), i \cdot (7 + 2\log n)]$.

Since we assumed that v is alive in the considered rounds, it will record the beeps heard in the interval $[1 + (i-1) \cdot (7+2\log n), i \cdot (7+2\log n)]$. As we argued above, only its neighbor w_1 is beeping (its own block ID) in this period, among the neighbors of v and v itself, therefore σ_v in this interval will be equal to the block ID of w_1. Hence, w_1 will be added to set N_v^* in the end of the considered interval (line 15). This happened at most $t \cdot (7 + 2\log n)$ rounds after the considered round we numbered 1, because it takes place within the considered periodic procedure (lines 6 - 16) of lenth $t \cdot (7 + 2\log n)$. As argued earlier, the considered round, from which we start counting in this analysis, occurs less than $t \cdot (7 + 2\log n)$ rounds after both v and w_1 are awaken; hence, in less than $2t \cdot (7 + 2\log n)$ rounds after both v and w_1 are awaken, v updates its set N_v^* by adding w, which block ID it has heard meanwhile. $\square$

Proof of Theorem 3. The existence of code $\mathbf{M}$ used in Algorithm 2, constructable in polynomial time, is guaranteed by Theorem 2 in Section 3.

Safety property comes from Lemma 14, while Inclusion property – from Lemma 15. Note that sets N_v^* are monotonic, in the sense of inclusion – nothing is ever removed from them. Time complexity also follows from Lemma 15, even in the stronger sense that each node learns its neighbor at most $2t \cdot (7 + 2\log n) \leq O(\Delta^2 \log^2 n)$ after both of them are awaken.[5] $\square$

---

[5]There is no need to wait with time measurement until other neighbors are awaken, as typically assumed in the literature; however, that classic measurement also holds in our case, because once a neighbor is added to set N_v^*, it says there and could be considered as already added after the round when all neighbors of v become awake.