Optimal Backup Strategies Against Cyber Attacks

Optimal Backup Strategies Against Cyber Attacks

Achiya Bar-On[1], Itai Dinur[2], Orr Dunkelman[3], Rani Hod[1], Nathan Keller[1], Eyal Ronen[4], and Adi Shamir[4]

[1] Department of Mathematics, Bar-Ilan University, Israel
{abo1000,rani.hod}@gmail.com, nkeller@math.biu.ac.il
[2] Computer Science Department, Ben-Gurion University, Israel
dinuri@cs.bgu.ac.il
[3] Computer Science Department, University of Haifa, Israel
orrd@cs.haifa.ac.il
[4] Computer Science Department, The Weizmann Institute, Rehovot, Israel
{eyal.ronen,adi.shamir}@weizmann.ac.il

Abstract. In this paper we introduce the new problem of finding the best way to protect a computer system against cyber and ransomware attacks by choosing an optimal backup scheme using k storage devices. While in standard backup schemes it is beneficial to backup as frequently as possible, in the case of sophisticated cyber attacks any attempt to connect a backup device to an already infected computer is likely to stealthily corrupt its data and thus make it unusable when the actual attack happens. Our formalization of the problem casts it as a special case of an online/offline optimization problem, in which the defender tries to minimize the maximal extra cost caused by his lack of knowledge about the time of the initial infection.
Any backup scheme can be viewed as a very simple pebbling game where in each step any one of the k backup pebbles can be moved to any point to the right of all the pebbles along the time axis, and the goal of the game is to keep the pebbles as evenly spread as possible at all times. However, its optimal solution is surprisingly complicated and leads to interesting combinatorial questions which are reminiscent of questions in discrepancy theory. For small values of k, we find provably optimal backup strategies for all $k < 10$, and each case seems to be somewhat different: For $k = 3$ the best schedule uses backup times which form a simple geometric progression based on the golden ratio, but already for $k = 4$ we show that no geometric progression can be optimal and the efficiency of the best online scheme is worse than the efficiency of the best offline scheme by a strange factor of $2/(1 + \cos(2\pi/7))$. For $k = 8$ the optimal order of device updates becomes highly complicated: $1, 2, 4, 7, 5, 3, 1, 7, 5, 3, 7, 1, 4, 2, 4, 5 \ldots$, while for $k = 9$ it is much simpler. We then consider the case of arbitrarily large values of k, and prove a matching upper and lower bound of $\ln 4$ on the asymptotic efficiency of optimal backup schemes when k goes to infinity.

1 Introduction and Notation

A crucial element in the maintenance of any computer system is to keep up-to-date backups of all the files stored in the system. Standard backup schemes are designed to provide fast recovery in case there is a natural disaster (disk crash, fire, earthquake, etc.), and there is vast literature about how to do it. However, most of these publications consists of either descriptions of particular commercial systems (such as [2,4,9]) or general advice by government organizations such as NIST (e.g., [11,12]), whose bottom line is that one should continuously update in parallel several backup disks which are located at different physical locations. In this paper we show that the design of backup strategies against malicious adversaries requires a completely different approach, which had not been previously considered by the academic research community. The main difference is that sophisticated cyber attackers usually try to destroy (or corrupt in a way which is not immediately detectable) all the backup copies of data files made over many months or even years before launching the actual attack (victims in such cases usually go out of business, as happened in [5]). Similarly, perpetrators of advanced ransomware attacks typically wait several weeks before displaying their financial demands, in order to have a chance to encrypt all the data files on any external storage device which may be intermittently connected to the PC to backup its data (see [7]). Note that in such cases it is dangerous to keep a backup system permanently connected to the main system, it does not help to update multiple backup devices in parallel, and one should not cycle too quickly through all the available backup devices (e.g., by using seven disk-on-keys and connecting a different one to the PC each day of the week).

In this paper, we introduce the first mathematical formalization of this important problem which makes it amenable to rigorous combinatorial analysis. We assume that the system administrator has k storage devices. His *backup scheme* consists of an infinite sequence of update actions (d_n, t_n), where in the n-th action he updates device d_n at time t_n for a monotonically increasing and unbounded sequence of update times. For the sake of simplicity, we assume that at time zero the file system was empty, and that each update action instantaneously replaces the previous contents of the device with the full current state of the file system. In addition, we assume that backup devices never fail, unless they are connected to an already infected system (which corrupts all their data). At any time, we define the *snapshot* of the scheme as the sequence of times $S = (T_1, T_2, \ldots, T_k)$ at which each device was last updated, and add a superscript n when we want to refer to the snapshot after update action number n. Since we can always rename the backup devices, we simplify our notation by referring to the device that currently contains the i-th oldest data as device number i, and thus the k numbers in the snapshot are always sorted. Note that in this notation, updating the sequence of devices $1, 1, 1, \ldots$ always updates the device which currently holds the oldest data (and thus we cyclically go through all the physical storage devices in a round-robin way), whereas updating the sequence of devices $k, k, k, \ldots$ always updates the device which holds the newest data (and thus only a single physical storage device gets repeatedly updated).

When a natural disaster such as a disk crash strikes, it happens at a random time T, and its existence becomes immediately known. The system administrator solves the problem by using the latest available backup, and the recovery cost is typically proportional to $T - T_k$ (which corresponds to how out-of-date it is). Our model is different since it is based on two different time points: a secret time T'

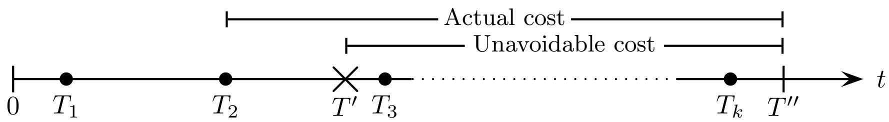

Fig. 1. Unavoidable cost vs. actual cost.

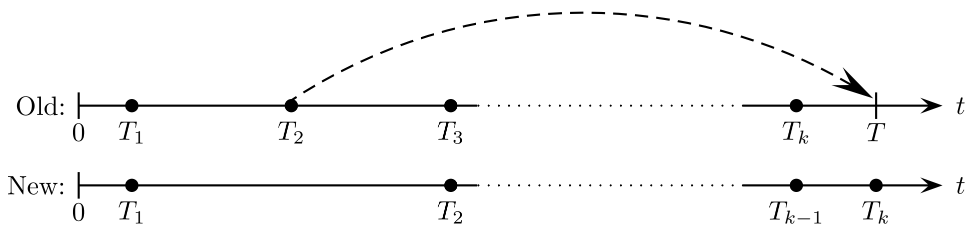

Fig. 2. Transition from old to new snapshot for the update action $(2, T)$.

in which the initial infection happens, and a public time T'' in which the files on the main system are destroyed. We assume that after the attack happens at T'', a forensic investigation will make T' known to the defender, and thus he will know which backup device contains the freshest reliable data: this will be the largest T_i which is smaller than T' in the current snapshot at time T'' (if no such backup is available, the defender can always go back to the empty file system at time zero). The total cost of the recovery will then be proportional to $T'' - T_i$. However, it is important to note that a cost of $T'' - T'$ is unavoidable even when the defender uses the best conceivable backup scheme in which he makes a full backup just before T'. The additional cost which can be attributed to his lack of knowledge about T' is thus proportional only to $T' - T_i$ (see Fig. 1). This casts the problem as a special type of an online/offline optimization problem, in which we want to analyze the maximum extra cost (in an additive rather than multiplicative sense) that can be inflicted on an online defender who does not know the infection time T' compared to a hypothetical offline defender who knows it and can schedule his backups accordingly. It is easy to see that for any possible attack time T'', the maximum value of $T' - T_i$ will be achieved when the infection time T' happens just before the end of the longest interval (T_i, T_{i+1}) in the current snapshot S at time T'', since this would make the backup created at time T_{i+1} unusable, and will force an online defender to use the maximally out-of-date backup created at time T_i as the best available data. The best strategy of the defender is thus to make this longest interval as short as possible at all times.

The main difficulty for the defender is that as time progresses, he cannot nudge each backup device from its old time to a slightly later time in order to keep them evenly spread out — he can only replace an old backup by the current state of the file system. The effect that an update action (d, t) has on the current snapshot $T_1, T_2, \dots, T_k$ is described in Figure 2: The update eliminates the d-th oldest backup point from the snapshot, and combines the two intervals (T_{d-1}, T_d) and (T_d, T_{d+1}) into a single interval (T_{d-1}, T_{d+1}) whose length is the sum of the two previous lengths. In addition, it renames all the later devices in the snapshot by subtracting 1 from their indices. Finally, it adds a new interval which extends from the last previous update time T_k to the new update time T. This is a particularly simple pebbling game, in which we repeatedly push backup pebbles to the right along the one dimensional time axis. It is reminiscent of the way we create new Fibonacci numbers by adding together the last two numbers in the sequence, but in our case we have k interval sizes in the current snapshot and can choose which pair of consecutive numbers we would like to replace by their sum (while adding at the end an arbitrary new number to keep the number of numbers fixed at k). The goal of the defender is to use his allowed actions in order to keep all the backup times as evenly spread as possible between zero and T as T goes to infinity. The shortest possible length of the longest interval in a snapshot is T/k (which is created by placing the k backups in an arithmetic progression of times ending at T), and we say that interval i is *c-compliant* if $T_i - T_{i-1} \leq cT/k$ (where we assume that $T_0 = 0$ for simplicity). A backup scheme is *c-efficient* if all the interval lengths at all times T are bounded by cT/k. Note that making c as small as possible requires careful compromises since if the current snapshot is too uniform (e.g., when all its intervals have exactly the same length), then the next snapshot will necessarily be far from uniform by having some interval which is twice as long as the other intervals. A better strategy will thus be to have some variety in the interval lengths, so that it will always be possible to combine two relatively short consecutive intervals whose total length will not be too long. For any k, we denote by c_k the smallest possible value of c among all the possible backup schemes which use k storage devices. The smallest possible value of c_k is clearly at least 1, and can be easily shown to be at most 2 by using the simple strategy of starting from one arithmetic progression whose step is δ, and changing it into another arithmetic progression whose step is 2δ by updating every second backup device. However, finding the best update strategy for each k is an interesting combinatorial problem whose solution is surprisingly complicated and which may be of independent interest.

Our paper is thus a combination of elements from many different disciplines: The original motivation of the problem comes from cryptography, its formalization can be viewed as an online/offline optimization problem,[1] its methodology can be viewed as a pebbling game,[2] and its goal is reminiscent of discrepancy theory.[3]

The paper is structured as follows: In Section 2 we describe some basic properties of c-efficient schemes, and use them in Section 3 to find provably optimal backup schemes for all values of k smaller than 10. In Section 4 we describe a recursive

[1] Originally defined by Karp in [6] and extensively studied in [3].

[2] See [10] for a comprehensive survey of applications of pebbling games in complexity theory and [1] for another interesting application in cryptography.

[3] This theory tries to characterize the unavoidable deviations from a uniform distribution, as described in [8].

construction of backup schemes whose efficiencies c_k converge to $\ln 4 \approx 1.3863$ as k goes to infinity, and in Section 5 we prove the optimality of our construction by proving a matching lower bound on the asymptotic efficiencies of all the possible backup schemes. Section 6 concludes the paper and poses some open problems.

2 Basic Observations

The following two observations tell us where exactly we should put our focus when designing an efficient backup scheme. The first one deals with compliance of interval $k+1$:

Property 1. If two consecutive update actions in a k-backup scheme occur at times t_n and t_{n+1} and $t_{n+1} > t_n / (1 - c/k)$ then, for any snapshot taken between time $t_n \cdot k / (k-c)$ and t_{n+1}, interval $k+1$ cannot be c-compliant.

Proof. Let $S = (T_1, \ldots, T_{k-1}, T_k)$ be the snapshot at such a time T. We have $T_k = t_n$ while $T > t_n / (1 - c/k)$. Thus $T - T_k > cT/k$, i.e., interval $k+1$ is not c-compliant. ∎

This observation tells us that the sequence $(t_n)_{n=1}^{\infty}$ of update times in a c-efficient scheme must satisfy $t_{n+1} \leq t_n / (1 - c/k)$ for all $n \geq 1$, giving rise to the following definition.

Definition 1. *Fix $q > 1$. A k-backup scheme is called q-subgeometric if the sequence $(t_n)_{n=1}^{\infty}$ of update times satisfies $t_{n+1} \leq t_n \cdot q$ for all $n \geq 1$. In case of equality the scheme is called q-geometric.*

Remark 1. We may scale the update times sequence by any positive constant, as compliance and thus efficiency are homogenous conditions.

The second observation tells us what is necessary to render a scheme efficient, besides being subgeometric.

Property 2. To check if a $1/(1 - c/k)$-subgeometric k-backup scheme is c-efficient, it suffices to verify c-compliance only for intervals in the initial snapshot and intervals in standard snapshots — those taken immediately after an update action. Moreover, it suffices to only check the compliance of interval d in a standard snapshot taken immediately after updating device d.

Proof. Let $S = (T_1, \ldots, T_{k-1}, T_k)$ be a snapshot taken at time T and let (d, T_k) be the most recent update action. By subgeometry interval $k+1$ is c-compliant. For $j = 1, \ldots, k$, the snapshot did not change between time T_k (immediately after the update) to time T, so if interval j was c-compliant at time T_k it remains c-compliant at time $T \geq T_k$ as well. Furthermore, all intervals in S were present in the previous snapshot S' besides d and k, and the latter is c-compliant since it has the same length as interval $k+1$ in S' when considered just before the update. ∎

Henceforth we only deal with the initial snapshot S_0 and the standard snapshots S_n taken at time t_n for $n \geq 1$. In particular, there is no need to consider interval $k+1$.

Next we make two observations about updated devices, which are useful in both upper and lower bounds.

Property 3. Without loss of generality, a k-backup scheme never updates the newest device, i.e., $\max\,(d_n)_{n=1}^{\infty} \le k-1$.

Proof. Fix a c-efficient scheme and consider a snapshot $S_n = (T_1, \ldots, T_{k-1}, T_k)$ followed by update action (k, T'_k), so $S_{n+1} = (T_1, \ldots, T_{k-1}, T'_k)$. If interval k is c-compliant in S_{n+1} then for any $T_k \le T \le T'_k$ we have $T_{k-1} \ge T'_k\,(1-c/k) \ge T\,(1-c/k)$, i.e., $T - T_{k-1} \le cT/k$. Hence we can simply skip the update action (k, T'_k) altogether and the scheme remains c-efficient. ∎

Property 4. Without loss of generality, a k-backup scheme updates the oldest device infinitely often, i.e., $\liminf\limits_{n\to\infty} d_n = 1$.

Proof. Fix a c-efficient scheme and consider its standard snapshots $S_n = (T_1^n, \ldots, T_k^n)$ for $n \ge 1$. For notational convenience let $\Delta_j^n = T_j^n - T_{j-1}^n$ for $n \ge 1$ and $j = 1, \ldots, k$. The update times sequence $(t_n)_{n=1}^{\infty}$ is unbounded and $t_n = T_k^n = \sum_{j=1}^k \Delta_j^n$, so there must exist some minimal $J < k$ for which the sequence $(\Delta_J^n)_{n=1}^{\infty}$ is unbounded. If $J = 1$ we are done; otherwise we show how to modify the scheme, while maintaining c-efficiency, such that $\left(\Delta_{J-1}^n\right)_{n=1}^{\infty}$ is unbounded as well.

Let $M = \sup_n \Delta_{J-1}^n$ and pick N such that $\Delta_J^N > M$. No further update action (d_n, t_n) can update a device $d_n < J$ since that would result in $\Delta_{J-1}^n > M$. In particular, we have $\Delta_J^n > M$ for all $n \ge M$. Pick $n > N$ such device J is updated at time t_n and $\Delta_{J+1}^{n-1} > M$. It is possible since $\left(\Delta_{J+1}^n\right)_{n=M}^{\infty}$ is unbounded and cannot decrease unless device J is updated. We modify the scheme to update device $J-1$ instead of J at time t_n. Instead of creating a c-compliant interval of length

$$\Delta_J^n = \Delta_J^{n-1} + \Delta_{J+1}^{n-1} > \Delta_J^{n-1} + M,$$

this update action now creates an interval of length

$$\Delta_{J-1}^n = \Delta_{J-1}^{n-1} + \Delta_J^{n-1} \le \Delta_J^{n-1} + M,$$

which is still c-compliant. The scheme remains c-efficient since future updates only touch intervals $j \ge J$ and Δ_J^n did not grow. This process can be repeated as long as $\sup_n \Delta_{J-1}^n$ remains finite. Note that in the limiting scheme device $J-1$ is updated infinitely often so $\sup_n \Delta_{J-1}^n$ cannot be finite; thus $\left(\Delta_{J-1}^n\right)_{n=1}^{\infty}$ must be unbounded. ∎

Remark 2. An important consequence of Property 4 is that we can essentially ignore compliance of the initial snapshot by *rebasing*, i.e., running the scheme until all backups present in the initial snapshot are updated at least once and treating the then-current snapshot as the new starting snapshot S_0. Note that the new S_0 consists only of members of the original update times sequence $(t_n)_{n=1}^{\infty}$.

Recall that the sequence of update times in an efficient scheme is subgeometric; the next observation tells us that we can assume it actually grows exponentially fast.

Property 5. Without loss of generality, the sequence $(t_n)_{n=1}^{\infty}$ of update times in a c-efficient k-backup scheme satisfies $t_{n+2} > t_n \cdot q$, where $q = 1/(1 - c/k)$.

Proof. Starting with a c-efficient scheme, we consider device updates in their natural order and modify the scheme, while maintaining efficiency, such that the property holds. At each step, the only modifications we make are to skip or delay an update, which ensures that $(t_n)_{n=1}^{\infty}$ is still unbounded. The basic simple idea in all modifications is that if an update that merges two intervals is possible at some time T, i.e., the newly created interval is c-compliant, then an update that merges the same intervals is also possible at any time $T' > T$.

If the property does not hold, consider the smallest n for which $t_{n+2} \leq t_n \cdot q$. We show we can either skip one of the update at times t_{n+1}, t_{n+2} or delay the update at time t_{n+2} until time $t_{n+1} \cdot q > t_n \cdot q$. For $i = 1, 2$, denote by x_i the next time at which the device at t_{n+i} is updated, and by ℓ_i the label of the actual device updated.[4] Note that $\ell_1 \neq \ell_2$ by Property 3, and that the order of x_1 and x_2 is undetermined.

If $x_1 < x_2$ (the device ℓ_2 at t_{n+2} is in place by the time the device ℓ_1 at t_{n+1} is updated), then eliminate the device update at t_{n+1} (keeping the sequence q-subgeometric as $t_{n+2} \leq t_n \cdot q$) and switch roles between the two devices throughout the rest of the scheme. Namely, update ℓ_2 from its previous update (before being updated at t_{n+2}) directly at time x_1,[5] and update ℓ_1 from its previous update directly at time t_{n+2} (and then again at x_2 and so forth).

Otherwise, $x_2 < x_1$; consider the time period $(t_{n+1}, t_{n+1} \cdot q]$ and denote by ℓ_3 the device that is removed from it at the latest time before x_1 (note that there is at least one such device, namely ℓ_2, hence it may be that $\ell_3 = \ell_2$). Delay the update of ℓ_3 to time $t_{n+1} \cdot q$ and skip all other updates in $(t_{n+1}, t_{n+1} \cdot q]$ that are removed from this time period before x_1. In other words, for each device (except ℓ_3) updated in the time period $(t_{n+1}, t_{n+1} \cdot q]$ and updated again in the time period $(t_{n+1} \cdot q, x_1)$, update it directly from its previous update (before t_{n+1}) to its next one (after $t_{n+1} \cdot q$). ■

An immediate corollary of Property 5 is that $t_{n+i} > t_n \cdot q^{\lfloor i/2 \rfloor}$ for $i \geq 0$; in particular, $t_{n+i} > t_n \cdot q^2$ for $i \geq 4$. Our last observation says when we can get $t_{n+3} > t_n \cdot q^2$.

Property 6. Let $S = (T_1, \ldots, T_k)$ be a snapshot of a c-efficient k-backup scheme such that $T_j = t_n$ and $T_{j+1} = t_{n+3}$ for some $j = 1, \ldots, k-1$. Thus without loss of generality $t_{n+3} > t_n \cdot q^2$, where $q = 1/(1 - c/k)$.

Proof. Starting with a c-efficient scheme that satisfies Property 5, we modify it such that Property 6 holds as well. As in the previous proof, for $i = 1, 2$, denote by ℓ_i the

[4] In contrast to a temporary index of the device in the device sequence at some snapshot S, a label of a device is fixed.

[5] If device ℓ_2 was firstly used at t_{n+2}, then simply use ℓ_2 first at x_1 in the modified scheme.

label of the device at t_{n+i} and denote by x_i the time by which it is removed from the time period (t_n, t_{n+3}).

We show that if $t_{n+3} \leq t_n \cdot q^2$ then we can delay the update at time t_{n+1} until $t_n \cdot q$ and eliminate the update at time t_{n+2}. Note that $t_{n+1} \leq t_n \cdot q < t_{n+2}$ by Properties 1 and 5, hence after this change Property 5 continues to hold.

If $x_1 < x_2$ we switch the roles of the two devices by updating ℓ_2 directly at x_1 and ℓ_1 at $t_n \cdot q$ and again at x_2; otherwise, $x_2 < x_1$ and we simply delay the update of ℓ_1 to time $t_n \cdot q$ and skip the update at t_{n+2} by updating ℓ_2 directly at x_2 from its previous update. ■

3 Optimal Backup Schemes for Small Values of k

3.1 Round-robin and $k = 2, 3$

We now analyze the efficiency of the round-robin (RR) scheme, which always updates device 1 (the oldest). Besides serving as a first example, round-robin is optimal for $k \leq 3$ and will make an appearance within the asymptotically optimal construction of Section 4.

Proposition 1. *Round-robin is c-efficient if and only if $c \geq kr$, where r is the smallest root of $r = (1-r)^{k-1}$.*

Proof. Denote by $(t_n)_{n=1}^{\infty}$ the sequence of update times. Every standard snapshot in RR after the first $k-1$ takes the form $S_{n+k} = (t_{n+1}, t_{n+2}, \ldots, t_{n+k})$. If the scheme is $1/(1-c/k)$-subgeometric then $t_{n+k} \leq t_{n+1}(1-c/k)^{1-k}$; on the other hand, interval 1 is c-compliant when $t_{n+1} \leq ct_{n+k}/k$. By Property 2 the whole scheme is c-efficient if c satisfies $(1-c/k)^{k-1} \leq c/k$, thus $c \geq kr$ and the minimal c is obtained by taking a $1/(1-r)$-geometric update times sequence.

Note also that a q-geometric update times sequence yields an RR variant whose efficiency is $k \cdot \max\left\{1 - 1/q, q^{1-k}\right\}$ for any $q > 1$; indeed this is minimized by $q = 1/(1-r)$. ■

Remark 3. Round-robin is pretty bad for large k; indeed, its asymptotic efficiency $kr \approx \ln k - \ln \ln k$ is inferior to the simple bound $c_k \leq 2$ from the introduction.

The case $k = 2$ is made obvious by Property 3, since without loss of generality RR is the only scheme to consider. Thus $c_2 = 1$.

Proposition 2. *For $k = 3$ we have $c_3 = 3r_3 \approx 1.1459$, where $r_3 = \frac{3-\sqrt{5}}{2} \approx 0.38197$ is the smaller root of $x^2 - 3x + 1 = 0$.*

Proof. For the upper bound, RR is $3r_3$-efficient. Note that $1/(1-r_3) = \frac{1+\sqrt{5}}{2}$ is the golden ratio. For the lower bound, consider a $3r$-efficient 3-backup scheme and let $S_n = (x, y, z)$ be a snapshot. By subgeometry $z \leq y/(1-r) \leq x/(1-r)^2$ and by compliance of interval 1, $x \leq rz$. Together we have $(1-r)^2 \leq r$, i.e., $r^2 - 3r + 1 \leq 0$. Thus $r \geq r_3$. ■

3.2 Periodic schemes and $k = 4, 5$

Round-robin is no longer optimal for $k > 3$; nevertheless, the constructions we shortly present for $k = 4, 5$ still have a periodic structure, which is captured by the following definitions.

Definition 2. *A device sequence $(d_n)_{n=1}^{\infty}$ is called m-periodic for an integer $m \geq 1$ if $d_{n+m} = d_n$ for all $n \geq 1$. We denote it by writing the m-tuple $D = (d_1, \ldots, d_m)$.*

Typically we consider $m \geq 2$ as RR is the only 1-periodic device sequence satisfying Property 4.

Definition 3. *An update times sequence $(t_n)_{n=1}^{\infty}$ is called (q, m)-periodic for $q > 1$ and an integer $m \geq 1$ if $t_{n+m} = q^m \cdot t_n$ for all $n \geq 1$. We denote it by writing the m-tuple $P = (t_1, \ldots, t_m)$.*

Note that for $m = 1$, saying that an update times sequence is $(q, 1)$-periodic is equivalent to saying it is q-geometric; for $m \geq 2$ it is possible that a (q, m)-periodic sequence is not q-subgeometric, but only q'-subgeometric for some $q' > q$.

Definition 4. *A k-backup scheme is called (q, m)-periodic if it consists of an m-periodic device sequence D, a (q, m)-periodic update times sequence P, and its standard snapshot S_m after one full period is a scaling of its initial snapshot S_0 by a factor of q^m.*

To fully describe an (q, m)-periodic k-backup scheme, we need to provide q, D, P and the initial snapshot S_0. For convenience we sometimes normalize the last entry of S_0 to 1.

Remark 4. In fact, all constructions presented in this paper are periodic schemes; they are optimal for small k or asymptotically.

We continue with $k = 5$. Note that when applying Property 2 to a periodic scheme, it suffices to verify compliance within a single period.

Proposition 3. *For $k = 5$ we have $c_5 = 5r_5 \approx 1.2256$, where $r_5 \approx 0.24512$ is the (only) real root of $x^3 - 4x^2 + 5x - 1 = 0$.*

Proof. For the upper bound, set $q = 1/(1 - r_5)$ and consider the $(q, 2)$-periodic scheme with $S_0 = \left(1, q^2, q^3, q^4, q^5\right)$, $D = (3, 1)$, and $P = \left(q^6, q^7\right)$. The scheme is indeed $(q, 2)$-periodic, as $S_1 = \left(1, q^2, q^4, q^5, q^6\right)$ and $S_2 = \left(q^2, q^4, q^5, q^6, q^7\right) = q^2 S_0$. It is q-geometric so we just need to verify compliance of interval 3 in S_1: $q^4 - q^2 \leq r_5 q^6$ and interval 1 in S_2: $q^2 \leq r_5 q^7$. Altogether it remains to show

$$r_5 \geq \max\left\{q^{-5}, q^{-2} - q^{-4}\right\} = \max\left\{(1 - r_5)^5, (1 - r_5)^2 - (1 - r_5)^4\right\},$$

both of which are equal to r_5.

For the lower bound, consider a $5r$-efficient 5-backup scheme. It cannot be RR, which satisfies $(1 - r)^4 \leq r$ and in particular $r > 0.275 > r_5$. We thus pick a snapshot $S_n = (x, y, z, u, v)$ followed by the update steps $(1, w)$ and (d, t) for some $d \in \{2, 3, 4\}$. Thus S_{n+2} is one of (y, u, v, w, t), (y, z, v, w, t), or (y, z, u, w, t). In either case, compliance of S_n, S_{n+1} and S_{n+2} implies $v \leq r\,(v + w + t)$, which together with $v \leq (1 - r)\,w \leq (1 - r)^2\,t$ yields $\left((1 - r)^2 - r\right)(1 - r) \leq r$, i.e., $r^3 - 4r^2 + 5r - 1 \geq 0$. Thus $r \geq r_5$. ■

For $k = 2, 3, 5$ the upper bound was a geometric scheme, but for $k = 4$ the optimal sequence is a periodic, yet non-geometric scheme.

Proposition 4. *For $k = 4$ we have $c_4 = 4r_4 \approx 1.2319$, where $r_4 = (2 + 2\cos(2\pi/7))^{-1} \approx 0.307979$ is the smallest root of $x^3 - 5x^2 + 6x - 1 = 0$. Moreover, any geometric 4-backup c-efficient scheme must have $c \geq 4\tilde{r}_4 \approx 1.2707$, where $\tilde{r}_4 \approx 0.31767 > r_4$ is the real root of $x^3 - 3x^2 + 4x - 1 = 0$.*

Proof. For the upper bound, set $q = \sqrt{\alpha}$ where $\alpha = 1/\sqrt{r_4}$ is the largest root of $x^3 - x^2 - 2x + 1 = 0$ and consider the $(q, 2)$-periodic scheme with $S_0 = \left(1, \alpha, \alpha^3 - \alpha^2, \alpha^2\right)$, $D = (3, 1)$, and $P = \left(\alpha^4 - \alpha^3, \alpha^3\right)$. The scheme is indeed $(q, 2)$-periodic, as $S_1 = \left(1, \alpha, \alpha^2, \alpha^4 - \alpha^3\right)$ and $S_2 = \left(\alpha, \alpha^2, \alpha^4 - \alpha^3, \alpha^3\right) = \alpha S_0 = q^2 S_0$. The update times sequence P is $1/(1 - r_4)$-subgeometric if $(1 - r_4)\,\alpha^3 \leq \alpha^4 - \alpha^3$ and $(1 - r_4)\left(\alpha^3 - \alpha^2\right) \leq \alpha^2$; furthermore we need to verify compliance of interval 3 in S_1: $\alpha^2 - \alpha \leq r_4\left(\alpha^4 - \alpha^3\right)$ and interval 1 in S_2: $\alpha \leq r_4\alpha^3$. Altogether it remains to show

$$r_4 \geq \max\left\{2 - \alpha, 1 - \frac{1}{\alpha^2 - \alpha}, \frac{\alpha^2 - \alpha}{\alpha^4 - \alpha^3}, \alpha^{-2}\right\}.$$

Indeed the last three are equal to r_4 while the first is

$$2 - \alpha = \alpha^{-1}\left(1 - (\alpha - 1)^2\right) < \alpha^{-1}\left(1 - (\alpha - 1)\,\alpha^{-1}\right) = \alpha^{-2} = r_4,$$

using $\alpha - 1 > \alpha^{-1}$ since $r_4 + \sqrt{r_4} < r_3 + \sqrt{r_3} = 1$.

For the lower bound, consider a $4r$-efficient 4-backup scheme. If it is RR, it satisfies $(1 - r)^3 \leq r$ so $r \geq \tilde{r}_4$. Otherwise pick a snapshot $S_n = (x, y, z, u)$ followed by update steps $(1, v)$ and (d, w) for some $d \in \{2, 3\}$. The snapshot S_{n+2} is (y, u, v, w) or (y, z, v, w). The snapshot S_{n+3} after the next update step (d', t) can be one of four:

- If $S_{n+3} = (z, v, w, t)$ then $v = (v - z) + z \leq r\,(w + t)$ by compliance of $S_{n+2} = (y, z, v, w)$ and S_{n+3}, yielding $(1 - r)^3 \leq r^2$, i.e., $r > 0.43016 > \tilde{r}_4 > r_4$;
- If S_{n+3} is (y, v, w, t) or (u, v, w, t) then $v = y + (v - y) \leq r\,(v + t)$ by compliance of S_{n+1} and S_{n+3} or $u \leq rt$ by compliance of S_{n+3}; either way yields $(1 - r)^3 \leq r$, which again implies $r \geq \tilde{r}_4$;
- The only remaining option is $S_{n+3} = (y, u, w, t)$, which means $S_{n+2} = (y, u, v, w)$. Now

$$(1 - r)\,v \leq u = y + (u - y) \leq r\,(v + w)$$
$$w = y + (u - y) + (w - u) \leq r\,(v + w + t)$$

by compliance of all four snapshots, so

$$(1-2r)\,v \le rw$$
$$(1-r)^2\,w \le (1-r)\,r\,(v+t) \le (1-r)\,rv + rw$$

hence $(1-2r)\left((1-r)^2 - r\right) \le r^2\,(1-r)$, i.e., $r^3 - 6r^2 + 5r - 1 \ge 0$, which implies $r \ge r_4$.

Note that a $1/(1-r)$-geometric update times sequence would satisfy $(1-r)\,w = v$ in the last case, yielding $(1-r)^3 \le r$ one last time and implying $r \ge \tilde{r}_4$. ■

3.3 Casting the problem as a linear program

Fix $c \ge 1$ and a device sequence $(d_n)_{n=1}^{\infty}$. Can we choose an initial snapshot $S_0 = \left(T_1^0, \ldots, T_k^0\right)$ and a sequence $(t_n)_{n=1}^{\infty}$ of update times such that the resulting k-backup scheme is c-efficient?

Any standard snapshot S_n consists of a particular subset of the variables $\left\{T_j^0\right\}_{j=1}^{k}$ and $\{t_n\}_{n=1}^{\infty}$, and using $(d_n)_{n=1}^{\infty}$ we can determine exactly which. Furthermore, monotonicity and $1/(1-c/k)$-subgeometry of update times, and c-compliance of the snapshots are all expressed as linear inequalities. This gives rise to an infinite linear program $L = L\left(c; (d_n)_{n=1}^{\infty}\right)$, which is feasible whenever a c-efficient scheme with the prescribed device sequence exists. Note that all constraints are homogenous, so to avoid the zero solution we add a non-homogenous condition, e.g., $T_k^0 = 1$.

In addition, we are not interested in solutions where $(t_n)_{n=1}^{\infty}$ is bounded. This can happen, for instance, when $d_n = k-1$ for all $n \ge 1$.[6] Luckily, using Property 5 we can restrict our attention to backup schemes with exponentially increasing update times, so we add to L the linear inequalities therein. Now L is feasible if and only if a c-efficient scheme with the prescribed device sequence exists; in other words, c_k is the infimum over $c \ge 1$ for which there exists a device sequence $(d_n)_{n=1}^{\infty}$ such that $L\left(c; (d_n)_{n=1}^{\infty}\right)$ is feasible.

As an infinite program, L is not too convenient to work with. We can thus limit our attention to subprograms L_N for some finite N, which only involve the variables $\left\{T_j^0\right\}_{j=1}^{k}$ and $\{t_n\}_{n=1}^{N}$ and the relevant constraints. By itself, L_N can no longer ensure the existence of a c-efficient scheme, but can be used to prove lower bounds on c_k in the following way. Write $\Sigma = \{1, \ldots, k-1\}$.

Definition 5. *A finite sequence $D \in \Sigma^*$ is called a c-witness if $L_{|D|}(c; D)$ is infeasible.*

Definition 6. *A set $\mathcal{D} \subset \Sigma^*$ of finite sequences is called blocking if any infinite sequence $(d_n)_{n=1}^{\infty}$ over Σ has some prefix in $\mathcal{D}$.*

[6] This may not seem a valid scheme to consider, given Property 4; however, we might have to consider an arbitrarily long prefix of the device sequence without device 1 when solving a finite subprogram.

k	$\tilde{c}$	$\tilde{q}$	$(1-\tilde{c}/k)^{-k/2}$	D	m	Geometric?	Optimal?
2	1	2	2	(1)	1	Yes	Yes, RR
3	1.145898	1.618093	2.058171	(1)	1	Yes	Yes, $\lvert\mathcal{D}\rvert = 1$
4	1.231914	1.342363	2.088146	$(1,3)$	2	No	Yes, $\lvert\mathcal{D}\rvert = 3$
5	1.225612	1.324718	2.019801	$(1,3)$	2	Yes	Yes, $\lvert\mathcal{D}\rvert = 5$
6	1.296634	1.239553	2.076001	$(1,2,3,1,3,5)$	6	No	Yes, $\lvert\mathcal{D}\rvert = 601$
7	1.310296	1.208296	2.06552	$(1,3,4,1,5,3)$	6	No	Yes, $\lvert\mathcal{D}\rvert = 3005$
8	1.320138	1.159761	2.057263	$(1,2,4,7,5,3,1,7,5,3,7,1,4,2,4,5)$	16	No	Yes, $\lvert\mathcal{D}\rvert = 51691$
9	1.325768	1.15984	2.048492	$(1,5,3,5,1,5,6,3)$	8	No	Yes, $\lvert\mathcal{D}\rvert = 911662$
10	1.334405	1.132085	2.046483	$(1,5,3,5,1,5,6,3,1,5,9,3,5,9)$	14	No	?
11	1.342994	1.123932	2.046568	$(1,3,5,6,1,6,2,10,6,3,6,1,6,2,6,3,9,6)$	18	No	?
12	1.354008	1.121687	2.051024	$(1,2,3,5,6,7,1,2,6,3,6,7,1,2,6,3,6,9,7)$	19	No	?
13	1.355001	1.114038	2.045151	$(1,3,6,7,4,7,1,7,8,3)$	10	No	?
14	1.360472	1.097269	2.045409	$\begin{pmatrix} 1,4,2,6,7,4,7,8,1,8,2,3,7,12,4, \\ 7,8,1,4,7,2,7,8,4,13,8,1,8,4,2, \\ 7,4,7,8,1,8,4,2,7,12,4,7,13,8 \end{pmatrix}$	44	No	?

Table 1. Computationally-verified upper bounds on c_k for $2 \le k \le 14$.

Proposition 5. *If there exists a blocking set of c-witnesses for some $c \ge 1$, then $c_k > c$.*

We now describe a strategy to approximate c_k to arbitrary precision. The lower bound of Proposition 5 does not assume any periodicity of the backup scheme. For the upper bound, however, we limit our focus to periodic constructions. For a finite sequence $D \in \Sigma^*$ of length m, we can augment $L_m(c; D)$ with the periodicity equality constraint $S_m = q^m \cdot S_0$; call the resulting program $L^*(c, q; D)$. This is a finite linear program, which we can computationally solve given c, q, and D. Although $q \le 1/(1 - c/k)$ is not known to us, we can first compute an approximation $\tilde{q}$ of q by solving $L_{10m}\left(c; D^{10}\right)$, and then solve $L^*(c, \tilde{q}; D)$. Using binary search, we can compute a numerical approximation $\tilde{c}$ of the minimal c for which $L^*(c, \tilde{q}; D)$ is feasible. Lastly, we can enumerate short sequences $D \in \Sigma^*$ in a BFS/DFS-esque manner and take the best $\tilde{c}$ obtained.

The results in Table 1 were obtained by a Python program that follows this strategy. The 'Optimal' column specifies whether a blocking set $\mathcal{D}$ of $(c - \epsilon)$-witnesses was obtained for $\epsilon = 10^{-5}$. The '$(1 - \tilde{c}/k)^{-k/2}$' column shows how tight is the lower bound of Corollary 2.

At first it seems that Proposition 5 cannot be used to pinpoint c_k exactly, since any finite blocking set $\mathcal{D}$ of $(c_k - \epsilon)$-witnesses for some $\epsilon > 0$ leaves an interval of uncertainty of length ϵ. The following proposition eliminates this uncertainly.

Proposition 6. *For every finite sequence $D \in \Sigma^*$ there is finite set $C_D \subset \mathbb{R}$ such that the feasibility of $L_{|D|}(c; D)$, for some $c \ge 1$, only depends on the relative order between c and members of C_D. In particular, there exists some $\epsilon > 0$ such that if D is a $(c - \epsilon)$-witness but not a c-witness then D is also a c'-witness for all $c - \epsilon < c' < c$.*

Proof. Fix $D \in \Sigma^*$. Treating c as a parameter, note that the subprogram $L_{|D|}(c; D)$ is feasible if and only if the polytope $\mathcal{P}_D(c)$ defined by $L_{|D|}(c; D)$ is nonempty.

Decreasing c shrinks $\mathcal{P}_D(c)$ until some critical c_D for which $\mathcal{P}_D(c_D)$ is reduced to a single vertex, at which a subset of the linear constraints are satisfied with equality. Hence c_D is a solution of some polynomial equation determined by the relevant constraints. The set of constraints is finite, thus there are finitely many polynomial equations that can define c_D, and we can take C_D as the set of all their roots. Now take ϵ to be smaller than the distance between any two distinct elements of C_D. ■

Remark 5. Note that when ϵ is small enough, we can actually retrieve the polynomial equations defining c and q from the polytope $\mathcal{P}_D(\tilde{c}, \tilde{q})$; using this method we get an algebraic representation of c_k rather than a rational approximation.

4 Asymptotically Optimal Upper Bounds

In this section we describe a family of periodic geometric k-backup schemes. In contrast to our experience from Table 1 — that optimal/best known schemes are geometric only for $k = 2, 3, 5$ — this family is rich enough to include asymptotically optimal, i.e., $(1 + o(1))\, c_k$-efficient, schemes.

4.1 A recursive geometric scheme

Fix a real number $q > 1$ and an integer $t \geq 0$. We describe a k-backup scheme $\mathcal{B}(q, K)$, where K is a $(t+2)$-subset $\{0, \ldots, k\}$ whose elements are

$$k = k_0 > k_1 > k_2 > \cdots > k_t > k_{t+1} = 0.$$

$\mathcal{B}(q, K)$ is $\left(q, 2^t\right)$-periodic and q-geometric, and its device update sequence $D = (d_n)_{n=1}^{2^t}$ is defined as $d_n = 1 + k_{\mu(n)+1}$, where $\mu(n)$ is the largest $\mu \leq t$ for which 2^μ divides n. As per Remark 2, we can rely on rebasing and there is no need to define the initial snapshot S_0 explicitly.

Example 1. For $K = \{0, 3, 5, 9, 19\}$ we get $D = (10, 6, 10, 4, 10, 6, 10, 1)$.

Although $\mathcal{B}(q, K)$ was defined explicitly above, it can be viewed also as a recursive scheme: the base case $t = 0$ is the round-robin k-backup scheme $\mathcal{B}(q, \{0, k\})$; for $t \geq 1$, $\mathcal{B}(q, K)$ alternates between updating the $(k_1 + 1)$-st oldest backup and between acting according to the inner k_1-backup scheme $\mathcal{B}\left(q^2, K \setminus \{k_1\}\right)$.

Let us elaborate a bit more on the recursive step. In every snapshot $S_n = (T_1, \ldots T_k)$ we have $T_j = q^{n+j}$ for $k_1 + 1 \leq j \leq k$ since we never update backups younger than $k_1 + 1$. In every *odd* snapshot S_n we have just updated the $(k_1 + 1)$-st oldest backup, so $T_{k_1} = q^{n+k_1-1}$ while $T_{k_1+1} = q^{n+k_1+1}$. This means that $\log_q T_j$ for $j = 1, \ldots, k_1$ all have the same parity as $n + k_1 - 1$ in any snapshot S_n. We thus treat $S' = (T_1, \ldots, T_{k_1})$ as a snapshot of a k_1-backup scheme, which operates at half speed and never sees half of the backups. The inner backup scheme can rightfully be called $\mathcal{B}\left(q^2, K \setminus \{k_1\}\right)$, as the common ratio of the update times sequence for the backups that do make it to the inner scheme is q^2, and taking only the even locations of D yields a 2^{t-1} periodic sequence $(d'_n)_{n=0}^{\infty}$ such that $d'_n = d_{2n} = 1 + k_{\mu(2n)+1} = 1 + k_{\mu(n)+2}$.

4.2 Analyzing the recursive scheme

First we determine exactly how efficient $\mathcal{B}(q,K)$ can be for any q and K, and then we work with a particular choice.

Denote by $r(q,K)$ the maximum of

$$1-q^{-1}; \tag{1a}$$

$$\max\left\{q^{-e(\ell)}\left(q^{2^\ell}-q^{-2^\ell}\right)\right\}_{\ell=0}^{t-1}; \text{ and} \tag{1b}$$

$$q^{2^t-e(t)}, \tag{1c}$$

where

$$e(\ell)=\sum_{i=0}^{\ell}2^i\left(k_i-k_{i+1}\right)\quad \text{for } \ell=0,1,\ldots,t.$$

Theorem 1. *Given q and K, the minimal c for which $\mathcal{B}(q,K)$ is c-efficient is $c=r(q,K)\cdot k$.*

Proof. We proceed by induction on t. The base case $t=0$ is the round-robin scheme, which we already know to be $k\cdot\max\left(1-q^{-1},q^{1-k}\right)$-efficient.

For $t\geq 1$, fix $\tilde{r}$ such that $\mathcal{B}(q,K^*)$ is $\tilde{r}k$-efficient and consider a critical interval, i.e., an interval of length $\delta=\tilde{r}\cdot T_k$ in some snapshot $S_n=(T_1,\ldots,T_k)$. This interval is either between T_{k-1} and T_k, and then we get (1a), or an interval created by merging two smaller intervals. We consider even and odd snapshots separately.

When n is odd, this interval is between $T_{k_1}=q^{n+k_1-1}$ and $T_{k_1+1}=q^{n+k_1+1}$, so

$$\left(q^1-q^{-1}\right)q^{n+k_1}=\tilde{r}\cdot T_k=r\cdot q^{n+k};$$

this gives us case $\ell=0$ of (1b).

When n is even, this interval is among $S'=(T_1,\ldots,T_{k_1})$ so $\delta\geq r'\cdot T_{k_1}$, where $r'=r\left(k_1,K\setminus\{k_1\}\right)$ and $T_{k_1}=q^{n+k_1}$. By induction r' is the maximum of

$$1-\left(q^2\right)^{-1}; \tag{2}$$

$$\max\left\{\left(q^2\right)^{-e'(\ell)}\left(\left(q^2\right)^{2^\ell}-\left(q^2\right)^{-2^\ell}\right)\right\}_{\ell=0}^{t-2}; \text{ and} \tag{3}$$

$$\left(q^2\right)^{2^{t-1}-e'(t-1)}, \tag{4}$$

where

$$e'(\ell)=\sum_{i=0}^{\ell}2^i\left(k_{i+1}-k_{i+2}\right)=\frac{1}{2}\left(e(\ell+1)-k_0+k_1\right).$$

Substituting $\delta=\tilde{r}\cdot T_k=r\cdot q^{n+k}$ we get that

$$\tilde{r}\geq q^{k_1-k}\cdot\left(1-q^{-2}\right)=q^{-e(0)}\left(1-q^{-2}\right),$$

which is a relaxation of case $\ell = 0$ of 1b;

$$\tilde{r} \geq q^{k_1 - k} \cdot q^{-2e'(\ell)} \left(q^{2^{\ell+1}} - q^{-2^{\ell+1}}\right) = q^{-e(\ell+1)} \left(q^{2^{\ell+1}} - q^{-2^{\ell+1}}\right) \text{ for } \ell = 0, \ldots, t-2,$$

which covers the remaining cases of (1b); and

$$\tilde{r} \geq q^{k_1 - k} \cdot q^{2^t - 2e'(t-1)} = q^{2^t - e(t)},$$

which is simply (1c). Altogether we showed that $\tilde{r} \geq r(q, K^*)$.

Repeating the argument when starting with a critical interval of the inner scheme shows that $\mathcal{B}(q, K^*)$ is indeed $r(q, K^*) \cdot k$-efficient. ■

Given an integer $k \geq 2$, let $t = \lfloor \log_2 k \rfloor - 1$. Define $K^* = \{k_0, \ldots, k_{t+1}\}$ by $k_i = \lfloor 2^{-i} k \rfloor$ for $i = 0, \ldots, t$ and $k_{t+1} = 0$. Note that $k_0 = k$ and that $k_t \in \{2, 3\}$.

Theorem 2. *For an appropriate choice of q, $\mathcal{B}(q, K^*)$ is $(1 + o(1)) \ln 4$-efficient.*

Proof. Write $x = \log_2 k$ and $\epsilon = 3/x$. Let $\gamma = (1 + \epsilon) \frac{\ln 4}{k}$ and let $q = e^\gamma$. Observe that $q^{k/2} = 2^{1+\epsilon}$ for our choice of q. We use Theorem 1 to show that $\mathcal{B}(q, K^*)$ is $k\gamma$-efficient when k is large enough. Clearly (1a) holds since

$$1 - q^{-1} = 1 - e^{-\gamma} < \gamma;$$

It remains to verify (1b) and (1c), handled by Propositions 7 and 8 respectively. ■

Proposition 7. *For $k \geq 2^{13}$ and γ, ϵ, x, q as above, $q^{-e(\ell)} \left(q^{2^\ell} - q^{-2^\ell}\right) < \gamma$ for all $\ell = 0, \ldots, t-1$.*

Proposition 8. *For $k \geq 5$ and γ, ϵ, x, q as above, $q^{2^t - e(t)} < \gamma$.*

Before proving Propositions 8 and 7 we would like to simplify $e(\ell)$ for our K^*.

Claim. For $\ell = 0, \ldots, t$ we have $e(\ell) > (\ell + 1) k/2 - 2^\ell$; moreover, $e(t) > (t + 2) k/2 - 2^t$.

Proof. We have

$$\begin{aligned} e(\ell) &= \sum_{i=0}^{\ell} 2^i (k_i - k_{i+1}) = k_0 - 2^\ell k_{\ell+1} + \sum_{i=1}^{\ell} 2^{i-1} k_i \\ &= k - 2^\ell \left\lfloor 2^{-\ell-1} k \right\rfloor + \sum_{i=1}^{\ell} 2^{i-1} \left\lfloor 2^{-i} k \right\rfloor \\ &\geq k - \frac{k}{2} + \frac{1}{2} \sum_{i=1}^{\ell} \left(k - 2^{i-1}\right) = \frac{k}{2} (\ell + 1) + 1 - 2^\ell. \end{aligned}$$

The slightly improved bound for $\ell = t$ is obtained by observing that $2^t k_{t+1} = 0$ in the above calculation since $k_{t+1} = 0$. ■

Proof (of Proposition 8). By the claim above

$$e(t)-2^t>(t+2)\,k/2-2^{t+1}\geq xk/2-k$$

so

$$q^{2^t-e(t)}\leq q^{k-xk/2}=\frac{4^{1+\epsilon}}{8k}\leq\frac{\ln\left(4^{1+\epsilon}\right)}{k}=\gamma,$$

where the last inequality is true since $y\leq 8\ln y$ for $2\leq y\leq 26$ and indeed for $k\geq 5$ we have $0<\epsilon<1.3$ and $4<4^{1+\epsilon}<25$. ■

Proof (of Proposition 7). By the claim above

$$\begin{aligned}
q^{-e(\ell)}\left(q^{2^\ell}-q^{-2^\ell}\right)&<q^{2^\ell-(\ell+1)k/2}\left(q^{2^\ell}-q^{-2^\ell}\right)\\
&=2^{-(1+\epsilon)(\ell+1)}\left(q^{2^{\ell+1}}-1\right)\\
&=2^{-(\ell+1)}\cdot 8^{-(\ell+1)/x}\left(e^{2^{\ell+1}\gamma}-1\right)\\
&<8^{-(\ell+1)/x}\frac{\gamma}{1-2^{\ell+1}\gamma},
\end{aligned}$$

using $e^x-1<x/(1-x)$ for all $0<x\leq 1$. Thus, it remains to show that

$$8^{-(\ell+1)/x}\leq 1-2^{\ell+1}\gamma=1-2^{\ell+1-x}(1+\epsilon)\ln 4.$$

Let $z=x-(\ell+1)$ and note that $1\leq z\leq x-1$ as $0\leq\ell\leq t-1$. Define

$$f(x,z)=1-2^{1-z}\left(1+\frac{3}{x}\right)\ln 2-8^{z/x-1};$$

to conclude the proof we show that $f(x,z)$ is positive for all $1\leq z\leq x-1$ and $x\geq 13$.

First we compute some partial derivatives of f.

$$\begin{aligned}
\frac{df(x,1)}{dx}&=\frac{3\ln 2}{x^2}\left(1+8^{1/x-1}\right);\\
\frac{df(x,x-1)}{dx}&=\frac{3\ln 2}{x^2}\left(2^{2-x}-8^{-1/x}\right)+2^{2-x}\left(1+\frac{3}{x}\right)\ln^2 2;\\
\frac{\partial f(x,z)}{\partial z}&=\left(1+\frac{3}{x}\right)2^{1-z}\ln^2 2-\frac{3\ln 2}{x}8^{z/x-1};\\
\frac{\partial^2 f(x,z)}{(\partial z)^2}&=-\left(1+\frac{3}{x}\right)2^{1-z}\ln^3 2-\left(\frac{3\ln 2}{x}\right)^2 8^{z/x-1}.
\end{aligned}$$

Now $\frac{df(x,1)}{dx}>0$ everywhere, so $f(x,1)$ is increasing and $f(x,1)\geq f(13,1)>0$ for all $x\geq 13$. Next $\frac{df(x,x-1)}{dx}<0$ for $x>7$, so$f(x,x-1)$ is decreasing and

$$f(x,x-1)>\lim_{x\to\infty}f(x,x-1)=\lim_{x\to\infty}1-8^{-1/x}=0$$

k	t	efficiency of $\mathcal{B}(q^*, K^*)$	q^* is smallest root of	$(q^*)^{k/2}$	effective ϵx as % of τ
2	0	1	$x-2$	2	< 0
4	1	1.527864045	x^2-x-1	2.61803399	38.63%
8	2	1.446619893	x^4-x-1	2.220744085	24.69%
16	3	1.414522345	x^8-x-1	2.096981559	15.40%
32	4	1.399982156	$x^{16}-x-1$	2.045751025	9.337%
64	5	1.393037798	$x^{32}-x-1$	2.022250526	5.520%
128	6	1.389641669	$x^{64}-x-1$	2.010975735	3.196%
256	7	1.389039657	$x^{959}-\left(x^{128}-1\right)/(x-1)$	2.006538067	2.996%
512	8	1.38976776	$x^{2175}-\left(x^{256}-1\right)/(x-1)$	2.005370202	4.265%
1024	9	1.389961428	$x^{4863}-\left(x^{512}-1\right)/(x-1)$	2.004616597	5.003%
2048	10	1.389901672	$x^{10751}-\left(x^{1024}-1\right)/(x-1)$	2.004083324	5.413%
4096	11	1.3897339	$x^{23551}-\left(x^{2048}-1\right)/(x-1)$	2.003678733	5.631%
8192	12	1.389529892	$x^{51199}-\left(x^{4096}-1\right)/(x-1)$	2.003356204	5.738%
16384	13	1.389323191	$x^{110591}-\left(x^{8192}-1\right)/(x-1)$	2.003090123	5.785%
32768	14	1.389128152	$x^{237567}-\left(x^{k/2}-1\right)/(x-1)$	2.002865287	5.799%
65536	15	1.388949844	$x^{507903}-\left(x^{k/2}-1\right)/(x-1)$	2.002671985	5.796%
131072	16	1.388789052	$x^{1081343}-\left(x^{k/2}-1\right)/(x-1)$	2.002503614	5.786%
262144	17	1.388644741	$x^{2293759}-\left(x^{k/2}-1\right)/(x-1)$	2.002355444	5.772%

q^* for $t \geq 7$ is smallest root $\neq 1$ of $x^{kt/2+k/4} - x^{kt/2+k/4-1} - x^{k/2} + 1$.

Table 2. Performance of the recursive k-backup scheme for $k = 2^{t+1}$.

for all $x > 7$. Lastly, $f(x,z)$ is concave in z as $\frac{\partial^2 f}{(\partial z)^2} < 0$ everywhere. Thus

$$\min_{1 \leq z \leq x-1} f(x,z) = \min\{f(x,1), f(x,x-1)\} > 0$$

for all $x \geq 13$. ■

Remark 6. Theorem 2 chooses q suboptimally. Empirical evidence shows that, for all $k \geq 2$, the optimal $q = q^*$ for $\mathcal{B}(q, K^*)$ satisfies (1a) and one of (1b) and (1c). In other words, it is the smallest root of either $1 - x^{-1} = x^{2^t - e(t)}$ or $1 - x^{-1} = x^{-e(\ell)}\left(x^{2^\ell} - x^{-2^\ell}\right)$ for some $\ell = 0, \ldots, t-1$.

Remark 7. With additional effort the error term in Theorem 2 can be improved by a factor of almost 6 to $\epsilon = \tau/x$, where $\tau = -\log_2 \ln 2 \approx 0.53$. The major obstacle is that cases $\ell = t-2$ and $\ell = t-1$ of (1b) need to be done separately since the appropriate $f(x,z)$ in Proposition 7 is negative for $z < \log_2(\ln 4/(1-\ln 2)) \approx 2.1756$. Choosing $\epsilon = \tau'/x$ for $\tau' < \tau$ violates (1c) for large enough $k = 2^{t+2} - 1$.

Remark 8. We verified that $\mathcal{B}(q^*, K^*)$ is $(1 + \tau/x)\ln 4$-efficient for $2 \leq k \leq 2^{13}$ as well. See Tables 2 and 3 for the cases $k = 2^{t+1}$ and $k = 2^{t+2} - 1$, respectively.

5 Asymptotically Optimal Lower Bounds

In this section we prove lower bounds on c_k, focusing on asymptotic lower bounds in which k grows to infinity.

k	t	efficiency of $\mathcal{B}(q^*, K^*)$	q^* is smallest root of	$(q^*)^{k/2}$	effective ϵx as % of τ
3	0	1.145898034	$x^2 - x - 1$	2.058171027	< 0
7	1	1.318433761	$x^8 - x^7 - 1$	2.075892596	< 0
15	2	1.408092224	$x^{24} - x^{23} - 1$	2.09450451	11.62%
31	3	1.448810165	$x^{64} - x^{63} - 1$	2.099878619	42.25%
63	4	1.46399549	$x^{160} - x^{159} - 1$	2.097270918	63.36%
127	5	1.466865403	$x^{384} - x^{383} - 1$	2.091122952	76.82%
255	6	1.464278319	$x^{896} - x^{895} - 1$	2.083917032	85.05%
511	7	1.459602376	$x^{2048} - x^{2047} - 1$	2.076835761	89.98%
1023	8	1.454408143	$x^{4608} - x^{4607} - 1$	2.070357915	92.91%
2047	9	1.449377613	$x^{10240} - x^{10239} - 1$	2.064618545	94.66%
4095	10	1.444769023	$x^{22528} - x^{22527} - 1$	2.059600382	95.72%
8191	11	1.440647199	$x^{49152} - x^{49151} - 1$	2.055228333	96.39%
16383	12	1.436994729	$x^{106496} - x^{106495} - 1$	2.051413108	96.83%
32767	13	1.43376402	$x^{229376} - x^{229375} - 1$	2.048069607	97.14%
65535	14	1.430900622	$x^{491520} - x^{491519} - 1$	2.045123383	97.36%
131071	15	1.428352881	$x^{1048576} - x^{1048575} - 1$	2.042511814	97.54%
262143	16	1.426075306	$x^{2228224} - x^{2228223} - 1$	2.040183169	97.69%

q^* is smallest root of $x^{(t+1)(k+1)/2} - x^{(t+1)(k+1)/2-1} - 1$.

Table 3. Performance of the recursive k-backup scheme for $k = 2^{t+2} - 1$.

We start by proving a simple asymptotic lower bound $c_k \geq 2 - \ln 2 - o(1)$, and then improve it to $c_k \geq (1 - o(1)) \ln 4$, which is asymptotically optimal via the matching upper bound of Section 4.

5.1 Stability and bounding expressions

Obtaining lower bounds requires viewing the problem from a different perspective. It will sometimes be more convenient to refer to a certain physical device, without considering its temporary index in the device sequence at some snapshot S (which is variable and depends on S).

Given a k-backup scheme, we define a function $BE(s)$ and use it to bound its efficiency from below. The parameter s is related to the notion of stability, which we now define.

Definition 7. *Fix a k-backup scheme. A device updated at time T is called s-stable, for some $s = 1, \ldots, k-1$, if at least s previous devices are updated before the next time it is updated.*

By Property 4 we can assume all devices get updated eventually; this means that in a snapshot $S = (T_1, \ldots, T_k)$, where T_k is a time by which all devices have been updated from the initial snapshot, we have that the device updated at time T_{k-s} is s-stable for $s = 1, \ldots, k-1$.

For convenience, the proofs in this section assume the update times sequence is normalized by a constant. This is captured by the following definition.

Definition 8. *A k-backup scheme is called s-normalized if an s-stable device is updated at time $R_0 = 1$.*

Given an s-normalized k-backup scheme, we define a sequence of times $1 = R_0 < R_1 < \cdots < R_s$ as follows: R_i for $i \geq 1$ is the time at which the i-th device is *removed* from $(0, 1]$. In other words, R_1 is the time $T > R_0$ at which some device is updated; R_2 is the time $T > R_1$ at which we update the next device that was previously updated in $(0, 1]$ (but not at $T > 1$), and so forth. Note that the device updated at time R_0 is not updated at any time R_i for $i = 1, \ldots, s$ by the definition of stability. Now we are ready to define $BE(s)$.

Definition 9. *The T-truncated bounding expression of an s-normalized k-backup scheme is $BE_T(s) = \sum_{i=1}^{s} U_i$, where $U_i = \min\{T, R_i\}$.*

The bounding expression plays a crucial role in proving lower bounds, based on Proposition 9 below. We note that the truncated bounding expression only depends on the scheme's behavior until time T, and hence the bounds that can be obtained from it are not tight for $k > 3$. Nevertheless, the lower bound we obtain using BE_2 in Corollary 2 is asymptotically optimal, since the gap between it and the upper bound Theorem 2 tends to zero as k grows to infinity.

Remark 9. It is possible to analyze BE_T beyond $T = 2$ and obtain tight lower bounds for larger values of k. However, there is no asymptotic improvement and the analysis becomes increasingly more technical as k grows.

5.2 Asymptotic lower bound of $2 - \ln 2 \approx 1.3068$

To simplify the analysis, we assume k is even. It can be extended to cover odd values of k as well, but this gives no asymptotic improvement since $c_{k+1} \leq c_k \cdot \frac{k+1}{k}$ for all k, so we only lose an error term of $O(1/k)$, which is of the same order as the error terms in Corollaries 1 and 2.

To simplify our notation we write $b = c/k$ throughout this section.

Proposition 9. *Any $(k/2)$-normalized bk-efficient k-backup scheme satisfies $BE_2(k/2) \geq 1/b$.*

Proof. At time $R_0 = 1$, the time period $(0, 1]$ contains k intervals of length $\leq b$, giving rise to the inequality $b \cdot k \geq 1$. At time R_1, a device is removed from $(0, 1]$ and it now contains one interval of length $\leq b \cdot R_1$ (two previous intervals, each of length $\leq b$, were merged), and $k - 2$ intervals of length $\leq b$, giving $b \cdot (k - 2 + R_1) \geq 1$.

At time R_2, an additional device is removed from the time period $(0, 1]$, hence it must contain an interval of length $\leq b \cdot R_2$ formed by merging two previous intervals. We obtain $b \cdot (k - 4 + R_1 + R_2) \geq 1$, since the remaining $k - 3$ intervals must include $k - 4$ intervals of length $\leq b$ and one (additional) interval of length at most $\leq b \cdot R_1$. Note that this claim holds regardless of which device is updated at R_2, and it holds in particular in case one of the intervals merged at time R_2 contains the intervals merged at R_1 (in fact, this case gives the stronger inequality $b \cdot (k - 3 + R_2) \geq 1$).

In general, for $j = 1, 2, \ldots, k/2$, at time R_j the time period $(0, 1]$ must contain j distinct intervals of lengths $\leq b \cdot R_i$ for $i = 1, 2, \ldots, j$, and $k - 2j$ intervals of length

$\leq b$. This gives the inequality

$$k - 2j + \sum_{i=1}^{j} R_i \geq 1/b.$$

Let $j \leq k/2$ be the largest index such that $R_j \leq 2$, so $U_i = R_i$ for all $1 \leq i \leq j$ and $U_i = 2$ for all $j < i \leq k/2$. Now at time R_j we have

$$1/b \leq k - 2j + \sum_{i=1}^{j} R_i = (k/2 - j) \cdot 2 + \sum_{i=1}^{j} U_i = \sum_{i=j+1}^{k/2} U_i + \sum_{i=1}^{j} U_i = BE_2(k/2).$$

■

Now we need an upper bound on the bounding expression. For the simpler lower bound of $2 - \ln 2$ we use the following proposition.

Proposition 10. *Any $(k/2)$-normalized bk-efficient k-backup scheme has $R_i \leq 1/(1-bi)$ for $i = 1, \ldots, k/2$.*

Proof. At time $T = 1/(1-bi)$, all intervals are of length at most $bT = b/(1-bi)$. Since $T - R_0 = 1/(1-bi) - 1 = bi/(1-bi)$, for any $\epsilon > 0$ the time period $(R_0, T+\epsilon]$ must consist of at least $i+1$ intervals, implying that the i-th device was removed from the time period $(0, 1]$ by time T. ■

Proposition 11. *Let $b < \frac{1}{2}$. Any $(k/2)$-normalized bk-efficient k-backup scheme satisfies*

$$b \cdot BE_2(k/2) \leq \ln 2 + \frac{b}{1-2b} + bk - 1.$$

Proof. For $i \leq \lfloor 1/2b \rfloor$, we have $R_i \leq 1/(1 - b\lfloor 1/2b \rfloor) \leq 2$, but we cannot assure that $R_i \leq 2$ for $i > \lfloor 1/2b \rfloor$. By Proposition 10,

$$b \cdot BE_2(k/2) \leq b \cdot \left(\sum_{i=1}^{\lfloor 1/2b \rfloor} R_i + \sum_{i=\lfloor 1/2b \rfloor + 1}^{k/2} 2 \right) \leq \sum_{i=1}^{\lfloor 1/2b \rfloor} \frac{b}{1-bi} + bk - 1.$$

Now for $b < \frac{1}{2}$ the sum on the right-hand side can be bounded by

$$\sum_{i=1}^{\lfloor 1/2b \rfloor} \frac{1}{1/b - i} \leq \int_0^{1/2b} \frac{dx}{1/b - x - 1} = \ln\left(\frac{1}{b} - 1\right) - \ln\left(\frac{1}{2b} - 1\right)$$
$$= \ln 2 + \ln\left(1 + \frac{b}{1-2b}\right) \leq \ln 2 + \frac{b}{1-2b},$$

establishing the proposition. ■

Corollary 1. *For all even $k \geq 4$ we have $c_k \geq 2 - \ln 2 - o(1)$.*

Proof. Fix a $(k/2)$-normalized c_k-efficient k-backup scheme, and write $b = c_k/k < \frac{1}{2}$. By Propositions 9 and 11 we have $c_k = bk \geq 2 - \ln 2 - 1/\left(k/c_k - 2\right) \geq 2 - \ln 2 - 1/\left(k-2\right)$.

■

5.3 Improved asymptotic lower bound of $\ln 4 \approx 1.3863$

We now improve the asymptotic lower bound to $\ln 4$. This result is a simple corollary of the following lemma, which gives a tighter upper bound on the bounding expression.

Lemma 1. *For any s-normalized bk-efficient k-backup scheme such that $1 \leq s \leq k/2$ and $(1-b)^{-k/2} \leq 2$ we have $b \cdot BE_2(s) \leq (1-b)^{-s} - 1$.*

Proof. The proof is by induction on s. For the base case $s = 1$, one device is updated at time at most $1/(1-b)$, giving $b \cdot BE_2(1) \leq b/(1-b) = \frac{1}{1-b} - 1$.

Assume the hypothesis holds for all $i \leq s-1$ and our goal is to prove it for $i = s > 1$. Without loss of generality we assume the scheme satisfies all properties of Section 2. Consider a snapshot at time $T = 2$, and denote by R' the update time of the first device in the time period $(R_0, T] = (1, 2]$ at the snapshot time $T = 2$. We would like to apply the induction hypothesis from time R', but this cannot be done directly since it is not guaranteed that the device last updated at R' in the snapshot at time $T = 2$ is $(s-1)$-stable (potentially, less than s devices are removed from $(0, 1]$ at $T = 2$). To overcome this problem, recall that the truncated bounding expression BE_2 only considers the scheme up to time $T = 2$ by setting $U_i = \min\{2, R_i\}$. Consequently, we can analyze a slightly different scheme with the same bounding expression $BE_2(s)$ in which the device at R' in time $T = 2$ is $(s-1)$-stable.[7] The modification is simple: if the original scheme removes $s' \geq s$ devices from $(0, 1]$ in $(1, 2]$, no change is required; otherwise, $s' < s$ and the modified scheme would simply remove $s - s'$ additional arbitrary devices from $(0, 1)$ at time $T = 2$. This transformation leaves $BE_2(s)$ unchanged, and we can analyze it instead. Note that the modified scheme maintains all properties of Section 2 at times $T < 2$.

We first consider the case in which there is no device update in the time period $(R_0 = 1, R')$, implying that $R' = R_1 \leq 1/(1-b)$. We can now apply the induction hypothesis from $T = R_1$ with $i = s-1$ since at least $s-1$ devices are removed from $(0, R_1)$ before the device at R_1 is updated again, namely, the device at R_1 is $(s-1)$-stable (we have an $(s-1)$-normalized backup scheme). Therefore

$$\begin{aligned} b \cdot BE_2(s) &\leq b \cdot BE_2(s-1) \cdot R_1 + b \cdot R_1 = (b \cdot BE_2(s-1) + b) \cdot R_1 \\ &\leq \left(\left((1-b)^{1-s} - 1\right) + b\right) \cdot \frac{1}{1-b} = (1-b)^{-s} - 1. \end{aligned}$$

[7] There are other ways to solve the problem and apply the induction hypothesis, e.g., by extending the definition of a stable device. However, this seems to require slightly more complex definitions and induction hypothesis.

Note that the multiplication of $BE_2(s-1)$ with R_1 undoes the normalization of the bounding expression at time $T = R_1$, and the addition with $b \cdot R_1$ is because $BE_2(s)$ should account for R_1, but $BE_2(s-1)$ should not.

We also note that this actually proves a (slightly) stronger result, since when calculating $BE_2(s)$ from $T = 1$, we do not add terms larger than $T = 2$, but when calculating $BE_2(s-1)$ from $T = R_1$, the restriction is looser, namely, not to add terms larger than $T = 2 \cdot R_1 > 2$. Therefore, if $BE_2(s-1)$ actually contains terms in the time period $(2, 2 \cdot R_1]$, then $BE_2(s)$ is strictly smaller than $(1-b)^{-s} - 1$.

We are left to prove the hypothesis for $i = s$ given that there is at least one device update in the time period $(R_0 = 1, R')$. Since there in no device in the time period $(R_0 = 1, R')$ in the snapshot at $T = 2$, then $R' \leq R_0 + 2b = 1 + 2b$. Therefore, $R'/R_0 \leq 1 + 2b < 1/(1-b)^2$ and by Property 6 there is exactly one update in $(R_0 = 1, R')$. Therefore, the update in $(R_0 = 1, R')$ occurred at time R_1, and we denote by ℓ the label of the actual device involved. Furthermore, we have $R' = R_2$.[8]

Denote by x the time (after R_1) of the next update of ℓ ($R_2 < x \leq 2$). After time x, all devices were removed from $(R_0 = 1, R_2)$, hence $R_2 \leq 1 + b \cdot x$. As in the previous case, we apply the induction hypothesis from $T = R' = R_2$ with $i = s - 1$ since we are assured that at least $s - 1$ devices are removed from $(0, R_2)$ before R_2 is updated (including ℓ), hence the device updated at R_2 is $(s-1)$-stable. We get $b \cdot BE_2(s) \leq b \cdot BE_2(s-1) \cdot R_2 + b\left(R_2 + \frac{1}{1-b} - x\right)$. Note that we add $b\left(\frac{1}{1-b} - x\right)$ to the right hand side (to bound $b \cdot BE_2(s)$) since ℓ is first updated at $R_1 \leq \frac{1}{1-b}$ after $T = 1$, and not at x, which is the time it is first updated after R_2 (as considered in $BE_2(s-1)$). Once again, we will prove a slightly stronger result than required, as $BE_2(s-1)$ calculated from $T = R_2$ may contain terms which are larger than 2.

[8] The only use of truncating the bounding expression at $T = 2$ in the proof is to limit the number of updates in $(R_0 = 1, R')$ to one.

Recalling that $R_2 \leq 1 + bx$, we obtain

$$\begin{aligned}
b \cdot BE_2(s) &\leq b \cdot BE_2(s-1) \cdot R_2 + b\left(R_2 + \frac{1}{1-b} - x\right) \\
&= \left(b \cdot BE_2(s-1) + b\right) R_2 - b\left(x - \frac{1}{1-b}\right) \\
&\leq \left((1-b)^{1-s} - 1 + b\right)(1+bx) - b\left(x - \frac{1}{1-b}\right) \\
&= \left((1-b)^{-s} - 1\right)(1-b)(1+bx) - b\left(\frac{x - bx - b}{1-b}\right) \\
&= \left((1-b)^{-s} - 1\right)\left(1 + b\left(x - bx - b\right)\right) - \frac{b\left(x - bx - b\right)}{1-b} \\
&= (1-b)^{-s} - 1 + b\left(x - bx - b\right)\left((1-b)^{-s} - 1 - \frac{1}{1-b}\right) \\
&= (1-b)^{-s} - 1 + b\left(x - \frac{1}{1-b}\right)\left((1-b)^{1-s} - 2 + b\right).
\end{aligned}$$

In order to show that $b \cdot BE_2(s) \leq (1-b)^{-s} - 1$, it is sufficient to show that

$$b\left(x - \frac{1}{1-b}\right)\left((1-b)^{1-s} - 2 + b\right) \leq 0.$$

Obviously $b > 0$; there are 3 device updates in the time period $[1, R_2]$, so by Property 5, $x > R_2 > \frac{1}{1-b}$; lastly, $s \leq k/2$ so $(1-b)^{-s} \leq (1-b)^{-k/2} \leq 2$ by the lemma's assumption, which gives

$$(1-b)^{1-s} - 2 + b \leq 2\left(1-b\right) - 2 + b = -b < 0.$$

This completes the induction and the proof of the lemma. ∎

Corollary 2. *For all even $k \geq 2$ we have $(1 - c_k/k)^{-k/2} \geq 2$. In particular, $c_k \geq (1 - o(1)) \ln 4$.*

Proof. Write $b = c_k/k$ and assume for the sake of contradiction that $(1-b)^{-k/2} < 2$. By Lemma 1 and Proposition 9 we have $1 \leq b \cdot BE_2(k/2) \leq (1-b)^{-k/2} - 1$ for a $k/2$-normalized c_k-efficient k-scheme, so $(1-b)^{-k/2} \geq 2$, contradicting our assumption. Now

$$\frac{c_k}{k} \geq 1 - 2^{-2/k} = 1 - e^{-\ln 4/k} \geq \frac{\ln 4}{k} - \frac{1}{2}\left(\frac{\ln 4}{k}\right)^2,$$

hence $c_k \geq \left(1 - \left(\ln 2\right)/k\right) \ln 4$. ∎

6 Concluding Remarks and Open Problems

In this paper we rigorously defined the new problem of how to use k backup devices in order to optimally protect a computer system against sophisticated cyber and ransomware attacks. For small values of $k < 10$ we described concrete backup schemes and proved their optimality, and for large values of k we found matching upper and lower bounds on the asymptotic efficiency of such schemes. The most interesting open problems are to consider other formalizations of the problem (e.g., using randomized schemes, average rather than worst case efficiency measures, or a variable number of storage devices), and to find additional provably optimal backup schemes for $k \geq 10$.

References

1. Joël Alwen, Binyi Chen, Chethan Kamath, Vladimir Kolmogorov, Krzysztof Pietrzak, and Stefano Tessaro, *On the Complexity of Scrypt and Proofs of Space in the Parallel Random Oracle Model*, Advances in Cryptology, Proceedings of EUROCRYPT 2016, Lecture Notes in Computer Science 9666, pp. 358–387, Springer, 2016.
2. Barracuda, *Comprehensive Data Protection Trusted by Millions*. Available online at `https://www.barracuda.com/products/yosemiteserverbackup`.
3. Allan Borodin and Ran El-Yaniv, *Online Computation and Competitive Analysis*, Cambridge University Press, 1998.
4. Cat Ellis, *The best free PC backup software 2017*, 2017. Available online at `http://www.techradar.com/news/the-best-free-pc-backup-software`.
5. Jeff Goldman, *Code Spaces Destroyed by Cyber Attack*, June 2014. Available online at `http://www.esecurityplanet.com/network-security/code-spaces-destroyed-by-cyber-attack.html`.
6. Richard M. Karp, *On-Line Algorithms Versus Off-Line Algorithms: How Much it is Worth to Know in Advance*, Technical report TR-92-044, July 1992. Available online at `http://www.icsi.berkeley.edu/pubs/techreports/TR-92-044.pdf`.
7. Maria Korolov, *Will your backups protect you against ransomware?*, May 2016. Available online at `http://www.csoonline.com/article/3075385/backup-recovery/will-your-backups-protect-you-against-ransomware.html`.
8. Raghu Meka, *Combinatorial Discrepancy: A Tutorial*, 2013. Available online at `http://reu.dimacs.rutgers.edu/~vtuma/disc.pdf`.
9. Microsoft, *Backing Up Your Server*. Available online at `https://technet.microsoft.com/en-us/library/cc753528(v=ws.11).aspx`.
10. Jakob Nordstrom, *Pebble Games, Proof Complexity, and Time-Space Trade-offs*, Logical Methods in Computer Science, Vol. 9, No. 3, 2013.
11. Marianne Swanson, Pauline Bowen, Amy Wohl Phillips, Dean Gallup, and David Lynes, *Contingency Planning Guide for Federal Information Systems*, NIST Special Publication 800-34 Rev. 1, May 2010. Available online at `http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-34r1.pdf`.
12. Marianne Swanson and Barbara Guttman, *Generally Accepted Principles and Practices for Securing Information Technology Systems*, NIST Special Publication 800-14, September 1996. Available online at `http://csrc.nist.gov/publications/nistpubs/800-14/800-14.pdf`.