Near-linear time, Leakage-resilient Key Evolution Schemes from Expander Graphs

Adam Smith and Ye Zhang

Pennsylvania State University
{asmith, yxz169}@cse.psu.edu

Abstract. We develop new schemes for deterministically updating a stored cryptographic key that provide security against an internal adversary who can control the update computation and leak bounded amounts of information to the outside world. Our schemes are much more efficient than the previous schemes for this model, due to Dziembowski, Kazana and Wichs (CRYPTO 2011). Specifically, our update operation runs in time quasilinear in the key length, rather than quadratic, while offering a similar level of leakage resilience.
In order to design our scheme, we strengthen the connections between the model of Dziembowski et al. and "pebbling games", showing that random-oracle-based key evolution schemes are secure as long as the graph of the update function's calls to the oracle has appropriate combinatorial properties. This builds on a connection between pebbling and the random oracle model first established by Dwork, Naor and Wee (CRYPTO 2005). Our scheme's efficiency relies on the existence (which we show) of families of "local" bipartite expander graphs of constant degree.

1 Introduction

Side-channel attacks have led the cryptographic community to develop tools for reasoning about the security of computations on partially compromised devices. Recently, Dziembowski, Kazana and Wichs [10] (henceforth "DKW") proposed a model for leakage-resilient updating of a stored secret key that protects against an internal attacker who can continuously leak a bounded amount of information to the outside world and even tamper with the device's internal computations. In this paper, we develop tools for reasoning about computations in the DKW model. These tools allow us to provide a highly efficient key evolution scheme as well as stronger connections to complexity theory, namely to the theory of pebbling games and expander graphs. Our new scheme tolerates a linear amount of leakage and runs in time quasilinear in the key length, improving significantly on the quadratic-time scheme of [10].

Key Evolution Schemes and the DKW Model. Consider a cryptographic key K that is stored on a device about which an attacker may be able to learn information gradually over time. To stop the entire key from being leaked, it is periodically updated via a deterministic key evolution scheme $\mathcal{K}$ to obtain a sequence of keys $K_0, K_1, K_2, \ldots$ where $K_{i+1} = \mathcal{K}(K_i)$. Determinism allows keys to be updated independently on separate devices. The hope is that the update operator $\mathcal{K}$ effectively erases the effect of previously learned information. If we limit only the *amount* of leakage between updates, no deterministic scheme is secure since the attacker may directly ask for bits of a future key (for example, if the key is N bits long, the attacker could ask for a single bit of $K_N = \mathcal{K}^{(n)}(K_0)$ at each of the first N steps, effectively leaking the entire key K_N). To get around this, researchers have studied restricted classes of functions that can be leaked at each step. In other words, we think of a highly constrained "small" adversary *inside* the device who leaks information to an outside "big" adversary. For example, in addition to restricting the length of the leaked information, one might restrict the leakage to operate independently on separate parts of the memory (e.g., the "wire-probe" [14] and "split-state" models [6]), or to operate only on parts of memory that are explicitly touched by a computation [17], or to be from a computationally simple circuit class such as AC_0 [13].

The DKW model [10] assumes instead that the "small" adversary (the corrupted device) is limited in space. They do *not* assume that the computations inside the device are performed "honestly", but they do assume that they "look" right to the outside world, meaning that the correct round key is available when an update occurs. More specifically, they consider a two-part attacker $(\mathcal{A}_s, \mathcal{A}_b)$, where $\mathcal{A}_s$ has access to the initial key K_0. $\mathcal{A}_b$ is unlimited in communication or storage, but $\mathcal{A}_s$ is significantly restricted:

- When the i-th update occurs, $\mathcal{A}_s$ holds the correct key $K_i = \mathcal{K}^{(i)}(K_0)$ in memory.
- $\mathcal{A}_s$ can send up to c bits to $\mathcal{A}_b$ during each "round" (that is, between any two updates). To avoid a trivial attack, c must be less than $|K|$.
- $\mathcal{A}_s$ may use any algorithm that works with total *space* at most s bits. We denote by $s_{\text{extra}} = s - s_{\text{honest}}$ the difference between s and the space s_{honest} used by an honest implementation of $\mathcal{K}$. For the model to make sense, s_{extra} must be nonnegative.
- $\mathcal{A}_s$ and $\mathcal{A}_b$ are limited to reasonable computation. We use the random oracle model (as do [10]); the number of oracle calls made by the adversary is bounded by the parameter q.

The key evolution scheme is secure roughly if the leakage on the first i updates lets $\mathcal{A}_b$ learn nothing about the later keys $K_{i+1}, K_{i+2}, \ldots$. Formalizing this is delicate (see "Our Contributions", below). Intuitively though, any key evolution scheme must somehow prevent $\mathcal{A}_s$ from making a copy of the key (since then it could compute future keys and leak information about them while still keeping the current key around), so s must be less than $|K| + s_{\text{honest}} \approx 2|K|$ for a secure key evolution scheme to be possible. An ideal scheme would allow the honest evaluation algorithm to use time and space as close to $|K|$ as possible, while tolerating $c \approx |K|$ leakage and $s_{\text{extra}} \approx |K|$ extra space.

At first glance, it seems that a simple solution to this problem is to use a sufficiently complicated hash function to update the key at each step. If we use the random oracle model and assume that the oracle maps $\{0,1\}^{|K|}$ to $\{0,1\}^{|K|}$, then the scheme can indeed be proven secure when $s < 2|K| - \log q$ and $c < |K| - \log q$. But such a naïve version of the random oracle abstraction hides too much in this setting: a hash function may be computable from a tiny summary of its input (hash functions based on the Merkle-Damgård paradigm, for example, are insecure in the DKW model; see [10] for details).

Instead, we seek to design schemes that provably prevent an untrusted device from computing future keys while keeping the current key available. We use the random oracle to model a "small" hash function H that maps $\{0,1\}^{dw}$ to $\{0,1\}^{w}$ for a small constant d and fixed word length w. The update function $\mathcal{K}$ operates on longer keys and tolerates leakage far above the length of the hash function's inputs and outputs.

DKW [10] proposed an elegant key evolution scheme that is secure in the random oracle model as long as $4c + s_{\text{extra}}$ is significantly less than $|K|/2$. The result is remarkable in that the key length $|K| = nw$ can be very long even when the hash function output length w is short, yet the leakage and adversarial work space can both be linear in $|K|$. Their update algorithm requires only $s_{\text{honest}} = |K| + w$ bits, but it requires at least $\frac{3}{2}n^2$ hash function calls, which is quadratic in the key length.

1.1 Our Contributions

1. We give a new key evolution scheme, also in the random oracle model, for which the key update step can be done in quasilinear time $O(n \log n \log q)$ for keys of length $|K| = nw$ (assuming hash evaluations take constant time). This improves dramatically over the n^2 running time in DKW. As with DKW, the extra space s_{extra} and leakage c in our scheme can both be linear in $|K|$. Specifically, the update algorithm can be run in space $s_{\text{honest}} = (1+\delta)|K|$ for an arbitrarily small constant $\delta > 0$ and it is secure roughly as long as $4c + s_{\text{extra}} < |K|/8$ (slightly worse than the $|K|/2$ tolerance of DKW).
2. We strengthen the connections between the DKW model and pebbling theory, showing that random-oracle-based key evolution schemes are secure as long as the "graph" of the update function's calls to the oracle has appropriate combinatorial properties. This builds on a connection between pebbling and the random oracle model first established by Dwork, Naor and Wee [9] and built on by DKW [11,10]. Our scheme's efficiency relies on the existence (which we show) of families of δ-*local* bipartite expander graphs of constant degree.

3. We provide a precise formalization in the standard model for the problem described by DKW. Their definition of security was highly specific to a particular class of schemes in the random oracle model. We provide a stand-alone security definition and prove that the simplified security definition in [10] implies our definition.

1.2 Background and Further Related Work

Leakage Resilience. Many different models for leakage-resilience have been considered in the literature, and a survey is beyond the scope of this paper. At a high level, many works have sought to design schemes resilient against "limited" leakage of secret keys. A common model is to bound the number of leaked bits (such as with "relative leakage", e.g. [1] or "bounded retrieval", e.g. [12,3,5]). More general models bound only computational difficulty of guessing the secret given the leakage (e.g. [7,21]). Subsequent work investigated leakage that occurs continually over many time steps (e.g. [8,4,16]). Schemes with a deterministic update are vulnerable to leakage on future keys [12,20]; this leads naturally to the models of restricted leakage mentioned above, including the DKW model.

Pebbling and Random Oracles. Pebbling games were first investigated as a way to prove time/space tradeoffs in circuit complexity [19]. Given a directed acyclic graph G, the *inputs* (or sources) are vertices of in-degree 0, and *outputs* (or sinks) are vertices of out-degree 0. A pebbling strategy begins with special markers ("pebbles") on the input vertices and seeks to cover all the outputs with pebbles, under the restriction that a pebble can only be placed on a vertex when all of its immediate predecessors have been covered.

Pebbling games were first used in cryptography by Dwork, Naor and Wee [9] in the context of proofs of work. They observed a strong connection between pebbling games and the random oracle model: Given a graph G and a hash function H, they design a boolean function whose computational complexity subject to a space constraint (in the RO model) is given by number of moves needed to pebble G subject to a contraint on the number of available pebbles. More specifically, each vertex of the graph is assigned a label of length w (the output size of the hash function). Input vertices are labeled using the function's inputs, and other vertices are labeled by the hash of the labels of their predecessors. The output of the function is the concatenation of the labels of output vertices. This connection between pebbling and the RO model was developed further by DKW to design "one-time" pseudorandom functions [11] and leakage-resilient key evaluation schemes [10]. Assuming the availability of a random oracle, DKW showed that the computations in their model could be made to correspond to a variant of pebbling (see Section 4). We develop new techniques for analyzing such pebbling games in this work.

A class of graphs called *superconcentrators* [18] emerged as important for pebbling lower bounds. "Stacks" of superconcentrators (that is, graphs constructed by placing many superconcentrators in sequence) require exponentially many moves to pebble using a sublinear number of pebbles [15]. We use such a stack in our construction. While we cannot use the results from previous work directly (since we use a variant of standard pebbling games), we modify a key lemma from Lengauer and Tarjan [15] (the "basic lower bound") for our analysis.

A line of research focused on construction superconcentrators with as sparse as possible, eventually obtaining constructions $O(n)$ edges for n inputs and outputs [18,2]. Our constructions use sparse superconcentrators, but we require graphs with several additional properties and so again we cannot use the results from the literature directly.

1.3 Overview of Our Construction and Techniques

The scheme of DKW defined a key update function $\mathcal{K}$ via a very simple graph (using random oracle to get a boolean function as in Dwork et al. [9]). The graph has n inputs and n outputs (and thus the key length is nw). In between, there are $\frac{3}{2}n$ layers, each with n vertices. Each vertex j on a given layer i has edges *to* vertices j and $j+1(\bmod n)$ on layer $i+1$, and edges *from* vertices $j-1(\bmod n)$ and j on layer $i-1$. It is possible to pebble this graph using only $n+2$ pebbles, and hence it is possible evaluate their update function using space $(n+2)w = |K| + 2w$. However, the graph has $\frac{3}{2}n^2$ vertices and thus requires $O(n^2)$ time to evaluate.

A key observation is that one can think of the DKW graph as being "generated" by a much simpler graph, the cycle C_n. Namely, if we identify vertices on two adjacent layers i and $i+1$, we get a graph on n vertices where each vertex j is connected to vertices $j-1$ and $j+1$ (as well as to itself). In a nutshell, our construction generates a key evaluation scheme from a more complex graph. This allows us to prove security using far fewer layers ($O(\log n \log q)$ layers, instead of n).

The graph of our key update scheme consists of a stack of $O(\log n \log q)$ copies of a "base" bipartite graph of constant degree. We need two apparently contradictory properties from the graphs: *locality* and *vertex expansion*.

Locality is the property that, if we order left and right vertices from 1 to n, then edges only exist between vertices that are "nearby" in the ordering (at most δn indices apart, for a small constant δ). If the bipartite graphs are local, then our graph can be pebbled using just $(1+\delta)n$ pebbles, and so the update function can be evaluated in space $s_{\text{honest}} = (1+\delta)nw$.

Vertex expansion is the property that small sets of vertices on the left are connected to "many" vertices on the right. We show that key evolution schemes based on expander graphs are secure even against attackers with time exponential in the height of the stack used to define the update function. The proof has two components: first, we show that stacks of $\log n$ expanders are superconcentrators, and hence that learning anything about future keys requires $\mathcal{A}_{\text{s}}$ to "sacrifice" a large amount of memory that cannot be used to compute the current round key. The second component is to show that $\mathcal{A}_{\text{s}}$ also needs a large amount of memory to be able to eventually compute the current round key. This argument uses expansion more directly (that is, it does not go through superconcentrators), and is much more technically involved. Taken together, the two components show that any successful attack requires $s_{\text{honest}} + \Omega(|K|)$ memory, even when $\Omega(|K|)$ leakage is possible.

Finally, we show that for every $\delta > 0$, one can find constant-degree, *local* vertex expanders, completing the construction.

2 Graph-based Key Evolution Schemes

We work with a specific class of key evolution schemes, following the approach of Dwork et al. [9]. Let $G = (V, E)$ be a graph. The input set $I(G)$ of G is the set of vertices with in-degree of 0. Similarly, the output set $O(G)$ is the set of all vertices with out-degree 0. We denote by $V(G) = V$ the set of vertices of G and by $E(G) = E$ is the set of edges.

Definition 1 (Key Evolution Scheme as a Graph). *Let $H : \{0,1\}^{dw} \to \{0,1\}^w$ be a random oracle. Let $G_{\mathcal{K}}$ be a directed graph with n inputs $I_1, \ldots, I_n$ and n outputs $O_1, \ldots, O_n$ such that each vertex in $G_{\mathcal{K}}$ has indegree either 0 or d. We define a key evolution scheme $\mathcal{K} : \{0,1\}^{nw} \to \{0,1\}^{nw}$ as follows: Let K_i denote the i-th round key and write $K_i = (r_{i1}, r_{i2}, \ldots, r_{in})$ where each r_{ij} has w bits. Assign a w-bit value (called a* label*) $r(v)$ to each vertex v in $G_{\mathcal{K}}$. For inputs $I_1, \ldots, I_n$, set $r(I_j) = r_{ij}$ ($j = 1, \ldots, n$). For $v \notin I(G_{\mathcal{K}})$, let $t_1, \ldots, t_d$ be d vertices connected to v and set $r(v) = H(r(t_1), \ldots, r(t_d))$. Then the output K_{i+1} is defined as $(r(O_1), r(O_2), \ldots, r(O_n))$ (the concatenation of labels of the outputs).*

The key evolution scheme in [10] can be described as a grid graph that has $\frac{3}{2}n$ layers and n vertices in each layer. Specifically, the j-th vertex at the i-th layer $v_{i,j}$ is connected to $v_{i+1,j}$ and $v_{i+1,(j+1) \bmod n}$.

2.1 Security Models

We consider two security models. The first, more general one is given in the standard model, and does not assume anything about the structure of the key evolution scheme or its analysis. Because of space constraints, it is given in Definition 18 (Appendix A). The second model, due to DKW [10], is specific to graph-based key evolution schemes in the random oracle model. We show that the specific definition (Definition 3) implies the general one (Theorem 6, Appendix A).

We denote by *round* the period between two successive updates.

Definition 2 **((c, s, q) adversary).** *An adversary $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_b)$ is a (c, s, q) adversary in the random oracle model if:*

1. *$\mathcal{A}_s$ can store at most s bits at any time,*
2. *$\mathcal{A}_s$ can send at most c bits of communication to $\mathcal{A}_b$ in each round,*
3. *$\mathcal{A}_b$ has unlimited storage space and can send unlimited communication to $\mathcal{A}_s$, and*
4. *$\mathcal{A}_s$ and $\mathcal{A}_b$ call the random oracle at most q times overall.*

Given a vertex v, we say that the adversary *evaluates $r(v)$ via the random oracle* if it calls the oracle H on the correct labels $r(t_1), \dots, r(t_d)$ of v's predecessors.

Definition 3 **((c, s, q, ϵ)-Security [10]).** *Consider a graph-based key evolution with graph G and oracle $H : \{0,1\}^{dw} \to \{0,1\}^w$. Fix a round $u > 0$. Consider the following security game* Game_3 *between a challenger $\mathcal{C}$ and an adversary $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_b)$:*

– *$\mathcal{C}$ picks K_0 uniformly at random in $\{0,1\}^{nw}$ and gives K_0 to $\mathcal{A}_s$.*
– *For $i = 1, ..., u$, $\mathcal{A}_s$ outputs a string $\tilde{K}_i \in \{0,1\}^w$. Round i ends when $\mathcal{A}_s$ begins to output $\tilde{K}_i$.*

The event E occurs if $\tilde{K}_i = K_i$ for all $i = 1, \dots, u$. The event E_1 occurs if, before the end of round u, *either $\mathcal{A}_s$ or $\mathcal{A}_b$ evaluates the label of any vertex $r_{u+1,j}$ (for $j \in \{1, \dots, n\}$) defining the $u+1$-st key K_{u+1}.*

The advantage of $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_b)$ at round u is defined as:

$$Adv_{\mathcal{A}}^{\mathsf{Game}_3} = \begin{cases} 0 & \text{if } \Pr[E] = 0; \\ \Pr[E_1|E] & \text{otherwise.} \end{cases}$$

We say that a key evolution scheme is ϵ-secure (against (c, s, q)-adversaries in Game_3*) if for every (c, s, q) adversary $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_b)$, the advantage $Adv_{\mathcal{A}}^{\mathsf{Game}_3}$ is at most ϵ for all rounds u.*

3 Quasilinear-time Key Evolution Schemes

The graph of our scheme consists of a stack of copies of a bipartite graph B, which itself is built from a "base" graph G with special properties: vertex expansion and locality. In the following, we will explain these two properties and our main construction. Then, we will show that our construction can be updated in quasilinear time (Theorem 2) and is secure (Theorem 3).

Definition 4. *An undirected graph $G = (V, E)$ is a (K, A) vertex expander if for every $S \subset V$ and $|S| \leq K$, $|\Gamma(S)| \geq A|S|$, where $\Gamma(S) = \{v | u \in S \wedge (u, v) \in E\}$.*

Definition 5. *An undirected graph $G = (V, E)$ on n vertices $V = \{1, 2, \dots, n\}$ is δ-*local *if $|i - j| \leq \delta n$ for every $(i, j) \in E$.*

The DKW scheme is based on the cycle graph, which is δ-local with $\delta = 1/n$. Unfortunately, the cycle is a poor expander. We show that by letting δ be a small constant, we can in fact get asymptotically good expanders.

Theorem 1 (Existence). *For every $\delta > 0$, there exists a constant d such that, for all sufficiently large n, there exists a d-regular δ-local graph G that is a $(\frac{4n}{5}, A = 1 + \frac{\delta}{2})$ vertex expander.*

To prove the theorem, we show that the probability that a *random* d-regular δ-local graph is a $(\frac{4n}{5}, 1+\frac{\delta}{2})$ vertex expander is close to 1, when d is a sufficiently large constant. The details are shown in Appendix B.

Definition 6 (Double Cover Graph). *Let $G = (V, E)$ be an undirected graph on n vertices (without loss of generality, let $V = \{1, \ldots, n\}$). Its double cover graph $B(G)$ is a directed bipartite graph $(L \cup R, E')$ such that $L = R = V$. Edges in E' are directed from L to R and therefore $E' \subset L \times R$. For each $(u, v) \in E$, we add (u, v) and (v, u) to E'. Furthermore, we add (i, i) to E' for $i = 1, \ldots, n$.*

Note that the input and output sets of $B(G)$ are L and R. We say $B(G)$ is δ-local if G is δ-local.

Definition 7. *Let G be an undirected graph on n vertices. Given $h \in \mathbb{N}^+$, a* stack of h copies of G, *denoted $\Gamma(G, h)$, is a layered DAG defined as follows: Let $B_1, \ldots, B_h$ be h copies of the double cover graph $B(G)$, and identify the outputs of B_i with the inputs of B_{i+1} for $1 \le i < h$, so $\Gamma(G, h)$ has $h + 1$ layers and $n(h + 1)$ vertices.*

Now, we are ready to describe our construction:

Definition 8 (Our Construction). *The graph $G_{\mathcal{K}}$ of our scheme is defined by two ingredients: (a) an undirected graph G, which is assumed to be a d-regular, δ-local, $(\frac{4n}{5}, A = 1 + \frac{\delta}{2})$-vertex expander for constants $d \in \mathbb{N}$ and $\delta > 0$; (b) an integer parameter t. Then, $G_{\mathcal{K}}$ is $\Gamma(G, M)$, where $M = th$, $h = c_A \log n$ and c_A is a constant depending on A.*

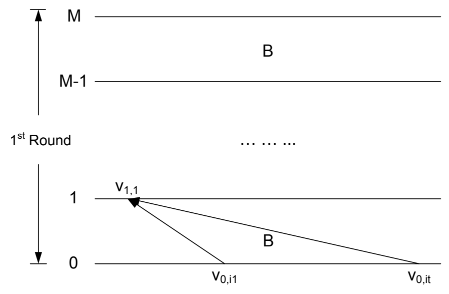

Fig. 1. Key Evolution Scheme as a Graph $G_{\mathcal{K}} = \Gamma(G, M)$.

Locality ensures that $G_{\mathcal{K}}$ can be updated in quasilinear time:

Theorem 2 (Efficiency). *The update function $\mathcal{K}$ defined by our construction can be computed in time $O(tn \log n)$ (assuming constant-time oracle calls) and $(1 + 2\delta)nw$ space.*

Proof. To evaluate $\mathcal{K}$, it suffices to evaluate the labels of the outputs of $\Gamma(G, M)$. Let $V_0, \ldots, V_M$ be the M layers of $\Gamma(G, M)$. At the beginning, assign n vertices at V_0 their own values, using the input. To evaluate the j-th vertex at V_1, as $B(G)$ satisfies δ locality, we need to know the values of at most $2\delta n$ vertices (e.g., those in $[j - \delta n, j + \delta n]$). Suppose that we have evaluated the j-th vertex and we now want to evaluate the

$(j+1)$-st vertex in V_1. We need to keep values for at most the inputs in $[j+1-\delta n, j+1+\delta n]$. Therefore, we can forget the value of the $(j-\delta n)$-th vertex at V_0 and evaluate the $(j+1)$-th vertex in V_1. Continuing in this manner, we need to keep values of at most $2\delta n + n$ vertices in $\Gamma(G, M)$ in memory. For every k, given that all vertices at V_k are evaluated, the number of random oracle calls to evaluate all vertices at V_{k+1} is $O(n)$. The total number of oracle calls to evaluate $\Gamma(G, M)$ is $O(nM) = O(tn \log n)$ as $M = tc_A \log n$ where c_A is a constant depending on A. □

The parameter t is set based on the class of adversaries that we consider. More specifically, if q is the number of random oracle calls made by an adversary $\mathcal{A}$, then our key evolution scheme can be updated in time $O(n \log n \log q)$, which is quasilinear in n:

Theorem 3 (Security). *For all $w, q, \lambda \in \mathbb{N}$, $c, s > 0$, $0.06 \geq \delta > 0$ and large enough n, if $\frac{4c+s+2\lambda}{w - \log q} \leq 1.12n$ and $t > \frac{\log q}{\log 1.01} + 3$, then the key evolution scheme $\mathcal{K}$ is $(\frac{q}{2^w} + 2^{1-\lambda})$-secure against (c, s, q) adversaries in the random oracle model.*

Proof. This is a corollary to Theorem 4. When $(1.01)^{t-3} > q$ and n is large enough, we have:

$$q < (1.01)^{t-3} < \frac{n}{2d+1}\left(\frac{2.1n}{2n+4.1}\right)^{t-3}.$$

Therefore, we can set $t > \frac{\log q}{\log 1.01} + 3$. □

Note that our key evolution scheme is meaningful when $s > (1+2\delta)nw$, where δ can be arbitrarily small. The DKW scheme is $(\frac{q}{2^w} + 2^{-\lambda})$-secure when $\frac{4c+s+2\lambda}{w-\log q} < 1.5n$, which exhibits a better leading constant than our bound, $\frac{4c+s+2\lambda}{w-\log q} < 1.12n$. However, their scheme needs $O(n^2)$ time to update while our scheme requires a quasilinear time only. In summary, if $4c + s \leq 1.12|K|$, our scheme does exist, is secure according to Definition 3 and can be updated in a quasilinear time.

4 Pebbling Games and Random Oracle Models

In this paper, we apply graph theory, specifically pebbling games [9], to prove security in the random oracle model. Specifically, we can translate any computation involving random oracle calls into a pebbling game where we can pebble colors on a graph. For example, considering $r = H(r_1, r_2)$ where H is a random oracle, if the adversary $\mathcal{A}$ knows the value of r, intuitively, $\mathcal{A}$ should also know both r_1 and r_2. Otherwise, the probability that $\mathcal{A}$ can guess r correctly is negligible. To capture this, we can construct a graph (V, E) such that $V = \{v, v_1, v_2\}$ and $E = \{(v_1, v), (v_2, v)\}$. The values associated with v, v_1, v_2 are $r(v) = r, r(v_1) = r_1, r(v_2) = r_2$. Then, we define a pebbling strategy by placing a color (i.e., a pebble) on $u \in V$. For $u \in V$ with indegree 0, we can place a pebble if we know the value $r(u)$; for $u \in V$ with indegree $\neq 0$, if all predecessors of u have been pebbled, we can place a pebble on u. In our example, v can be pebbled, if v_1 and v_2 get pebbled first, because both (v_1, v) and (v_2, v) are in E. The above rules to pebble a vertex v capture the computation process how we evaluate $r(v)$ via H. More generally, we can define a pebbling game as follows with two colors: black and red (with respect to $\mathcal{A}_s$ and $\mathcal{A}_b$).

Definition 9 (Pebbling Rules).

1. *A red pebble can be placed on any vertex already containing a black pebble.*
2. *If all predecessors of a vertex v are pebbled (each may contain different colors), a black pebble can be placed on v.*
3. *A black pebble can be removed from any vertex.*

Definition 10. *Let $G_{\mathcal{K}}$ be a key evolution scheme (as a graph). $\overline{G}$ is defined as an infinite stack of copies of $G_{\mathcal{K}}$. For $k \in \mathbb{N}^{+} \cup \{0\}$, we define V_k to be the set of vertices on the k-th layer of $\overline{G}$. Moreover, $V_{\geq k} = \cup_{i=k}^{+\infty} V_i$.*

Considering a (c, s, q) adversary $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_b)$ that plays with Game_3, we will have a transcript on when $\mathcal{A}$ calls the random oracle H and with which inputs. The input values are associated with vertices in $\overline{G}$. As we have discussed, we can convert this transcript into a pebbling strategy consisting of a series of pebbling moves. The resulting pebbling strategy is called ex-post-facto [9,10]. In fact, we can show that, for all values that $\mathcal{A}$ evaluates via the random oracle, the associated vertices will be pebbled as well in the ex-post-facto strategy in $\overline{G}$, with exactly the same order of random oracle calls [9,10]. Moreover, given c and s, we can bound the number of pebbles used by the ex-post-facto strategy:

Definition 11 (X-bounded). *Let B_u be the maximum number of black pebbles on $V_{\geq uM}$ in the u-th round. Let R_u be the number of times that rule 1 (Definition 9) is applied in the u-th round. Given $X \in \mathbb{R}$, we say that a pebbling strategy Ψ is X-bounded if for each round $u \geq 0$:*

$$2R_u + B_u \leq X.$$

The following lemma follows from [10]. In this paper, we explicitly investigate the relationship between the number of random oracle calls q (made by $\mathcal{A}$) and the number of moves in the resulting ex-post-facto strategy:

Lemma 1. *Consider the key evolution scheme as a graph $G_{\mathcal{K}}$ with a constant degree d and $H : \{0,1\}^{dw} \to \{0,1\}^{w}$ as a random oracle. Let $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_b)$ be a (c, s, q) adversary in the random oracle model. Let Ψ be its ex-post-facto strategy. Then, for any $\lambda > 0$, Ψ is valid consisting of at most $(2d+1)q$ moves and is $\frac{4c+s+2\lambda}{w-\log q}$-bounded with probability at least $1 - \frac{2}{2^{\lambda}} - \frac{q}{2^{w}}$. The probability is over the choice of the random oracle H and the 0-th round key K_0.*

Proof. This is a corollary of Theorem 4.10 [10]. Specifically, given each oracle call made by $\mathcal{A}$, the reduction algorithm will generate at most two moves (i.e., placing a red pebble and then removing a black pebble) for each inputs of the call. Moreover, it produces one move for each output. As the random oracle $H : \{0,1\}^{dw} \to \{0,1\}^{w}$ takes d inputs and there are at most q oracle calls made by $\mathcal{A}$, the reduction algorithm can produce at most $(2d+1)q$ moves. □

5 Security Analysis

Theorem 4. *For all $w, t, d \in \mathbb{N}$, $\lambda > 0$, let $H : \{0,1\}^{dw} \to \{0,1\}^{w}$ be a random oracle. If $\frac{4c+s+2\lambda}{w-\log q} \leq 1.12n$ and $q \leq \frac{n}{2d+1}(\frac{2.1n}{2n+4.1})^{t-3}$, then our key evolution scheme $G_{\mathcal{K}}$ is $(2^{1-\lambda} + \frac{q}{2^{w}})$-secure against any (c, s, q) adversaries in the random oracle model (Definition 3).*

To prove Theorem 4, we notice that, based on Lemma 1, the transcript for any adversary can produce a valid X-bounded pebbling strategy with high probability. Therefore, if we can show that, for any X-bounded pebbling game, no one can win as Definition 3, then we proved Theorem 4. More specifically, in term of pebbling game, at the end of any round u, Definition 3 asks that no one can pebble any vertices on $V_{(u+1)M}$ and then outputs the u-th round key.

To prove this, we show two lower bound results. The first lower bound shows that if the adversary can pebble any vertex on $V_{(u+1)M}$, at some point t, there do exist many pebbles between $V_{(u+1)M}$ and V_{uM}. This follows from the fact that, at each round u, $G_{\mathcal{K}}$ is a stack of t copies of a n-superconcentrator $\Gamma(G, h)$. The second lower bound shows that if the adversary can pebble all vertices at V_{uM} at the end of u-th round, at any time, there must be sufficient pebbles on or below V_{uM}. This is based on the fact that G is a vertex expander and h is large enough. However, based on these two lower bound results, at the point t, the number of pebbles on $\overline{G}$ exceeds the X-bounded assumption for a given X, which shows a contradiction.

In the following sections, we first prove that $\Gamma(G, h)$ is a n-superconcentrator. Then, we show the first and the second lower bound results. Proof of Theorem 4 is shown in Appendix C.

5.1 n-superconcentrator

Definition 12 (n-superconcentrator). *A graph G with input set I and output set O ($|I| = |O| = n$) is a n-superconcentrator provided that given any $S \subset I$ and $T \subset O$ with $|S| = |T| = k \leq n$, there are k vertex disjoint paths connected with S and T.*

We seek a n-superconcentrator construction that is a layered graph such that to pebble vertices in the i-th layer requires vertices in the $(i-1)$-th layer only. Moreover, it is built upon good expanders with $A > 1$ and therefore the number of layers can be logarithmic in n. The following lemma follows from the max-flow-min-cut theorem, whose proof can be found in [19].

Lemma 2. *Let G be a d-regular layered graph with the height h. Let I and O be its input set and output set such that $|I| = |O| = n$. Specifically, I is the 0-th layer and O is the $(h+1)$-th layer. For any $k \leq n$, for every $S \subset I$ and $T \subset O$ with $|S| = |T| = k$: there exist k vertex-disjoint paths from S to T if and only if removing any set of k-1 vertices from $V(G) - I - O$ cannot disconnect S from T.*

Lemma 3. *There exists $c > 0$, $\forall A > 1$, if $h \geq c \log_A n$, $\Gamma(G, h)$ is a n-superconcentrator when G is a $(\frac{4n}{5}, A)$ vertex expander.*

Proof. For any $k \leq n$, we want to show that vertices in S can reach T. Specifically, let $A_0 = S$ (and A_i be the set of vertices at i-th layer that are reachable from A_0). Similarly, we can define $B_0 = T$ and B_i. We want to show that at some point t: $|A_t \cap B_t| \neq \emptyset$. Intuitively, after a sufficient many t layers, $|A_t|$ will $> \frac{4n}{5}$ (if without removing $k-1$ vertices on the graph) because that G is a $(\frac{4n}{5}, A)$ vertex expander. Unfortunately, we may remove $k > m \geq \frac{n}{20}$ pebbles on the t-th layer which results that $|A_t| < \frac{3n}{4}$ (we count this as a "bad event"). However, we observe that we can grow it up again by vertex expansion property and the fact that there cannot be too many times of bad events (as $k \leq n$). The detailed proof can be found in Appendix D.

5.2 Lower Bound Results and Security Proof

In this section, we show the first lower bound via a modified Basic Lower Bound Argument (BLBA) lemma. Then, we introduce necessary tools (e.g., optimal width) to prove for the second lower bound (Theorem 5) and consequently the main security theorem (Theorem 4).

Lemma 4 ((modified) BLBA Lemma [15]). *In order to pebble S_b+S_e+1 outputs of a n-superconcentrator, starting with a configuration of at most S_b black and red pebbles and finishing with a configuration of at most S_e black and red pebbles on the graph, at least $N - S_b - S_e$ different inputs of the graph have to be pebbled and unpebbled.*

Proof. We prove it by contradiction. Suppose the claim is not true, then $\geq S_b + S_e + 1$ different inputs are not both pebbled and unpebbled. On the other hand, since $S_b + S_e + 1$ outputs of a n-superconcentrator are pebbled, there are $S_b + S_e + 1$ vertex-disjoint paths connecting those $S_b + S_e + 1$ inputs and outputs. Those vertex-disjoint paths have to be pebbled at some point. Initially there are at most S_b black and red pebbles and at the end there are at most S_e black and red pebbles, therefore, at least one of the path does not receive pebbles at the beginning point and at the ending point. Therefore, the input on this path has to be pebbled and unpebbled, which is contradiction with our assumption. □

Lemma 5 (First Lower Bound). *Let S be the maximum number of pebbles on $Y_{u+1} = \bigcup_{i=uM+1}^{(u+1)M} V_i$ at any configuration of the u-th round. Let $T(n, M, S)$ be the minimum number of moves needed to pebble one output of $V_{(u+1)M}$ during the u-th round. Assuming that at the initial configuration of the u-th round, there is no pebble on or above V_{uM}, then,*

$$T(n, M, S) > n(\frac{n-2S}{2S+1})^{t-3}.$$

Proof. We know that at the initial configuration of the u-th round, there are no pebbles on or above V_{uM}. In order to pebble any one output of $V_{(u+1)M}$, we have to pebble all its predecessors. Specifically, Y_{u+1} consists of t n-superconcentrators $C_1, \ldots, C_t$. V_{uM} is the input set of C_1 and $V_{(u+1)M}$ is the output set of C_t. Furthermore, C_t is empty initially. Therefore, we have to pebble all n inputs of C_t. On the other hand, the input set of C_t is identical to the output set of C_{t-1}. By applying (modified) BLBA, we need to pebble and unpebble $\lfloor \frac{n}{2S+1} \rfloor (n-2S)$ inputs of C_{t-1}. Moreover, we know that the output set of C_{t-2} is identical to the input set of C_{t-1}. In order to pebble $\lfloor \frac{n}{2S+1} \rfloor (n-2S)$ outputs of C_{t-2}, starting with any configuration on the graph, we can apply (modified) BLBA on C_{t-2}. We have $\lfloor \frac{n}{2S+1} \frac{n-2S}{2S+1} \rfloor (n-2S)$ of the inputs of C_{t-2} have to be pebbled and unpebbled. Iterating this BLBA argument to $C_{t-3}, \ldots, C_1$, we have

$$T(n, M, S) \geq \lfloor n(\frac{n-2S}{2S+1})^{t-3} \rfloor. \quad \square$$

Definition 13 (Optimistic Width). *Let $\Gamma(G, h)$ be the n-superconcentrator (cf. Definition 7) that is built from a $(\frac{4n}{5}, A)$-vertex expander. Then, the optimistic width of a list of integers $(a_0, \ldots, a_{k-1})$ w.r.t. $\Gamma(G, h)$ is, $OptWidth(a_0, \ldots, a_{k-1}) = (b_0, \ldots, b_{k-1})$ where:*

$$b_i = \begin{cases} a_0 & i = 0; \\ \min\{n, a_i + \frac{b_{i-1}}{A}\} & i > 0 \text{ and } b_{i-1} < \frac{4nA}{5}; \\ \min\{n, a_i + b_{i-1}\} & i > 0 \text{ and } b_{i-1} \geq \frac{4nA}{5}. \end{cases}$$

Definition 14. *Let G be a layered graph with k layers. Then, the projection of $V' \subset V(G)$: $proj(V') = (|V' \cap V_0|, \ldots, |V' \cap V_{k-1}|)$ (V_i is the set of vertices at the i-th layer of G).*

Definition 15. *Let $G = (V, E)$ be a graph. $\forall S \subset V$, the closure of S, denoted $[S]$, is defined recursively: (i) if $v \in S$, $v \in [S]$; (ii) if all children of v (i.e., $\{u | (u, v) \in E\}$) are in $[S]$, $v \in [S]$.*

Intuitively, $[S]$ includes all possible pebbles that can be derived from S. For any $k \in \mathbb{N}^+$, Let $\boldsymbol{v} = (v_0, \ldots, v_{k-1}) \in \mathbb{R}^k$ and $\boldsymbol{u} = (u_0, \ldots, u_{k-1}) \in \mathbb{R}^k$ be any two vectors. We say $\boldsymbol{v} \geq \boldsymbol{u}$ if and only if $v_i \geq u_i$ for all $i = 0, \ldots, k-1$.

Lemma 6. *The optimistic width w.r.t. $\Gamma(G, h)$ satisfies the following two properties.*

1. ***Upper Bound:*** *For any $U \subset V(\Gamma(G, h))$, $OptWidth(proj(U)) \geq proj([U])$.*
2. ***Addition:*** *Let $U, W \subset V(\Gamma(G, h))$. Let $(s_0, \ldots, s_{k-1}) = OptWidth(proj(U))$ and Let $(t_0, \ldots, t_{k-1}) = OptWidth(proj(U \cup W))$. Then, $0 \leq t_i - s_i \leq |W|$ for $i = 0, \ldots, k-1$.*

Proof. The proof is shown in Appendix E.

Definition 16 (Fully Covering Assumption). *Let r_u^* be the last configuration of the u-th round ($u \geq 0$). Then, all vertices on V_{uM} have to be pebbled at r_u^*.*

Definition 17. *A vertex is* **heavy** *if it contains a black pebble or a red pebble derived using the pebbling rule 1.*

Theorem 5. *For $u \geq 0$, let r_u^* be the last configuration of the u-th round. Let Heavy be the set of heavy pebbles. Define Q_u to be the set of pebbles at r_u^* except black pebbles at the uM-th layer. Let $Y_u = \cup_{i=(u-1)M+1}^{uM} V_i$. Under the fully covering assumption and X-bounded assumption ($1.12n > X > n$), for every round u and every configuration r at the u-th round, we have:*

1. $P_1(u) : OptWidth(proj(Q_u \cap Heavy)) < (2(X-n), \dots, 2(X-n), X-n, \dots, X-n, 1, \dots, 1)$.
2. $P_2(u)$: *(Second Lower Bound) Let $P(u,r)$ be the set of heavy red pebbles in Y_u derived in the u-th round and black pebbles in Y_u at the configuration r. Then,*
$$|P(u,r)| > 2n - X.$$
3. $P_3(u)$: *For any adversary $\mathcal{B}$ that makes at most T moves, at the end of round u ($u \geq 0$), there are no pebbles on or above $V_{(u+1)M}$, provided that $T \leq n(\frac{2.1n}{2n+4.1})^{t-3}$ (recall that $M = th$).*
4. $P_4(u)$: *Let S_u be the number of red pebbles on V_{uM} of the configuration r_u^*. Let S_u^* be the number of heavy red pebbles in Y_u derived in the u-th round. Then,*
$$S_u \leq S_u^*.$$

Proof. We prove it by induction on round number u. For $u = 0$, we have $|P(u,r)| = n > 2n - X$ if $X > n$. Moreover, $OptWidth(proj(Q_u \cap Heavy)) = OptWidth(proj(\emptyset)) < (2(X-n), \dots, 2(X-n), X-n, \dots, X-n, 1, \dots, 1)$. Initially, there are n pebbles on V_0 at the end of 0-th round. Therefore, $P_3(0)$ and $P_4(0)$ are trivially true. We assume the claim is true for $u \leq k$ and we prove the claim still holds for $(k+1)$-th round. Specifically, we prove the following lemmas hold:

1. $P_1(k) \Rightarrow P_2(k+1)$;
2. $P_3(k) \bigwedge P_2(k+1) \Rightarrow P_3(k+1)$;
3. $P_4(k) \bigwedge P_2(k+1) \Rightarrow P_4(k+1)$;
4. $P_1(k) \bigwedge P_3(k+1) \bigwedge P_4(k+1) \Rightarrow P_1(k+1)$. □

Proof ($P_1(k) \Rightarrow P_2(k+1)$). We have $proj([(Q_k \cup P(k+1,r)) \cap Heavy]) = proj([Q_k \cup P(k+1,r)])$ because all non-heavy pebbles are derived by heavy red ones. By the fully covering assumption, at the end of $(k+1)$-th round, $V_{(k+1)M}$ will be fully covered (by n pebbles). Therefore, the $(k+1)M$-th entry: $proj([Q_k \cup P(k+1,r)])_{(k+1)M} = n$. Hence, we have $proj([(Q_k \cup P(k+1,r)) \cap Heavy])_{(k+1)M} = n$. On the other hand, $(Q_k \cup P(k+1,r)) \cap Heavy = (Q_k \cap Heavy) \cup P(k+1,r)$ because $P(k+1,r)$ contains heavy pebbles only. By Lemma 6 part (1), we have

$$\begin{aligned} &OptWidth(proj((Q_k \cap Heavy) \cup P(k+1,r)))_{(k+1)M} \\ &= OptWidth(proj((Q_k \cup P(k+1,r)) \cap Heavy))_{(k+1)M} \\ &\geq proj([(Q_k \cup P(k+1,r)) \cap Heavy])_{(k+1)M} = n. \end{aligned}$$

However, by $P_2(k)$, we have $OptWidth(proj(Q_k \cap Heavy))_{(k+1)M} < X - n$. Therefore, $|P(k+1,r)| > n - (X-n) = 2n - X$ by Lemma 6 part (2). □

Proof ($P_3(k) \bigwedge P_2(k+1) \Rightarrow P_3(k+1)$). We prove it by contradiction. Assume that there exists an adversary $\mathcal{B}$ that makes at most $n(\frac{2.1n}{2n+4.1})^{t-3}$ moves can pebble some vertex in $V_{\geq(u+1)M}$. Since $P_3(k)$ is true, at the beginning of the $(k+1)$-th round, there are no pebbles on or above $V_{(k+1)M}$. Therefore, by the second lower bound lemma (Lemma 5), there exists a configuration r in the $(k+1)$-th round, such that $\mathcal{B}$ needs $> S = \frac{n}{4.1}$ space in Y_{k+2}. On the other hand, By $P_2(k+1)$, in the configuration r, there are $|P(k+1,r)| > 2n - X$ pebbles in Y_{k+1}. $Y_{k+1} \cap Y_{k+2} = \emptyset$. Therefore, the space needed in that configuration r is $> 2n - X + \frac{n}{4.1} \geq X$ when $X \leq 1.12n$, which shows a contradiction, given the X-bounded assumption. □

Proof $(P_4(k) \bigwedge P_2(k+1) \Rightarrow P_4(k+1))$. First, we prove that for any configuration r of the $(k+1)$-th round, there is no layer with $\geq \frac{4n}{5}$ red pebbles between the $(kM+1)$-th and the $(k+1)M$-th layer. We prove it by contradiction. Suppose that there exists a layer V_m such that there are $\geq \frac{4n}{5}$ red pebbles on it. Let T_0 be the set of heavy red pebbles between V_{kM+1} and V_m and T_1 be the set of heavy red pebbles on V_{kM}. By the property of n-superconcentrator, $|T_0| + |T_1| \geq \frac{4n}{5}$. Therefore, either $|T_0| \geq \frac{4n}{5} - \frac{X}{2}$ or $|T_1| \geq \frac{X}{2}$. We prove that both $|T_0| < \frac{4n}{5} - \frac{X}{2}$ and $|T_1| < \frac{X}{2}$ when $X \leq 1.12n$. We prove both of them by contradiction. First, we prove $|T_0| < \frac{4n}{5} - \frac{X}{2}$ given X-bounded assumption and $P_2(k+1)$. We assume that $|T_0| \geq \frac{4n}{5} - \frac{X}{2}$. By definition, we have $|T_0| \leq R_{k+1}$. By X-bounded assumption, we have $2R_{k+1} + B_{k+1} < X$. Therefore, $R_{k+1}+B_{k+1} < X-(\frac{4n}{5}-\frac{X}{2}) = \frac{3X}{2}-\frac{4n}{5}$. On the other hand, according to $P_2(k+1)$: $|P(k+1,r)| > 2n-X$. As $|P(k+1,r)| \leq R_{k+1} + B_{k+1} < \frac{3X}{2} - \frac{4n}{5}$, this leads to a contradiction because $2n - X \geq \frac{3X}{2} - \frac{4n}{5}$ when $X \leq 1.12n$.

Then, we prove for $|T_1| < \frac{X}{2}$ by induction hypothesis $P_4(k)$ and X-bounded assumption. On the contrary, we assume that $|T_1| \geq \frac{X}{2}$. First, by $P_4(k)$, we have $S_k \leq S_k^*$. By definitions, we have $|T_1| \leq S_k$ and $S_k^* \leq R_k$. Therefore, we have $R_k \geq \frac{X}{2}$. On the other hand, by X-bounded assumption, we have $R_k < \frac{X}{2}$. This leads to a contradiction.

Now, let t_i be the number of heavy pebbles on $V_{(k+1)M-i}$. We also assume that $S_{k+1} > S_{k+1}^*$. Since there is no layer with $\geq \frac{4n}{5}$ red pebbles between $V_{(k+1)M}$ and V_{kM+1}, we can use the expansion property. Specifically, we know that there are $S_{k+1} - t_0$ non-heavy red pebbles on $V_{(k+1)M}$. They are derived by $A(S_u - t_0) - t_1$ non-heavy red pebbles on $V_{((k+1)M-1}$. Apply the same argument till V_{kM+1}, we have:

$$S_k \geq A^{M-1} S_k - \sum_{i=0}^{M-1} A^{M-1-i} t_i.$$

On the other hand, we have $\sum_{i=0}^{M-1} A^{M-1-i} t_i \leq A^{M-1} \sum t_i \leq A^{M-1} S_{k+1}^*$. Therefore, we have:

$$S_k \geq A^{M-1}(S_{k+1} - S_{k+1}^*) > A^{M-1} \geq \frac{4n}{5}$$

when $M \geq \log_A \frac{4n}{5} + 1$.

However, by $P_4(k)$, we have $S_k^* \geq S_k \geq \frac{4n}{5}$. On the other hand, by X-bounded assumption (for $X \leq 1.12n$), $0.56n \geq \frac{X}{2} > S_k^*$ which implies a contradiction. Therefore, $S_{k+1}^* \geq S_{k+1}$. □

Proof $(P_1(k) \bigwedge P_3(k+1) \bigwedge P_4(k+1) \Rightarrow P_1(k+1))$. Let T be the set of heavy pebbles of r_{k+1}^* ; T^- is identical to T except black pebbles on $V_{(k+1)M}$. Then, we have:

$$\begin{aligned} OptWidth(Q_{k+1} \cap Heavy) &= OptWidth((Q_k \cup T^-) \cap Heavy) \\ &= OptWidth((Q_k \cap Heavy) \cup (T^- \cap Heavy)) \\ &= OptWidth((Q_k \cap Heavy) \cup T^-). \end{aligned}$$

On the other hand, we show that $|T^-| < X - n$. By the fully covering assumption, we have $(n - S_{k+1})$ black pebbles on the $(k+1)M$-th layer. By the X-bounded assumption, we have:

$$|T^-| + (n - S_{k+1}) < R_{k+1} + B_{k+1}.$$

On the other hand, we have $S_{k+1} \leq S_{k+1}^* \leq R_{k+1}$ by $P_4(k+1)$ and the definition of R_{k+1} (recall that R_{k+1} is the number of heavy red pebbles derived in the $(k+1)$-th round). Therefore, $|T^-| < 2R_{k+1} + B_{k+1} - n < X - n$. Moreover, by $P_3(k+1)$, we know that T^- does not contain any pebbles on or above $V_{(k+2)M}$. By Lemma 6 and $P_1(k)$, we have:

$$\begin{aligned} &OptWidth(OptWidth(Q_{k+1} \cap Heavy) = OptWidth((Q_k \cap Heavy) \cup T^-) \\ &< (2(X-n), \ldots, 2(X-n), 2(X-n), \ldots, 2(X-n), X-n, \ldots, (X-n), 1, \ldots, 1). \quad \square \end{aligned}$$

References

1. A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, 2009.
2. N. Alon and M. Capalbo. Smaller explicit superconcentrators. In *SODA 2003*.
3. J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, 2010.
4. Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, 2010.
5. S. S. Chow, Y. Dodis, Y. Rouselakis, and B. Waters. Priactical leakage-resilient identity-based encryption from simple assumption. In *CCS*, 2010.
6. F. Davì, S. Dziembowski, and D. Venturi. Leakage-resilient storage. In J. A. Garay and R. D. Prisco, editors, *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2010.
7. Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC*, 2010.
8. Y. Dodis, K. Haralambiev, A. Lo'pez-Alt, and D. Wichs. Cryptography against continuous memory attacks. In *FOCS*, 2010.
9. C. Dwork, M. Naor, and H. Wee. Pebbling and proofs of work. In *CRYPTO 2005*.
10. S. Dziembowski, T. Kazana, and D. Wichs. Key-evolution schemes resilient to space-bounded leakage. In *CRYPTO 2011*.
11. S. Dziembowski, T. Kazana, and D. Wichs. One-time computable self-erasing functions. In Y. Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 125–143. Springer, 2011.
12. S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, 2008.
13. S. Faust, T. Rabin, L. Reyzin, E. Tromer, and V. Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, 2010.
14. Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
15. T. Lengauer and R. E. Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *Journal of ACM*, 29:1087 – 1130, 1982.
16. A. B. Lewko, M. Lewko, and B. Waters. How to leak on key updates. In *STOC*, 2011.
17. S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.
18. N. Pippenger. Superconcentrators. *SIAM Journal on Computing*, 6:298 – 304, 1977.
19. L. G. Valiant. On non-linear lower bounds in computational complexity. In *STOC*, 75.
20. Y. Yu, F.-X. Standaert, O. Pereira, and M. Yung. Practical leakage-resilient pseudorandom generators. In *CCS 2010*.
21. T. H. Yuen, S. S. M. Chow, Y. Zhang, and S. Yiu. Identity-based encryption resilient to continual auxilary leakage. In *EUROCRYPT*, 2012.

A Security Definition in the Standard Model

Definition 18. *Fix a round $u \geq 0$. Consider a security game Game_{18} between a challenger $\mathcal{C}$ and an adversary $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_b)$. $\mathcal{C}$ picks K_0 and T_0 uniformly at random. $\mathcal{C}$ also computes $T_1 = K_{u+1}$ via the key evolution scheme $\mathcal{K}$ and flips a random coin b uniformly. Initially, $\mathcal{A}_s$ is given K_0 and $\mathcal{A}_b$ is given T_b. At the end of each u'-th round where $u' < u$, $\mathcal{A}_s$ outputs $\tilde{K}_{u'}$ to $\mathcal{C}$. At the end of the u-th round, $\mathcal{A}_b$ outputs its guess bit b' to $\mathcal{C}$. Finally, $\mathcal{A}_s$ outputs $\tilde{K}_u$ to $\mathcal{C}$. It is important that $\tilde{K}_u$ is output after b' is received. Let E be the event that $\tilde{K}_i = K_i$ for all $i = 1, \ldots, u$. The advantage of $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_b)$ at round u is defined as:*

$$Adv_{\mathcal{A}}^{\mathsf{Game}_{18}} = |\Pr[(b' = b) \wedge E] - \frac{1}{2}\Pr[E]|.$$

We say that a key evolution scheme is ϵ-secure (against Game_{18}) if for every adversary $A = (\mathcal{A}_s, \mathcal{A}_b)$, the advantage $Adv_{\mathcal{A}}^{\mathsf{Game}_{18}} \leq \epsilon$ for all rounds u.

Note that if $\Pr[E] = 1$, $Adv_{\mathcal{A}}^{\mathsf{Game}_{18}} = |\Pr[b' = b] - \frac{1}{2}|$. We also note that, if $\mathcal{A}_s$ and $\mathcal{A}_b$ can communicate arbitrarily, $\mathcal{A}_s$ can send K_0 to $\mathcal{A}_b$ and $\mathcal{A}_b$ evaluates K_{u+1} itself. In this case, no key evolution scheme is secure for $\epsilon < \frac{1}{2}$. On the other hand, [10] shows that if c and s are with some restrictions, there do exist secure key evolution schemes in the random oracle model.

Theorem 6. *Let $\mathcal{A} = (\mathcal{A}_s, \mathcal{A}_b)$ be an adversary in the random oracle model. Then,*

$$Adv_{\mathcal{A}}^{\mathsf{Game}_{18}} \leq \frac{3}{2} Adv_{\mathcal{A}}^{\mathsf{Game}_3}.$$

Proof. When $\Pr[E] = 0$, $Adv_{\mathcal{A}}^{\mathsf{Game}_{18}} = |\Pr[(b' = b) \wedge E] - \frac{1}{2}\Pr[E]| = 0$. The claim is true. Otherwise,

$$\begin{aligned} Adv_{\mathcal{A}}^{\mathsf{Game}_{18}} &= |\Pr[(b' = b) \wedge E] - \frac{1}{2}\Pr[E]| \\ &\leq |\Pr[b' = b|E] - \frac{1}{2}| \\ &\leq \Pr[E_1|E] + |\Pr[b' = b|E \wedge \overline{E_1}]\Pr[\overline{E_1}|E] - \frac{1}{2}|. \end{aligned}$$

Let $\delta = \Pr[b' = b|E \wedge \overline{E_1}] - \frac{1}{2}$. Then, we have:

$$\begin{aligned} |\Pr[b' = b|E \wedge \overline{E_1}]\Pr[\overline{E_1}|E] - \frac{1}{2}| &= |(\delta + \frac{1}{2})\Pr[\overline{E_1}|E] - \frac{1}{2}| \\ &= |\Pr[\overline{E_1}|E]\delta - \frac{1}{2}(1 - \Pr[\overline{E_1}|E])| \\ &\leq |\Pr[\overline{E_1}|E]\delta - \frac{1}{2}\Pr[E_1|E]| \\ &\leq |\delta| + \frac{1}{2}\Pr[E_1|E]. \end{aligned}$$

Therefore, $Adv_{\mathcal{A}}^{\mathsf{Game}_{18}} \leq \frac{3}{2}\Pr[E_1|E] + |\delta| = \frac{3}{2}Adv_{\mathcal{A}}^{\mathsf{Game}_3} + |\delta|$. On the other hand,

$$\begin{aligned} |\delta| &= |\Pr[b' = b|E \wedge \overline{E_1}] - \frac{1}{2}| \\ &= 0. \end{aligned}$$

Specifically, if the event E_1 does not occur, neither $\mathcal{A}_s$ nor $\mathcal{A}_b$ evaluates $r_{(u+1),j}$ for any $j \in \{1, \ldots, n\}$ via the random oracle calls. Therefore, the value $K_{u+1} = (r_{(u+1),1}, \ldots, r_{(u+1),n})$ is uniformly random to both $\mathcal{A}_s$ and $\mathcal{A}_b$ information theoretically. Then, the probability that $\mathcal{A}_b$ guesses correctly on $b' = b$ is exactly $\frac{1}{2}$. □

B Proof of Theorem 1

In order to prove it, we need some lemmas (which will be proved later).

Definition 19. *For $\forall k \in \mathbb{N}^+, \delta > 0$, let $S \subset [1, n]$ with size $|S| = k$; $T \subset [1, n]$ of size Ak. Define $p_i(S, T) = \frac{|T \cap \Gamma^*(v_i)|}{|\Gamma^*(v_i)|} \leq 1$ (p_i for short) where v_i is the i-th element in S and $\Gamma^*(v_i) = [v_i - \delta n, v_i + \delta n]$.*

Lemma 7. *For $\forall 1 > \beta \geq 0$, If $\frac{1}{k}\sum_{i=1}^{k} p_i \leq 1 - \beta$, then there are at least $(1 - \sqrt{1 - \beta})k$ elements in S whose $p_i \leq \sqrt{1 - \beta}$.*

Proof. Suppose that $> \sqrt{1 - \beta}k$ elements in S whose $p_i > \sqrt{1 - \beta}$. Then, we have $\frac{1}{k}\sum_{i=1}^{k} p_i > \sqrt{1 - \beta}^2 = 1 - \beta$, which shows a contradiction. □

Lemma 8. *Let $A = 1 + \frac{\delta}{2}$. For any sets S, T with $|S| = k$ and $|T| = Ak$, where $\frac{4n}{5} \geq k \geq \frac{2\delta n+1}{A+1}$ and assuming that n is large enough, then*

$$\max_{S,T} \frac{1}{k}\sum_{i=1}^{k} p_i \leq 1 - \frac{5\delta}{16}$$

.

Lemma 9. *For $\frac{2\delta n-1}{A+1} \geq k \geq 1$, assuming n is large enough, we have $\frac{1}{k}\sum p_i \leq \frac{Ak}{2\delta n} < 1$.*

Proof (Theorem 1). We prove this theorem by considering a random d-regular graph. We show that such a graph is *good* with probability greater than 0. Therefore, there will exist such a *good* expander graph.

Specifically, given any $1 \leq k \leq \frac{4n}{5}$, any sets S, T such that $|S| = k, |T| = Ak$, we consider the probability that $p(S,T) = Pr[\Gamma(S) \subset T]$ where we define $\Gamma(S) = \{u \in T | \exists v \in S : (v,u) \in E\}$ that is the set of vertices in T which are reachable from S. Specifically, let $S = \{v_1, \ldots, v_k\} \subset [1,n]$. Then, we have $p(S,T) = \prod_{v_i \in S} {p_i}^d$, as the graph that we consider is d-regular. Let $q(k)$ be the probability that there exist some sets S and T ($|S| = k$ and $|T| = Ak$) where vertex expansion is not satisfied (then, the graph is not a (K, A) vertex expander). Specifically, we have

$$\begin{aligned} q(k) &= \binom{n}{k}\binom{n}{Ak} p(S,T) \\ &\leq (\frac{en}{k})^k (\frac{en}{Ak})^{Ak} p(S,T). \end{aligned}$$

There are two cases. First, if $k < \frac{2\delta n-1}{A+1}$, according to Lemma 9, we have $\frac{1}{k}\sum p_i \leq \frac{Ak}{2\delta n} < 1$. Therefore, we can pick up a constant c ($c > 1$) such that $\frac{Ak}{2\delta n} < \frac{1}{c} < 1$. Then, we have $1 - \sqrt{\frac{Ak}{2\delta n}} \geq 1 - \sqrt{\frac{1}{c}}$. Due to Lemma 7, we have:

$$\begin{aligned} q(k) &\leq (\frac{en}{Ak})^{Ak} (\frac{en}{k})^k (\sqrt{\frac{Ak}{2\delta n}})^{(1-\sqrt{\frac{1}{c}})dk} \\ &\leq \frac{e^{(A+1)k}}{A^{Ak}} (\frac{2\delta}{A})^{(A+1)k} (\frac{Ak}{2\delta n})^{\frac{1}{2}(1-\sqrt{\frac{1}{c}})dk-(A+1)k} \\ &\leq 2^{-100k} \end{aligned}$$

as $\frac{Ak}{2\delta n} < 1$ and when d is a large enough constant (depending only on δ).

In the second case, when $\frac{4n}{5} \geq k \geq \frac{1.9\delta n}{A+1}$ (note that $\frac{2\delta n-1}{A+1} > \frac{1.9\delta n}{A+1}$ when n is large), we have

$$\frac{en}{k} \leq \frac{(A+1)e}{1.9\delta}.$$

Due to Lemma 8, we know that $\frac{1}{k}\sum p_i < 1 - \frac{5\delta}{16}$. Therefore, according to Lemma 7,

$$\begin{aligned} q(k) &\leq (\frac{(A+1)e}{1.9\delta})^k (\frac{(A+1)Ae}{1.9\delta})^{Ak} \sqrt{(1 - \frac{5\delta}{16})}^{(1-\sqrt{1-\frac{5\delta}{16}})dk} \\ &\leq 2^{-100k} \end{aligned}$$

when d is a large enough constant (depending only on δ). Then, we have $\sum_{k=1}^{\frac{4n}{5}} 2^{-100k} < 1$. $\square$

Proof (Lemma 8). Consider the set S that is an interval. We compute $\frac{1}{k}\sum p_i$ by counting on each element in T. Specifically, there are $k-2\delta n+1$ elements that each connects with exact $2\delta n$ elements of S. Moreover, there are 2 elements connecting with exact $2\delta n-1$ elements; 2 elements connecting with $2\delta n-2$ elements of S and so on.

Therefore, we can come up with a greedy strategy that works as follow. We first select those elements that connect with exact $2\delta n$ elements in S. If there are still openings in Ak elements, then we choose the two elements that connect $2\delta n-1$ elements of S into T and so on. We claim that our strategy is indeed optimal because otherwise if there exists an optimized solution T' that is different in at least one element in T with ours. Then, we can sort the elements in T' according to how many elements in S connect with it. Then, we will find that the one in T' that is different with ours will have a lower rank. Now, we can switch this element with the element in our solution and the resulting set T'' has a larger $\frac{1}{k}\sum p_i$ than T'. This shows that the set T produced by our strategy is optimal.

Therefore, assuming $t=ak+\delta n$ where $a=\frac{A-1}{2}$, we have:

$$\begin{aligned}\frac{1}{k}\sum_{i=1}^{k}p_i &= \frac{2\delta n(k-2\delta n+1)+2(2\delta n-1)+\ldots+2(2\delta-t)}{2\delta nk}\\ &= \frac{\delta n-\delta^2n^2+2ak\delta n-a^2k^2-ak+2\delta nk}{2\delta nk}\\ &= 1+a-\frac{a}{2\delta n}-\frac{\delta n-1}{2k}-\frac{a^2k}{2\delta n}\\ &\leq 1+\frac{2ak-\delta n+1}{2k}.\end{aligned}$$

Moreover, if $k\leq\frac{4n}{5}$,

$$\begin{aligned}2ak-\delta n+1 &\leq 2\cdot\frac{\delta}{4}\frac{4n}{5}-\delta n+1\\ &\leq -\frac{3\delta n}{5}+1\leq-\frac{\delta n}{2}\end{aligned}$$

when $n\geq\frac{10}{\delta}$.

Therefore, we have $1+\frac{2ak-\delta n+1}{2k}\leq 1-\frac{\delta n}{4k}\leq 1-\frac{5\delta}{16}$.

The above argument is applicable when $k\geq 2\delta n$. For $\frac{2\delta n}{A+1}\leq k<2\delta n$, we have $\frac{1}{k}\sum p_i\leq\frac{\delta}{4}+\frac{3-A}{4\delta n}$. When, $n>\frac{(3-A)}{(1-9\delta/16)4\delta}$ is large enough, we have $\frac{\delta}{4}+\frac{3-A}{4\delta n}<1-\frac{5\delta}{16}$.

Now, we consider that S' is not an interval. Let S' be any set of size k but S' is not an interval. We claim that the value of $\max_{S',T}\frac{1}{k}\sum p_i$ is lower than $\max_{S,T}\frac{1}{k}\sum p_i$. We process S' from right v_1 to left v_k. There are two cases. First we assume that the distance $d(v_1,v_2)=|v_1-v_2|$ (as $v_i\in[1,n]$) between v_1 and v_2 is $>2\delta n$. Then, we move elements $\{v_2,v_3,\ldots,v_k\}$ in S towards v_1 with $d(v_1,v_2)-1$ steps. Moreover, we move $T\cap[-\infty,v_2+\delta n]$ with $d(v_1,v_2)-1$ steps. Since the distance $d(v_1,v_2)>2\delta n$, we don't remove any elements in T that are originally connected with v_1. Moreover, since we move $[v_2,v_k]$ and $T\cap[-\infty,v_2+\delta n]$ with the same amount steps, this does not change any connection from T to $[v_2,v_k]$. The only change is that when we move elements in T towards v_1, some elements in T are now connected with v_1 and this will increase $\frac{1}{k}\sum p_i$.

For the second case, if $d(v_1,v_2)\leq 2\delta n$, we do the same steps as in the first case. Therefore, those elements in T that connect with $[v_2,v_k]$ will still connect with them. However, we need to carefully count on elements in T that connects with v_1. We first note that when we move elements in T as $d(v_1, v_2) < 2\delta n$, those elements $T \cap [v_1 - \delta n, v_2 + \delta n]$ still are $\subset [v_1 - \delta n, v_1 + \delta n]$. On the other hand, they may now overwrite some elements that does not move (because they are $\in [v_2 + \delta n + 1, v_1 + \delta n]$), then we have some openings. But consider that, those elements are only related with v_1 (i.e., no elements in S except v_1 connect with them before the movement), we can do the greedy strategy to assign the openings. As a summary, for each of the elements moved, they still connect with the same elements in $[v_1, v_k]$; for those elements that are not moved, we assign them to new positions such that they can connect with even more elements of $S = [v_1, v_k]$. Inductively, we show that after we move all elements in S and elements in T, we will have the value of $\frac{1}{k}\sum p_i$ which is not lower than before. This shows that the case S that is an interval maximizes $\frac{1}{k}\sum p_i$. □

Proof (Lemma 9). Consider $|S| = k$ and $|T| = Ak$ as above. For each element in T, it connects with at most $2\delta n$ elements. Therefore, some elements in T may connect all the elements in S as $|S| = k$ and $k < 2\delta n$. There are at most $2\delta n - k + 1$ such elements. As $\frac{2\delta n - 1}{A+1} \geq k$, we know that $Ak \leq 2\delta n + k - 1$. Therefore, we can ask that all elements in T connect all elements of S where $|S| = k$. Then, in this case, we have:

$$\begin{aligned}\frac{1}{k}\sum p_i &= \frac{Ak \cdot k}{2k\delta n}\\ &= \frac{Ak}{2\delta n} < 1\end{aligned}$$

when n is large enough. □

C Proof of Theorem 4

Proof. When $\Pr[E] = 0$, by Definition 3, $Adv_{\mathcal{A}}^{\mathsf{Game}_3} = 0$. Therefore, we can assume that $\Pr[E] \neq 0$. Let E' be the event that the ex-post-facto strategy is valid, is $(\frac{4c+s+2\lambda}{w - \log q})$-bounded and consists of at most $n(\frac{2.1n}{2n+4.1})^{t-3}$ moves.

$$\begin{aligned}Adv_{\mathcal{A}}^{\mathsf{Game}_3} &= \Pr[E_1|E] = \Pr[E_1|E' \wedge E]\Pr[E'|E] + \Pr[E_1|\overline{E'} \wedge E]\Pr[\overline{E'}|E]\\ &\leq \Pr[E_1|E' \wedge E] + \Pr[\overline{E'}|E]\\ &\leq 0 + 2^{1-\lambda} + \frac{q}{2^w}.\end{aligned}$$

If $\mathcal{A}$ evaluates the value of any vertex v in $V_{\geq(u+1)M}$ in the u-th round, v will also be pebbled in the ex-post-facto strategy. Moreover, since $\mathcal{A}$ makes at most q queries where $q \leq \frac{n}{2d+1}(\frac{2.1n}{2n+4.1})^{t-3}$, the generated ex-post-facto strategy consists of at most $n(\frac{2.1n}{2n+4.1})^{t-3}$ moves. Therefore, if the ex-post-facto graph is valid with $1.12n$-bounded and with at most $n(\frac{2.1n}{2n+4.1})^{t-3}$ moves, according to Theorem 5 part (3), we will not pebble any vertex in $V_{\geq(u+1)M}$. Therefore, the probability $\Pr[E_1|E' \wedge E] = 0$. $\Pr[\overline{E'}|E] = 2^{1-\lambda} + \frac{q}{2^w}$ is due to Lemma 1. □

D Proof of Lemma 3

Proof. Due to Lemma 2, we have to prove that for any $k \leq n$, the set of U (where $|U| < k$) cannot disconnect S from T. Specifically, let $U_i \subset U$ is the set of vertices of U at the i-th layer. Let $A_0 = S$ and $B_0 = T$. $A_{i+1} = \Gamma(A_i) - U_{i+1}$ where $\Gamma(A_i)$ is the set of vertices at the $(i+1)$-th layer that are reachable from A_i. Similarly, we define $B_{i+1} = \Gamma(B_i) - U_{i+1}$. Moreover, we let $a_i = |A_i|$, $b_i = |B_i|$ and $u_i = |U_i|$. Then, for all i such that $a_i \leq \frac{4n}{5}$, we have $a_{i+1} \geq Aa_i - u_{i+1}$ because G is $(\frac{4n}{5}, A)$ vertex expander.

Let $t_i = a_i + b_{h-i}$. We want to show that $\frac{1}{h}\sum_{i=1}^{h} t_i > n$. Then, there exists $i^* \in [1, h]$ such that $t_{i^*} > n$. Otherwise, we have for all i: $t_i < n$ and therefore $\frac{1}{h}\sum t_i < n$ which leads a contradiction. On the other hand, $a_{i^*} + b_{h-i^*} > n$ implies that $|A_{i^*} \cap B_{h-i^*}| \neq \emptyset$ because each layer i^* contains n vertices. To show $\frac{1}{h}\sum_{i=1}^{h} t_i > n$, we can simply show that both $\frac{1}{h}\sum_{i=1}^{h} a_i > \frac{n}{2}$ and $\frac{1}{h}\sum_{i=1}^{h} b_i > \frac{n}{2}$.

Now we prove $\frac{1}{h}\sum_{i=1}^{h} a_i > \frac{n}{2}$ (the proof to $\frac{1}{h}\sum_{i=1}^{h} b_i > \frac{n}{2}$ can be done similarly.) We first assume that $a_0 = k \geq \frac{4n}{5}$. Otherwise, we can use expansion property to grow it up to $\frac{4n}{5}$ in $\Theta(\log n)$ layers. Specifically, for any t, if $a_{t-1} < \frac{4n}{5}$: we have $a_t > Aa_{t-1} - u_t$. Iteratively,

$$\begin{aligned} a_t &> A^t a_0 - \sum_{j=1}^{t} A^{t-j} u_j \\ &> A^t a_0 - A^{t-1} \sum_{j=1}^{t} u_j \\ &> A^t a_0 - A^{t-1}(k-1) > A^{t-1}. \end{aligned}$$

If we set $t \geq \log_A(\frac{4n}{5}) + 1$, $a_t > A^{t-1} \geq \frac{4n}{5}$.

Now, let's assume $a_0 > \frac{4n}{5} > \frac{3n}{4} > \frac{n}{2}$. Then, we investigate $a_1, a_2, \ldots$. We say that a_i is *bad* if $a_i < \frac{3n}{4}$. Once we encounter a *bad* a_i, we will grow it up to $\frac{4n}{5}$ again by the above analysis.

We claim that there are at most 20 bad a_i(s). This is true because each time it takes at least $\frac{4n}{5} - \frac{3n}{4} = \frac{n}{20}$ vertices in U to bring $a_i > \frac{4n}{5}$ to some $a_j < \frac{3n}{4}$. Therefore, this can happen at most $\frac{k}{n/20} \leq \frac{n}{n/20} = 20$ times.

Let $h = c\log_A n$ for some constant c that will decide soon. We want $\frac{1}{h}\sum_{i=1}^{h} a_i > \frac{n}{2}$. Specifically, we need

$$\begin{aligned} \frac{1}{h}\sum_{i=1}^{h} a_i &\geq \frac{[h - 20(t+1)]\frac{3n}{4}}{h} \\ &> \frac{[c\log_A n - 20(\log_A n + 2)]\frac{3n}{4}}{c\log_A n} > \frac{n}{2} \end{aligned}$$

when $c > 60$ and n is large enough. □

E Proof of Lemma 6

Proof. First, we prove for the upper bound property. Let $(s_0, \ldots, s_{k-1}) = \mathit{OptWidth}(\mathit{proj}(U))$ and $(t_0, \ldots, t_{k-1}) = \mathit{proj}([U])$. We have to show $s_i \geq t_i$ for all $i = 0, \ldots, k-1$. We prove this by introduction. To do so, we assume $(\alpha_0, \ldots, \alpha_{k-1}) = \mathit{proj}(U)$ and $(\beta_0, \ldots, \beta_{k-1}) = \mathit{proj}([U])$. For the base case $i = 0$, we see that $s_0 = \alpha_0 = \beta_0 = t_0$.

Suppose that the claim is true for $i-1$, we have $s_{i-1} \geq t_{i-1}$ (i.e., s_{i-1} is a upper bound on the number of pebbles can be derived at the $(i-1)$-th layer of U). We want to show a upper bound on the number of pebbles that can be derived at the i-th layer. Let S_0 be the set of pebbles at the i-th layer that can be derived from the $(i-1)$-th layer. On the other hand, there are a_i pebbles at the i-th layer of U. Let S_1 be the set of a_i pebbles at the i-th layer. At the worst case, $S_0 \cap S_1 = \emptyset$. Therefore, $t_i \leq \min\{a_i + |S_0|, n\}$. Now, we have to bound on $|S_0|$.

This can be done on $\Gamma(G, h)$ where G is a $(\frac{4n}{5}, A)$ vertex expander. Recall that the graph between the i-th (for any i) and the $(i+1)$-th layers is a bipartite graph $B(G)$. Let $L(B(G))$ be the left side of $B(G)$

(i.e., that corresponds to the $(i-1)$-th layer) and $R(B(G))$ be the right side of $B(G)$ (i.e., that corresponds to the i-th layer). Specifically, for any set $V \subset L(B(G))$ and $|V| \leq \frac{4n}{5}$, we need at least $A|V|$ pebbles (in $R(B(G))$) to derive it. In this case, we have $|S_0| \leq \frac{s_{i-1}}{A}$ provided that $s_{i-1} < \frac{4An}{5}$. On the other hand, if $|V| \geq \frac{4n}{5}$, we know that s_{i-1} pebbles can derive at most s_{i-1} pebbles, which shows the upper bound property.

Now we prove for the addition property. We prove it via adding elements of W into U one by one. Specifically, suppose that the added element e is in the j-th layer. Let $(s_0, \ldots, s_{k-1}) = OptWidth(proj(U))$ and $(s'_0, \ldots, s'_{k-1}) = OptWidth(proj(U \cup \{e\}))$. Moreover, let
$(a_0, \ldots, a_{k-1}) = proj(U)$. Then, for $i < j$, $s'_i = s_i$. For $i = j$, without loose of generality, we assume $s'_{j-1} < \frac{4nA}{5}$, we have

$$\begin{aligned} s'_j &\leq \frac{s'_{j-1}}{A} + a_j + 1 \\ &\leq \frac{s_{j-1}}{A} + a_j + 1 \\ &= s_j + 1 \end{aligned}$$

as $A > 1$ and $s'_{j-1} = s_{j-1}$. Then applying it to $i = j+1$ and, without loose of generality, assuming $s'_j \leq \frac{4n}{5}$, we have:

$$\begin{aligned} s'_{j+1} &\leq a_{j+1} + \frac{s'_j}{A} \\ &\leq a_{j+1} + \frac{(s_j + 1)}{A} \\ &\leq s_{j+1} + 1 \end{aligned}$$

as $A > 1$. Applying the same argument to $i = j+2, \ldots, k-1$, we complete the proof. □