Technical Perspective on 'History-Independent Dynamic Partitioning: Operation-Order Privacy in Ordered Data Structures'

Radu Sion
Stony Brook University
r@zxr.io

Background. Retention regulations mandate that once sensitive data is deleted, no evidence about its past existence should be recoverable, even by a determined adversary. Typically, data is physically deleted from the storage medium by using compliant erasure mechanisms.

Nevertheless, evidence of past deletes and of the existence of the deleted data itself cannot be eliminated by simply overwriting the data. This is because modern data processing comes with significant side-effects pervading all the layers of a computing system. To fully expire data and implement compliant erasure, transactions that have previously involved the data, across all key layers of the computing system, need to be identified and possibly rolled back.

Even in non-transactional systems, by the time of a compliant erasure event, a plethora of (meta)data structures (indexes, caches, etc.) would have already been irreversibly impacted by the existence of the deleted data. Adversaries can then infer, with higher probability than guessing, whether the sensitive data existed or not, thus leaking some if not all of the regulated information.

Examples are numerous and include: evidence of past existence of delete data , the order in which votes were cast in a voting application , sensitive intermediate versions of a published document , etc.

For a more in-depth example consider the fact that a typical search index looks very different before an insertion of a key, and after its subsequent deletion – and this is a problem if the key is related to sensitive regulated data, e.g., "Sarah" an HIV patient's name. The simple fact that the sensitive key existed at some point in the index, uniquely (as a function of the sensitive key) shapes the future of the index's state, even if that key is then subsequently deleted.

Furthermore, the write history itself implicitly leaves artifacts in the layout of the storage medium, that can be used to infer the past existence of deleted records. For example, the current layout of data blocks on disk is a function of the sequence and timing of previous writes to file system or database search indexes. Illicit questions such as "was Sarah's record ever in the database of patients" can then be answered by looking at the storage layout of the search index on disk (which will look different depending on whether Sarah has previously been in the data set or not).

History Independence. The general idea behind history independence (HI) is to design useful data structures for which the underlying representation is not impacted by the operational history of the structures, making them thus "history independent". This would immediately eliminate the above-discussed types of structural information leaks post-deletion of sensitive information.

A sizable number of successful efforts have resulted in HI variants of many popular data structures. For example, HI alternatives for the extremely important and popular classic B-tree data structure include the B-treap, the B-skip-list, history-independent external-memory skip lists, and history-independent cache-oblivious B-trees.

Nevertheless, history independence does not come for free: these data structures are less efficient then their non-HI counterparts, in both structural complexity and performance.

History Independent Partitioning. One of the main reasons for which B-trees are difficult to morph efficiently into a HI variant is the fact that their inherent node partitioning operations (e.g., node splitting / merging upon insertion / deletion of keys) are severely history dependent.

Basically the resulting partitions look very different as a function of history of keys and operation ordering. This problem extends further to a large number of data structures which deploy similar internal partitioning mechanisms.

A major and interesting insight of the work by Bender, Farach-Colton, Goodrich and Komlós is to realize that designing a HI dynamic partitioning scheme[1] would allow it to be plugged into any of the partitioning-based data structures to achieve an HI variant thereof!

Naturally, this may require a few more additional changes to preserve HI across the entire data structure, but such changes should not be of concern as long as they do not impact overall efficiency.

The authors introduce a **dynamic partitioning scheme that is strongly history-independent**, i.e., uniquely representable. An adversary cannot infer any additional information after seeing the structure data representation multiple times.

The mechanism is efficient with $O(1)$ insert / delete operations in expectation and $O(BlogN/loglogN)$ with high probability in set size N.

This is a great foundational result and we expect it to impact the field in the years to come as researchers will deploy this mechanism across other partition-based data structures.

[1]Dynamic partitioning maintains a partition of ordered groups each of size $\Theta(B)$, of a set of ordered elements where B is a size-parameter.