Recovering Deleted Data From the Windows Registry

By

Timothy Morgan

Presented At

The Digital Forensic Research Conference

DFRWS 2008 USA Baltimore, MD (Aug 11th - 13th)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

http:/dfrws.org

Recovering Deleted Data From the Windows* Registry

By Timothy D. Morgan

Outline

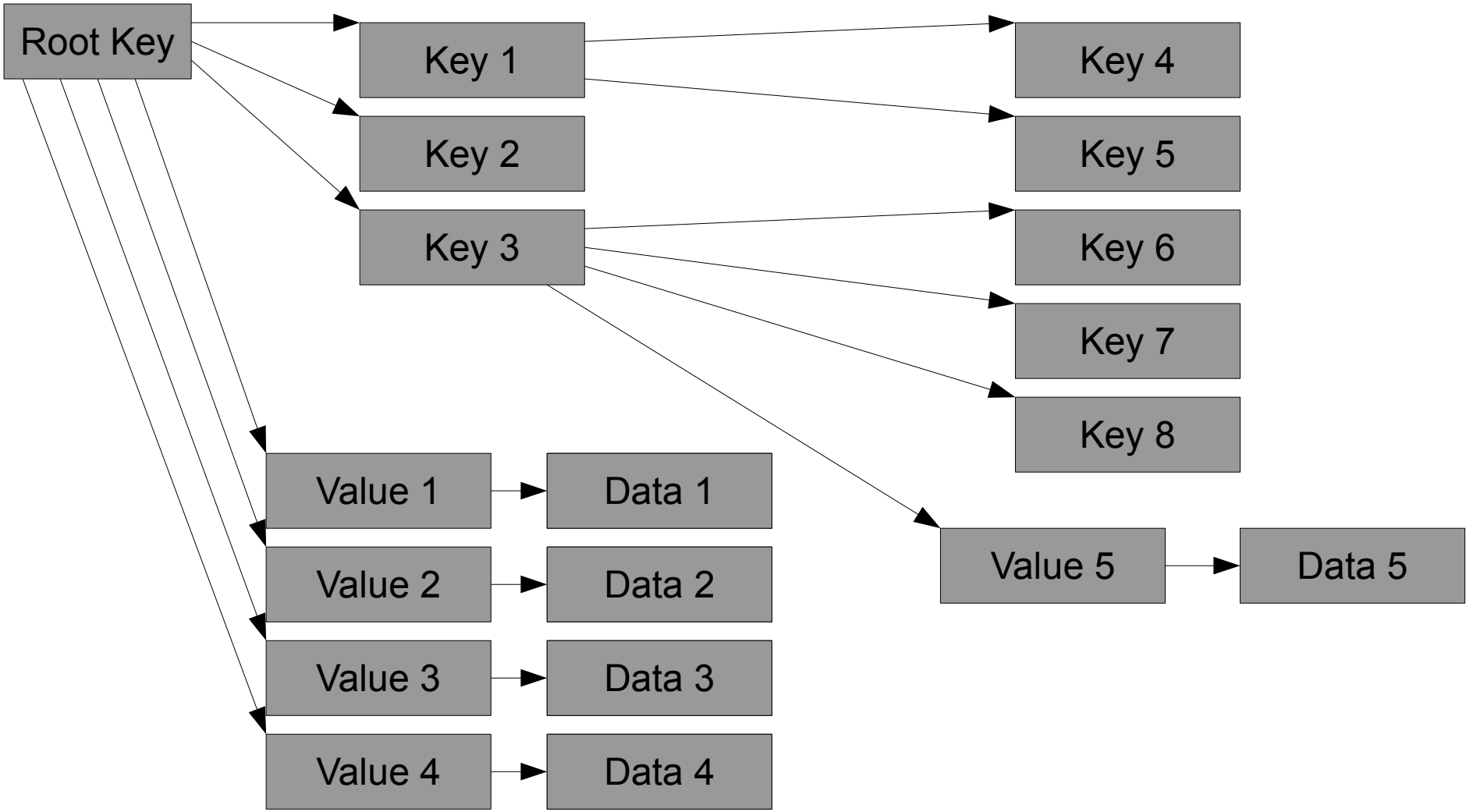
- Registry Overview
- Deletion Behaviors
- Recovery Algorithm
- Experimental Results
- Antiforensics & Future Work

Registry Overview

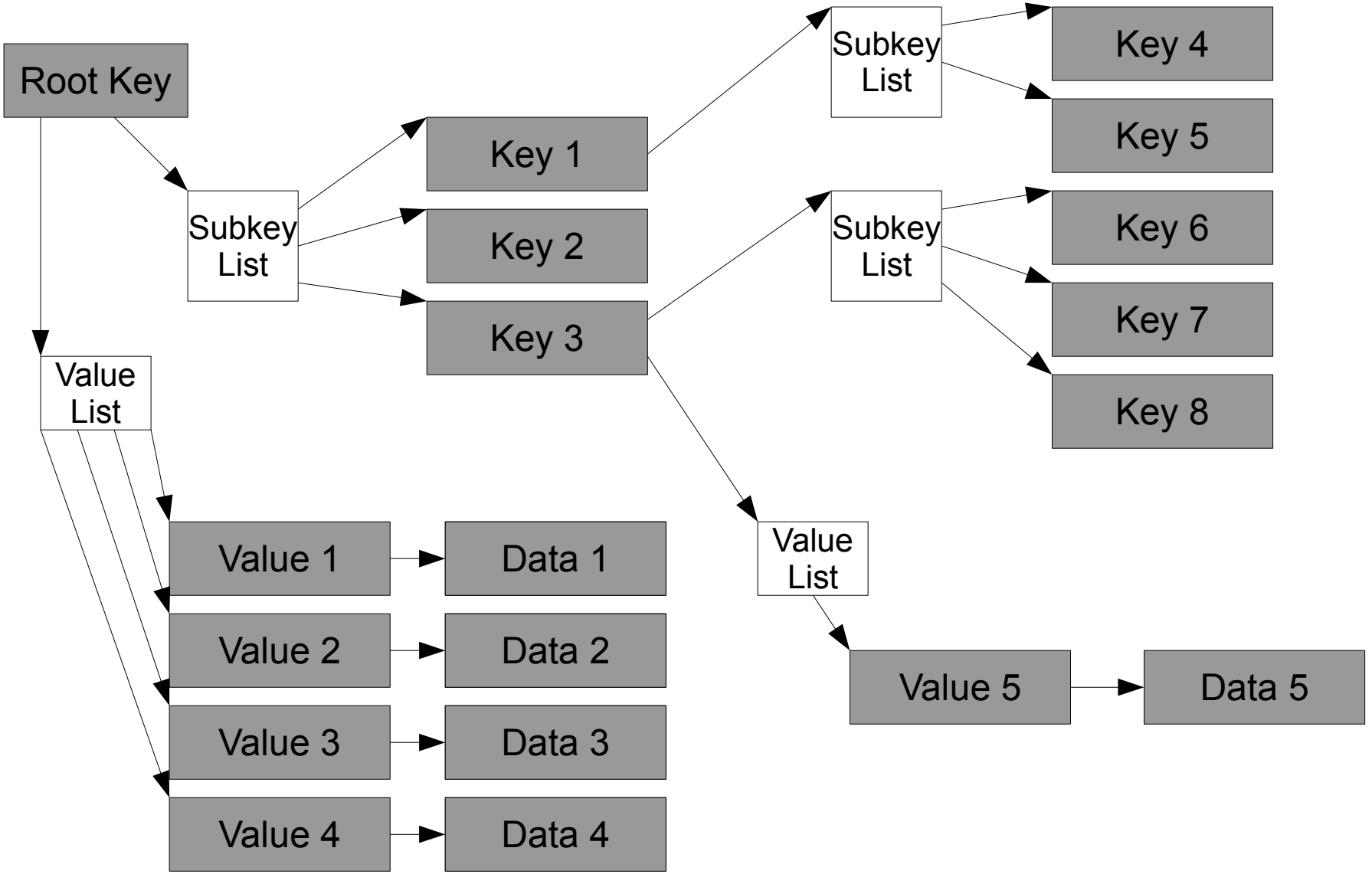
- The Windows registry is organized much like a filesystem.
  - Keys are analogous to directories, values analogous to files, data analogous to file contents
  - Keys have associated ownership and permissions
  - Registry "hives" are mounted under virtual paths much like filesystems are in UNIX[1]
- Some differences include:
  - Keys reference subkeys differently than values
  - Values store data type information
  - Records are small and are allocated differently to better work within a compact, variable-length file

1. UNIX is a registered trademark of the Open Group.

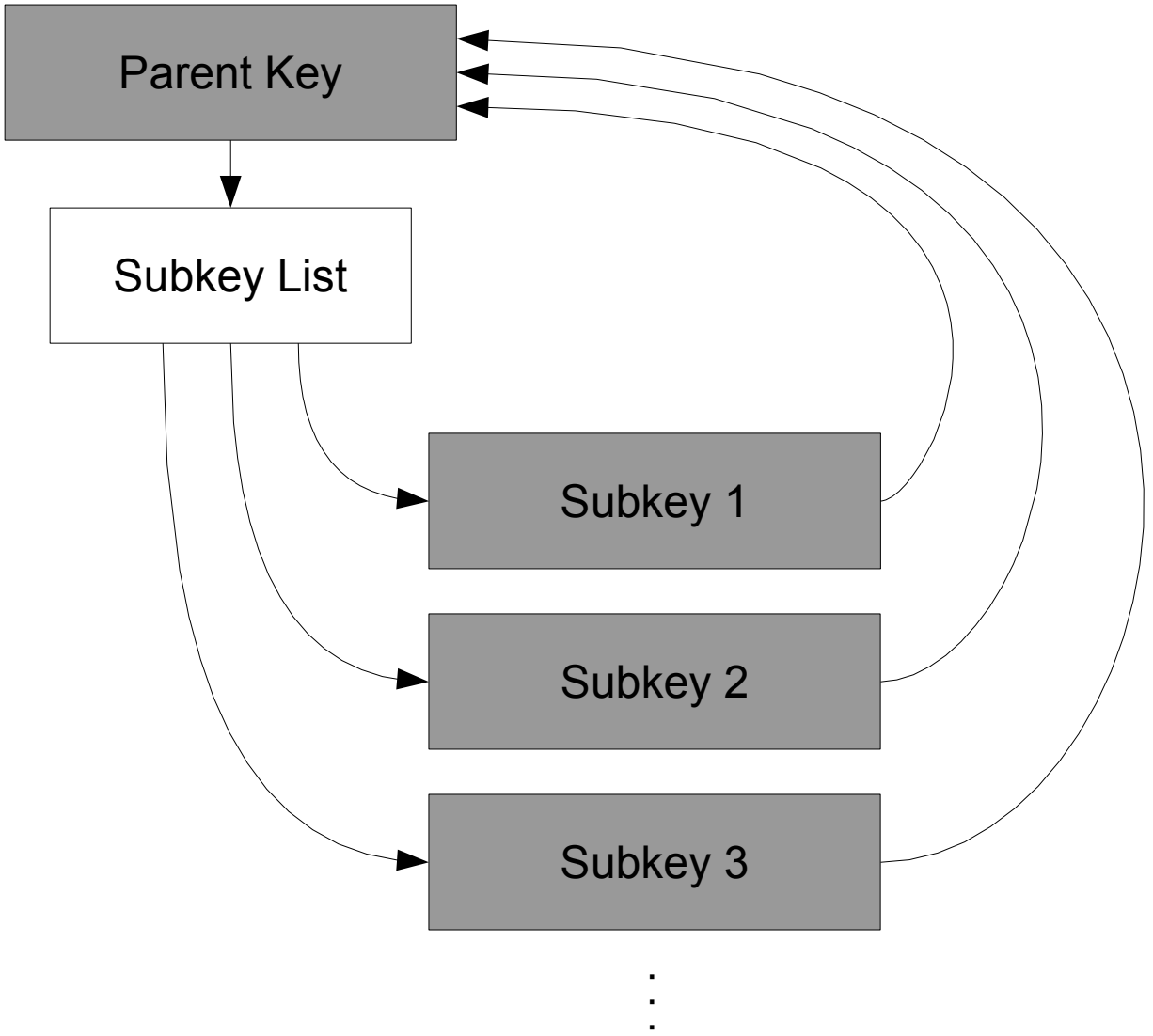
Hypothetical Hive Logical Diagram



Hypothetical Hive Data Structure Diagram



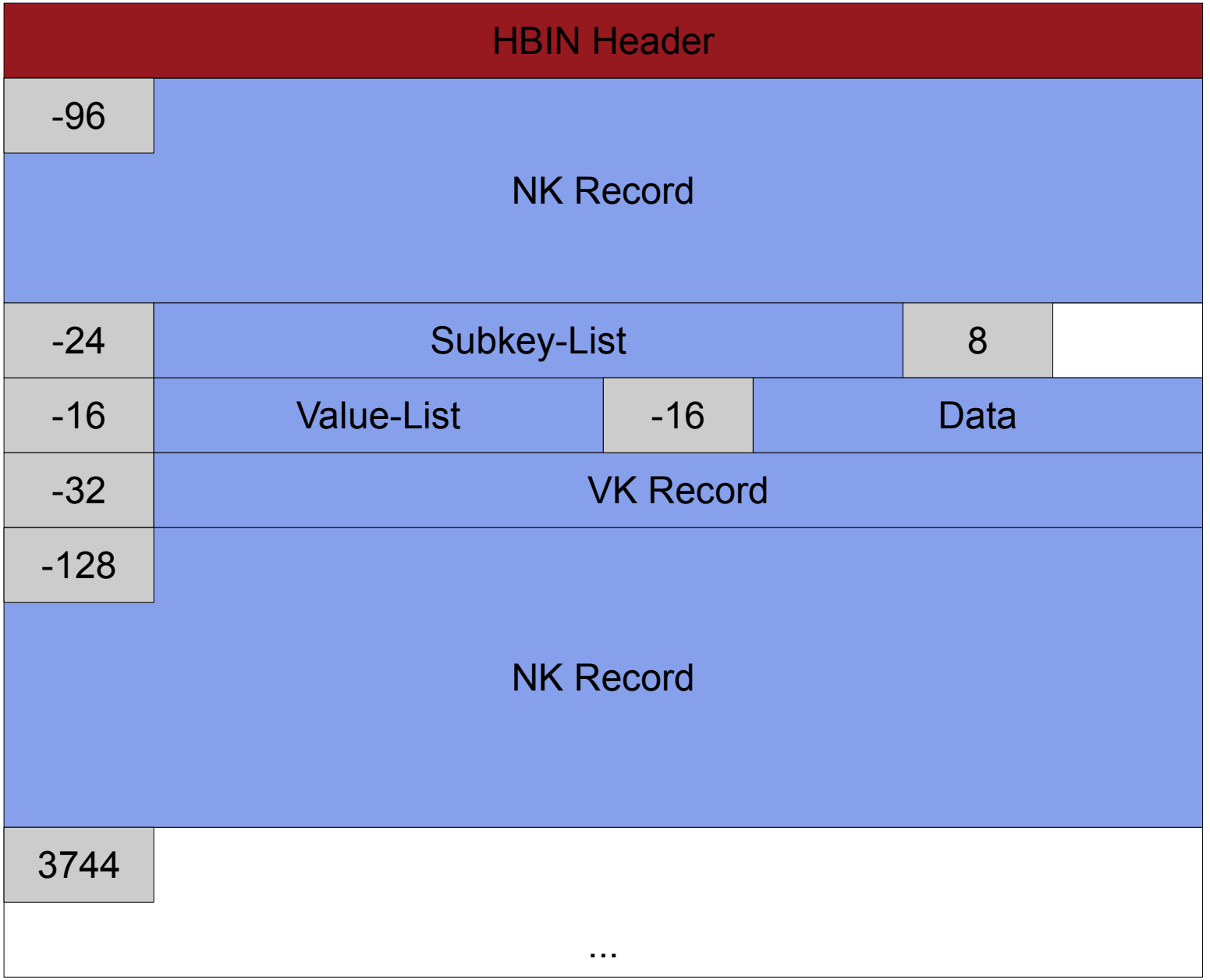
Data Structures
Redundant Parent Link



Record Storage: Hive Bins

- All physical data structures are stored in a hive inside blocks called "hive bins" (HBINs)
- Each data record is stored in cells of variable length
- Cells are organized in HBINs using simple length-prefix notation with a 32-bit signed length
  - A negative length value indicates a cell is currently occupied
  - Positive values indicate the cell is available for use

Hypothetical Hive Bin Diagram



Summary of Record Types

- Keys
  - "nk" signature, points to parent key & child keys/values
- Values
  - "vk" signature, contains type and pointer to data
- Subkey lists
  - Three types ("lf", "lh", and "ri") with header and 8-byte elements which contain a hash and a pointer
- Value lists
  - No signature, simple list of pointers to value records
- Security records
  - "sk" signature, contains Windows security descriptor
- Data
  - No signature, variable length raw value data

Deletion Behaviors

- When a cell is freed:
  - Any adjacent unallocated cells are merged; or
  - The cell's size is changed to a positive integer
  - Cell data is generally left intact

- In analyzing records, we must consider a number of aspects:
  - Changes to each record type when deleted
  - Effect of sequential deletions on subkey/value lists
  - Recursive deletion behaviors
  - Modification time changes
  - Platform-specific issues (i.e. Windows 2000)

Deletion Behaviors: Key Records

- Key records are important, since they tie most of the data structures together.

- The bad news:
  - The pointers to the subkey list and security record are destroyed (overwritten with 0xFFFFFFFF).
  - The number of subkeys is overwritten with 0.

- The good news:
  - The parent key and value list pointers are left intact.
  - The number of values is left intact.

Deletion Behaviors: Lists

- For subkey lists and value lists, there are two cases to consider:
  - Some or all listed elements are deleted
  - The parent key is deleted
- When elements of a list are deleted for either list type:
  - The elements are removed individually and the list is rewritten each time. Slack space is not wiped.
  - For both list types, if the final element is removed, the list is deallocated.
- When a parent key is deleted:
  - Subkey lists are corrupted by sequential deletions.
  - Value lists are left intact.

Deletion Behaviors: List Diagram

Original List:

After Deleting G:

After Deleting B:

After Deleting F:

After Deleting A:

After Deleting D:

Deletion Behaviors: Value & Data

- For the most part, these records are untouched when deleted.
- The only exception is under Windows 2000 where:
  - *Sometimes* the first four bytes of value records are overwritten with 0xFFFFFFFF. This corresponds to the magic number/signature and value name length fields.
  - *Sometimes* the first four bytes of data content is overwritten with 0xFFFFFFFF.
- Since this is specific to Windows 2000 and is inconsistent, it might just be a bug.

Deletion Behaviors: MTIMEs

- Recall that modification times are only stored on keys.
- This timestamp is updated when the key is deleted **and** the key has one or more subkeys.
- The only known exception is under Windows 2000 where the MTIME is not updated on a key even if it has subkeys.

Recovery Challenges

- In the best case, we have just enough information to reconstruct data structures.
- Unfortunately, we're commonly faced with broken links at multiple levels.
- When cells are merged, we lose track of where deleted records begin.
- Validation of structures is difficult. Old data could be confused with old records.

Recovery Approach

- Maximize trustworthiness and volume of data recovered by:
  - Developing detailed and precise validation methods
  - Making few assumptions about where deleted cells begin
  - Making conservative assumptions about where cells end
- Be greedy. Start with:
  - The most trustworthy data structures
  - Data structures which have the most context

Recovery Approach (cont.)

- A rough outline of the proposed algorithm is as follows:

```
Scan all unallocated cells for NK records.
  For each of these NK records, follow parent
    links up as far as possible, recording the
    path.

  For each NK record, recover as many VK and
    data records as possible.

Scan remaining unallocated cells for VK records.
  For each remaining VK record, recover any
    associated data records

Scan remaining unallocated cells for SK records.
```

Experimental Results

- The algorithm was implemented in a command line tool, reglookup-recover, as component of RegLookup.

- Using this tool, tests were conducted on both production and test environment installations to determine the:
  - Efficacy of the proposed algorithm
  - Amount of deleted data available to examiners

- Exploration of these topics was conducted on several Windows releases, including: 2000, 2003 SE/EE, XP, and Vista

Experimental Results (cont.)

- The amount of data recovered from registries varies widely.
  - Level of fragmentation is a key factor. Allocation strategies likely focus on minimizing this.
  - Some hives change seldom, others often.

- From a small sampling of hives:
  - Total unallocated space varies from 0% to 30%.
  - Percentage of unallocated space which is recoverable has been anywhere between 0% and 83%.

Experimental Results
Vista Case Study

- The reglookup-recover tool performed well against the system registry hive from a Vista test system:
  - The system was minimally used, though several third-party software packages had been installed.
  - 30% of the registry was unallocated. 83% of this unallocated space was recovered as deleted structures.
  - 5669 keys were recovered, 4101 of which could be associated with a parent key.
  - 16748 values were recovered, 6470 of which could be associated with a key.

Vista Case Study (cont.)

- Some of the more interesting keys recovered included:
  - 603 keys and values related to firewall policies
  - 123 keys and values under the new Vista registry virtualization area
  - Numerous different hardware configurations, including USB settings
  - 444 keys/values related to event log configurations
  - Miscellaneous LSA, group policy, and terminal services settings

Antiforensics

- While the algorithm proposed does well in recovering structures from typical registries, malicious registries are another story.
- The lack of authoritative parent-to-child links means that data structures may be spoofed by low-privileged users.
- Data may be easily hidden in the registry by simply setting unallocated cells to an allocated state.

Antiforensics (cont.)

- Often the best antiforensic techniques are the simplest.
- Numerous registry defragmentation tools exist which likely eliminate most deleted cells.
  - Plausible deniability of "performance tools"
- Creating a single value under every key and then deleting them would probably overwrite most recoverable records.
  - This has the benefit of changing MTIMEs as well.

Future Work

- Different tool behaviors
  - Even regedit.exe and reg.exe differ
  - Third-party tools
  - Antiforensics tools
- Deletion statistics
  - How fragmented do registries become over time?
  - How long do deleted cells typically survive based on cell size?
  - What is the allocation strategy and does this differ between Windows versions?

References

- These presentation slides, the paper, and additional information

  http://sentinelchicken.com/research/registry_recovery/

- First Looks: Basic Investigations of Windows Vista

  http://www.lancemueller.com/vistaceic2007.ppt

- RegLookup

  http://projects.sentinelchicken.org/reglookup/

- The Windows NT Registry File Format

  http://sentinelchicken.com/research/registry_format/