2009

# A Framework for data decay in client-server model

Matthew Taber

# A Framework for Data Decay in Client-Server Model

By

**Matthew Taber**

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Security and Information Assurance

Supervised by

Dr. Charles Border
Department of Networking, Security, and Systems Administration
B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY
August 2009

**Approved By:**

_____

Dr. Charles Border
*Primary Advisor – R.I.T. Dept. of Networking, Security, and Systems Administration*

_____

Dr. Peter Lutz
*Secondary Advisor – R.I.T. Dept. of Networking, Security, and Systems Administration*

_____

Dr. Bo Yuan
*Secondary Advisor – R.I.T. Dept. of Networking, Security, and Systems Administration*

# Thesis Release Permission Form

Rochester Institute of Technology
B. Thomas Golisano College of Computing and Information Sciences

**Title: A Framework for Data Decay in Client-Server Model**

I, Matthew Taber, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

_____
Matthew Taber

_____
Date

# Dedication

This paper is dedicated to my father, Allen Taber. Without his help and guidance, the report that you are about to read would not be what it is today. I would also like to dedicate this paper to my mother, Bonnie Taber, and sister, Sarah Taber, who continually supported me throughout the writing process.

# Acknowledgements

I would like to thank my committee advisor, Charles Border, and my committee members, Peter Lutz and Bo Yuan, for their assistance and guidance throughout the thesis process.

# Abstract

Data breach incidents are a growing concern in both the private and public sector. As an increasing amount of sensitive information is made available through information systems, the number of occurrences and the potential damage as a result of these breaches will continue to expand. There is a clear need for a framework to mitigate the dangers of leaking data while simultaneously allowing authorized individuals to gain access to necessary information. This thesis presents the concept of *data decay*, applied to a client-server model for retrieval of sensitive data. In this framework, files tagged by the framework and stored on client machines undergo a process in which they become increasingly non-existent to the client in proportion to the time elapsed since the last server connection.

In order for a client to continue using a file that has undergone the decay process, it must successfully authenticate to the central server to begin rebuilding the file. The time required to rebuild the file is computed by the length of time since the last client connection to the server, in addition to the level of sensitivity for the specific file (files tagged as highly sensitive have a faster decay rate than those tagged as moderately sensitive). Using this model, an adversary's window of opportunity to view sensitive information diminishes as the client remains disconnected from the central server. A regular user will find this framework a benefit in the ability to work remotely and have access to required sensitive information without the risk of unintentionally leaking it to unintended parties

# Table of Contents

Thesis Release Permission Form ..... ii

Dedication ..... iii

Acknowledgements ..... iv

Table of Contents ..... vi

List of Figures ..... xi

List of Tables ..... xii

Glossary ..... xiii

Chapter 1 Introduction ..... 1

1.1. Problem Summary ..... 3

1.1.1 Case Example ..... 3

1.2. Importance ..... 4

1.3. Objective ..... 4

1.4. Document Outline ..... 5

1.5. Introduction to Data Decay ..... 5

1.6. Review of Literature ..... 6

1.6.1 Growth of Data ..... 6

1.6.2 Short-Lived vs. Long-Lived Data ..... 6

1.6.3 Central vs. Distributed Storage ..... 9

1.6.4 Access Control, Confidentiality, and Integrity ..... 12

Chapter 2 Methodology ..... 14

2.1. Development of Key Concepts ..... 14
2.2. Development of Decay Framework ..... 14
Chapter 3 Results and Analyses ..... 16
3.1. Key Concepts ..... 16
3.1.1 Value of Data to a User ..... 16
3.1.1.1 Regulatory ..... 17
3.1.1.2 Productivity ..... 17
3.1.1.3 Personal ..... 18
3.1.1.4 Value Mix ..... 18
3.1.1.5 Value Known in Advance ..... 19
3.1.1.5.1 Rapidly Decreasing ..... 19
3.1.1.5.2 Slowly Decreasing ..... 20
3.1.1.5.3 Increasing ..... 20
3.1.1.6 Value Not Known in Advance ..... 20
3.1.1.7 Frequency of use ..... 21
3.1.2 Value of Data to an Adversary ..... 21
3.1.2.1 Loss of Data Confidentiality ..... 21
3.1.2.2 Loss of Data Integrity ..... 22
3.1.2.3 Loss of Non-Repudiation ..... 23
3.1.3 Data Pollution ..... 23
3.1.3.1 Corollaries to Environmental Pollution ..... 24
3.1.3.2 Causes and Cleanup ..... 25
3.1.3.3 Prevention ..... 26

3.1.4 Half-Life ..... 27
3.2. Potential Decay Frameworks ..... 27
3.2.1 File Corruption ..... 28
3.2.1.1 Break into increasing pieces ..... 28
3.2.1.2 Overwrite ..... 28
3.2.1.3 Encrypt/alter increasing segments ..... 29
3.2.1.4 Reduce Redundancy ..... 29
3.2.2 Key Corruption ..... 30
3.2.2.1 Break into increasing pieces ..... 31
3.2.2.2 Overwrite ..... 31
3.2.2.3 Encrypt/alter increasing segments ..... 32
3.3. Time Constants for Decay ..... 32
3.3.1 Grace Period Before Decay Starts ..... 32
3.3.2 Maximum Decay Rate ..... 33
3.3.3 Minimum Decay Rate ..... 33
3.3.4 Example Decay Formula ..... 33
3.3.5 Time Constant Override ..... 37
3.3.6 Data Recovery Example ..... 38
3.4. Determining Elapsed Time ..... 39
3.4.1 BIOS clock ..... 39
3.4.2 OS clock ..... 39
3.4.3 Network time server ..... 40
3.4.4 Anti-spoofing ..... 40

3.4.5 Correlation from Multiple Sources ..... 40
3.5. Security Drawbacks ..... 41
3.5.1 Inference Attacks ..... 42
3.5.2 Eavesdropping ..... 42
3.5.3 User Non-Compliance ..... 43
3.5.4 Man-in-the-Middle Attacks ..... 43
3.6. Usage Drawbacks ..... 44
3.6.1 Unintentional Loss ..... 44
3.6.2 Server Availability ..... 45
3.6.3 Time to Restore ..... 45
3.7. Comparison with Existing Methods ..... 46
3.7.1 Software Methods ..... 46
3.7.2 Manual Methods ..... 47
3.7.3 Hardware Methods ..... 47
3.7.3.1 Volatile Memory ..... 47
3.7.3.2 Ionizing Radiation ..... 48
3.7.3.3 No Local Memory ..... 49
3.7.3.4 Hardware Encryption ..... 49
3.7.4 Comparison Summary ..... 49
3.7.5 Anticipated User Reaction ..... 50
3.7.6 Future User Needs ..... 50
Chapter 4 Summary ..... 52
4.1. Conclusions ..... 52

4.2. Recommendations ..... 53

Chapter 5 Future Work ..... 54

5.1. Refined Data Decay Time Constants ..... 54

5.2. Improved Hardware Methods ..... 55

5.3. Impact of increasing distributed computing ..... 55

5.4. Spin-offs From Neurological Studies ..... 56

5.5. Artificial Intelligence ..... 56

References ..... 57

# List of Figures


1 Value of Data to User…….....16

2 Data Decay Plot……..…….……..…35

3 Data Decay Scenarios…….…….....37

4 Data Recovery Plot…….....38

# List of Tables

1 Corollaries to Environmental Pollution. ……..……25

2 Data Decay Scenarios……..36

# Glossary

**BIOS** Basic Input/Output System

**Client** A computer in a network that uses the services (as access to files or shared peripherals) provided by a server [MW]

**CPU** A Central processing unit (CPU) or processor is an electronic circuit that can execute computer programs [WP].

**Data** Information in numerical form that can be digitally transmitted or processed [MW]

**Data Breach** The unintentional release of secure information to an insecure environment [WP].

**DRAM** Dynamic Random Access Memory. A memory storage device that maintains data integrity while powered on, as long as the capacitive memory cells are periodically refreshed. If powered off, the memory is volatile and data is lost.

**Encryption** The process of transforming information (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key [WP]

**Half-Life** The half-life of a quantity whose value decreases with time is the interval required for the quantity to decay to half of its initial value [WP]

**Hash Function** A cryptographic hash function is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an accidental or intentional change to the data will change the hash value [WP]

**I/O** Input/output, or I/O, refers to the communication between an information processing system (such as a computer), and the outside world – possibly a human, or another information processing system. Inputs are the signals or data received by the system, and outputs are the signals or data sent from it [WP].

**HIPPA** The Health Insurance Portability and Accountability Act (HIPAA) regulates the use and disclosure of an individual's medical record or payment history [WP].

**Key** A “key” is a piece of information (a parameter) that determines the functional output of a cryptographic algorithm or cipher [WP]

**MITM** In cryptography, the man-in-the-middle attack (MITM) is a form of active eavesdropping in which the attacker makes independent connections with the victims and relays messages between them, making them believe that they are talking directly to each other over a private connection when, in fact, the entire conversation is controlled by the attacker [WP].

**Network** A computer “network” is a group of interconnected computers [WP]

**Network Time Protocol (NTP)** A protocol for synchronizing the clocks of computer systems over packet-switched, variable-latency data networks. In the Internet, NTP synchronizes computer system clocks to UTC [WP].

**Non-Repudiation** A service that provides proof of the integrity and origin of data or an authentication that, with high assurance, can be asserted to be genuine [WP].

**OS** An operating system (commonly abbreviated to either OS or O/S) is an interface between hardware and user; an OS is responsible for the management and coordination of activities and the sharing of the resources of the computer [WP].

**Redundancy** Data redundancy is the storage of additional information (entire data set or error-correcting codes) to provide fault tolerance, so that all or part of the data can be recovered in the case of disk failure or other accidental loss [WP]

**Signature** For messages sent through an insecure channel, a properly implemented digital signature gives the receiver reason to believe the message was sent by the claimed sender [WP].

**Server** Any combination of hardware or software designed to provide services to clients [WP]

**Spoof** A spoofing attack is a situation in which one person or program successfully masquerades as another by falsifying data and thereby gaining an illegitimate advantage [WP].

**SRAM** Static Random Access Memory. A memory storage device that maintains data integrity while powered on, with no requirement for refresh (hence the name “static”). If powered off, with no battery backup, the memory is volatile and the data is lost.

**Volatile Memory** Volatile memory, also known as volatile storage, is computer memory that requires power to maintain the stored information, unlike non-volatile memory which does not require a maintained power supply [WP].

**UNIX** A computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs [WP].

**UTC** Coordinated Universal Time is a time standard based on International Atomic Time (TAI). UTC is the time system used for many Internet and World Wide Web standards. In particular, the Network Time Protocol, which is designed to synchronize the clocks of many computers over the Internet, uses UTC [WP].

# Chapter 1 Introduction

There is a clear threat to enterprise and government entities that exists in the form of data breach incidents. Loss of sensitive information due to theft or misplacement can have severe consequences, which include identity theft, financial loss, and even compromise to national security.

An organization known as the Privacy Rights Clearinghouse maintains a list of recorded data breach incidents since January 2005. For only three years worth of collections, there are already an astounding number of data breaches in the United States. A figure of 245,130,070 records involving sensitive information has been included in reported data breaches since the Privacy Rights Clearinghouse first began reporting incidents [privacyrights.org].

Several independent researchers have compiled raw data from the Privacy Rights Clearinghouse records to provide a clear understanding of what and who were affected, as well as why. Beth Rosenberg provides a picture of events that occurred in 2006, categorized by public, private, educational, and medical organizations that experienced data breaches. There were a total of 126 data breach incidents in the private sector, of which 40% consisted of laptop theft. Public sector incidents saw a figure of 114 reported cases with 44% of the issues arising from human incompetence and software errors. There were 52 reported incidents in the education sector, with 52% of the incidents occurring by way of outside hackers. Last, but not least, medical centers had the fewest incidents at 30 reported cases, where laptop theft was the chief cause in 40% of these cases [Rosenberg].

Based on these statistics, there is a definitive need to address what can be done to reduce and ultimately eliminate data breach incidents. Therefore, I have proposed the development of a client-server model that employs a novel approach to the manner in which data is stored by the client devices. The method introduced is best described as a form of *data decay*, in which files uploaded to the client from the server have a rate of expiration or degradation that corresponds to their level of sensitivity. I refer to a client-server model, as this is the ideal transaction environment to integrate the concept of decaying data. A client authenticates with a central server that stores original copies of sensitive information. Upon successful authentication, a client can gain access to these sensitive files to conduct work from a device such as a laptop. However, once the client ends its session with the server, a data degradation process initiates wherein the files retrieved from the server will begin corruption and/or erasure until they reach the point at which they are completely useless.

In order to protect against an attacker spoofing elapsed time to avoid data decay, the model not only relies on system time, but also on time indicators from multiple sources. The decay process is what I like to refer to as the "Rip Van Winkle effect," where the longer that a client is detached from the central server, the greater the time that is required to repair the file for use. The mechanism that governs the data decay operation sits in the system or kernel level of an operating system, in order to prevent an attacker from potentially deactivating it. Additionally, for a client to obtain files tagged by the server to decay, the server checks the client to make sure that they are running the complimentary software in which to do so.

## 1.1. Problem Summary

Current security controls are not effective in solving the increasing frequency of data breach incidents on the rapidly growing amount of data being used and stored. These breaches often contain sensitive information that can cause damage to the reputation of the victim organization and result in harm to individuals. It is imperative that we look at new ways of safeguarding data stored on client platforms.

### 1.1.1 Case Example

In a 2009 report published by PGP Corporation and the Ponemon Institute, information was collected on the cost of data breach incidents in the US [PGP]. The study recorded information from 43 different organizations in 17 different industry sectors including commercial, healthcare, education, finance, and defense. It was determined that it cost an organization $202 per compromised customer record, with an average data breach incident cost at $6.65 million. This cost has risen by 40% since 2005 when data breach studies began. Of the incidents reported, it was found that 88% of them had involved some degree of insider negligence, and that only 44% of organizations had implemented use of encryption after the incident to safeguard against future issues.

The information in this recent report provides a sobering look at the cost of data breach incidents, the majority of them being the direct result of user non-compliance to security regulations. An important aspect of the data decay framework is that it proactively defends against the release of sensitive information in the event that a user loses or misplaces the client machine. The removal of access to sensitive information through its

decay creates a challenge for the adversary in order to leverage possession of the information against the owner of the data.

## 1.2. Importance

As the amount of stored information grows, the risk that sensitive information will be disclosed to unintended parties follows suit. This thesis aims to address the significant issue in the near future of data clutter and question our current belief in the collection and storage of any and all information produced by modern society. As computer security expert Bruce Schneier has discussed in past dialogues regarding the growth of information, he strongly believes that we face a *data pollution* problem that will become as important an issue as environmental pollution from industrialization [Schneier]. In many ways, how we address the data pollution issue will mirror current tactics to repairing the effects of environmental pollutants, and these methods will be described in detail later on in the paper.

## 1.3. Objective

The objective of this thesis is to present a suitable framework for implementing data decay in a client-server model. The framework will define what is necessary to carry out the tenants of data decay and address the potential concerns associated with its implementation. In addition, the thesis posits new concepts such as considering the "value of data to an adversary" and the "ability to protect data", in determining the rate at which this data will undergo the decay process. Another important aspect is the introduction of the concept of *gradually* corrupting a file instead of abruptly and completely removing or encrypting it, in order to make it less available to an adversary.

## 1.4. Document Outline

The paper is outlined as follows: In the first section, basic definitions are covered such as data decay, the notion of data pollution, and the half-life calculation for "decaying" files. Following this is a discussion on the value of data to users and adversaries, and the determination of whether it is increasing or decreasing in value. The next section will cover potential strategies for performing data decay, including formulas to demonstrate the factors that determine the rate at which the operation occurs. This is followed by an examination of determining with some accuracy the passage of time in order to defeat potential spoofing attacks against the framework. Results and conclusions are presented, with a discussion of future research objectives.

## 1.5. Introduction to Data Decay

Data decay is defined as the gradual degradation of a file over time. This degradation can be both intentionally and unintentionally triggered. With this definition in mind, we examine how to meet the goals of data decay through the utilization of intentional methods for introducing degradation in stored data. Additionally, we will examine methods for *undoing* the effects of degradation to a file in order to restore it to its original state.

## 1.6. Review of Literature

### 1.6.1 Growth of Data

We are faced with the issue increased data growth in the future, a problem that may pose as serious a threat to the information world as environmental pollution does today. Security professional Bruce Schneier discusses the importance of developing future information systems that deal with the growing issue of "data pollution." In an excerpt from his interview at the 2007 EDUCASE conference, Schneier compares the current information revolution to its predecessor, the industrial revolution, in the way that data pollution is the same as the pollution by-product of industrialization [Pasiewicz]. Computers are constantly producing new data and storing it indefinitely, and Schneier argues that in the coming decades we will be faced with a tumult of data clutter that will require cleanup in a manner similar to our response towards cleaning up industrial pollutants. This concept will be elaborated later on in the thesis, as the corollary between environmental and data pollution is examined.

### 1.6.2 Short-Lived vs. Long-Lived Data

There are two types of data that generally exist on storage systems, categorized by the length of time that the data is considered of value to the user. Data can have a long lifespan of usefulness, or a short period of time in which it is utilized. The notion of storing both short-lived and long-lived data is discussed in "A Framework for Evaluating Storage System Security" [Riedel]. The focus of the paper is to quantify the effectiveness of various secure storage systems. The authors of the paper seek to establish metrics to rate the effectiveness of existing controls, in order to better understand what dictates secure

storage within a unified framework. The two major forms of secure storage systems defined by the authors are those that encrypt data while in transit and those that encrypt data while it is stored on disk. At the beginning of the study it is asserted that encryption performed on data while it is stored on disk is superior to encryption while in transit [Riedel].

The security metrics used by Reidel et al involve a calculation of the security, performance, and convenience of use for a given secure storage solution. Emphasis is placed on the importance of convenience, as unhappy users of a secure storage system will ultimately find ways to circumvent the protection scheme. The factors for analyzing a secure storage system involve the players, attacks, security primitives, granularity of protection, and user inconvenience. The term *players* is further described as consisting of the data owners/readers/writers, transmission wires, and storage/group/namespace servers. Anything that is outside the permitted roles of these players is considered to be an *attack*, and any entity not described in the definition of a player is classified as an *adversary* [Riedel].

As mentioned earlier, two categories of data handled by secure storage systems are defined as *short-lived* and *long-lived* data. It is stated that current technologies for implementing network security have addressed the issue of securing short-lived data through the use of encrypted communications protocols. However, it is often the long-lived data stored on servers that has a large cause for concern. The lifespan of information is correlated to the probability of its eventual disclosure, as the longer data persists, the higher chance it has of being compromised [Riedel]. In this thesis it is proposed that data should have a restricted lifespan when stored in locations that cannot guarantee its protection

(client machines outside of the secure storage server's defenses), which falls in line with this correlation drawn by Reidel et al.

Reidel et al also state three categories of attacks: leak, change, and destroy. A *leak* attack occurs when an adversary gains access to some data. *Change* attacks occur when an adversary makes valid modifications to the data, which are not detected by users or the system as invalid. Lastly, *destroy* attacks are similar to change attacks. However the users or the system can detect that an invalid modification has occurred [Riedel]. Security primitives are defined to include authentication, authorization, and securing data on disk, securing data on the wire, key distribution, and revocation. These primitives are explained in detail, including the use of symmetric/asymmetric encryption to meet the goal of securing data on disk. An interesting notion is presented in that the data may be stored on an untrustworthy server or reside on a stolen laptop, thus it is imperative to obfuscate the data via encryption to avoid disclosure [Riedel]. In my proposed data decay framework, encryption plays an important part in ensuring that a data breach does not occur when an adversary has direct physical access to the storage disk of a seized asset and attempts to extract its contents.

By the end of the paper, Reidel et al have developed a process in which to analyze secure data storage systems and evaluate their merits. They find that there is less overhead and increased ease of use in systems that encrypt data stored at the server, as opposed to in-transit between the endpoints. There is, additionally, a greater probability that the data will not be compromised due to unintentional disclosure on an untrustworthy server storage point. This paper provides a useful frame of reference in the design and implementation of

the data decay framework, illustrating what must go into the design of secure storage systems and how to effectively rate their level of protection and usability. The authors themselves plan to design a secure data storage system based on their framework as they look ahead at continued research.

### 1.6.3 Central vs. Distributed Storage

When storing large amounts of data, one should consider whether it is better to keep the information in a central location, or take advantage of distributed storage technologies. In "Your Place or Mine?: Privacy Concerns and Solutions for Server and Client-Side Storage of Personal Information," the quandary of storing sensitive information in an organization is discussed [Mulligan-Schwartz]. The authors raise concerns about storage of data on a central server. Having all information stored at a single point makes that server a premiere target for intruders, or allows easy collection of sensitive records by government/law enforcement entities that request access. The solution suggested to mitigate these risks comes in the form of encrypting the entirety of data stored on the server and entrusting individual users with the ability to authenticate and decrypt necessary items. In this manner, liability is transferred from the company to the individuals, and even if access to the data is provided, it is virtually useless without the proper decryption scheme [Mulligan-Schwartz].

Another form in which data can be stored is via a client-side approach. Here the authors highlight that the distribution of sensitive data to multiple storage points lessens the attractiveness to acquire it by outside entities, in comparison to a single storage location. However, overall access to files is diminished, as the systems that house files must be active and participating in the appropriate network in order for retrieval to be a success. There is

also the chance that a client can lose possession of their system and expose their portion of sensitive information to a non-trusted party [Mulligan-Schwartz].

This is the concern that is highlighted by this thesis: clients should have access to sensitive information when required, but mechanisms to prevent potential loss should be in place. As learned in the server-side storage example, the data decay client-server model should ensure that the central database is encrypted, in addition to the user authentication phase.

In "Survivable Information Storage Systems," the authors present a secure storage architecture called PASIS and discuss the importance of designing secure systems to meet the demands of confidentiality, integrity, and availability [Wylie]. The goal of the authors is to present a robust system for handling critical information storage that is persistent and available, cannot be destroyed, and remains confidential to thwart unauthorized access. Additionally, the system should be designed to mitigate the effects of component failure and unexpected loss of data. The authors present PASIS and tout its decentralized nature and redundancy as a way to protect against these issues.

The chief principle in a survivable storage system is that no service, node, or person can be fully trusted. In order to reduce the level of risk associated with trusting a particular node, an ideal system is designed to be decentralized and incorporate data monitoring and maintenance components. Traditional storage systems make use of one or more servers at most, and do not truly offer the advantage of a decentralized system, according to Wylie et al. In the ideal survivable system, a multitude of independent nodes are utilized while still appearing as one logical unit to the client. Additionally, there is an issue with current

decentralized storage systems in that they are far more difficult to secure against malicious activities in comparison to a single storage node. The difficulty arises from the inability of the system to distinguish a real user from an adversary, thus it treats both requests the same.

To remediate the issues found in both central and decentralized storage solutions, Wylie et al present the PASIS architecture and discuss its merits of use. PASIS takes advantage of a client-side management system to facilitate the features lacking in existing decentralized systems. The clients communicate with the storage nodes by way of *threshold* schemes. In the threshold scheme is information used to reconstruct data by querying nodes that hold components necessary to form the complete data entity. This is a novel approach that ensures that each node stores only incomplete versions of the actual data, to avoid disclosure. This is a concept that is looked at in this thesis, as the decay operation would essentially break up the sensitive data into an incomplete form as time progresses, and would then have to be reconstructed upon the next successful connection to the server. Additionally, the notion of a decentralized storage solution that exists as a single logical entity to the end-user is an interesting concept that may be looked at for inclusion into the data decay framework, or future revisions of the system.

Wylie et al present experimental data as they compare the PASIS architecture to conventional systems of similar intent. Due to the decentralized nature of PASIS and the use of threshold schemes, it reduces the probability of loss of confidentiality from 20 to 0 percent if one node is compromised. If three nodes are compromised, the probability moves up to a mere 4.4 percent, in comparison to 60 percent disclosure probability in conventional

systems. The integrity of PASIS is equally impressive, as it takes more than 4 node losses in order to erase a piece of information completely.

The PASIS architecture is an excellent example of a secure storage system that utilizes decentralization, redundancy and data integrity mechanisms, and encoding of information to ensure that confidentiality/integrity/availability are upheld.

### 1.6.4 Access Control, Confidentiality, and Integrity

A secure system is one that is built in such a manner that safeguards against unauthorized access and ensures that stored information is not modified without the knowledge and consent of the owner. The importance of access control, confidentiality, and integrity are the focus of "Document Management System Security" by Birme. The author presents this topic in terms of access control, confidentiality, and integrity. A number of examples are given to define what access control is and how varying levels of sensitive information should be managed. In addition, the author examines mechanisms for authentication and describes these as: knowledge, ownership, and biology. Knowledge refers to what a user knows (i.e. authentication passwords or locations to sensitive information). Ownership is what the user possesses to authenticate, such as a physical or software key (e.g. banking customers and their use of ATM cards to gain account access). Biology refers to biometric authentication mechanisms (i.e. fingerprint, voice, photo recognition) [Birme].

Another concern that is brought up in the literature is the notion of trust. It is one thing to have access to information, but it is equally important that the authenticated parties can be trusted. Therefore, integrity checks must be established to ensure that users do not

tamper with information. One technique to verify information is to create hashes of the data to serve as a validation baseline [Birme].

In the framework which I have prescribed, integrity checks would be set in place through a robust revision control system with the capacity to audit changes to files, including the last user to touch the file and time of modification. Birme provides detail on how to effectively design and implement access control policy for the management of documents, which helps to serve as a guideline for the data decay framework operation.

With a growing concern for the confidentiality and integrity of data in a world where the volume of information is rapidly expanding, it is crucial that we begin looking at ways to control its dissemination. A framework for performing data removal and its restriction in regards to where it can be stored is paramount, as we continue to see large-scale data breaches at an increasingly frequent rate. As excellent as current data security and access control systems are, there is still the issue of human error and negligence. If a smarter system can be designed to seamlessly perform data wiping and management in a fashion transparent to the user, we can achieve the goal of securing data while reducing the impact of diminishing the user experience.

# Chapter 2 Methodology

## 2.1. Development of Key Concepts

The development and detail of key concepts in this thesis was based on research of existing literature related to the areas of study, in addition to discussions with industry and academic experts. The concept of data decay relies on many current and emerging technologies; therefore it is crucial to look closely at the successes and lessons learned from past researchers and experts who have examined these same issues of secure data storage.

Hypothesis 1: there exists a need to limit the lifetime of sensitive data that is stored on a system.

Through the use of literature searches, examining the growth of data and reported vulnerabilities, and examining corollaries to environmental pollution, it becomes possible to prove that such a need is apparent. The usefulness of literature searches relates to discovering relevant technology and processes required to effectively carry out the tenants of data decay. Justification for the use of the decay framework is found in the growth of data stored on systems and the ineffectiveness of current controls to protect against its dissemination to adversaries.

## 2.2. Development of Decay Framework

The framework for data decay was developed based on literature research and discussions with industry and academic experts. Based on the information collected from these activities, the framework was discussed and equations were designed to depict possible scenarios.

Hypothesis 2: a client-server system utilizing a rules-based scheme can effectively achieve the goals of data decay in a usable framework.

This hypothesis is verified through the development of the core framework concept, including data rules and schemes for decay/restoration activities. The use of example scenarios in addition to looking at existing methods aids in reinforcing the hypothesis that a rules-based framework is best.

# Chapter 3 Results and Analyses

## 3.1. Key Concepts

In this section, we'll discuss some key concepts, including the value of the data to a *user* versus an *adversary*, "data pollution", and "half-life."

### 3.1.1 Value of Data to a User

The value of data to a user depends on several factors, including regulatory, productivity, and personal reasons. These factors affect the length of time that information should be stored on a system, as shown in the graph below.

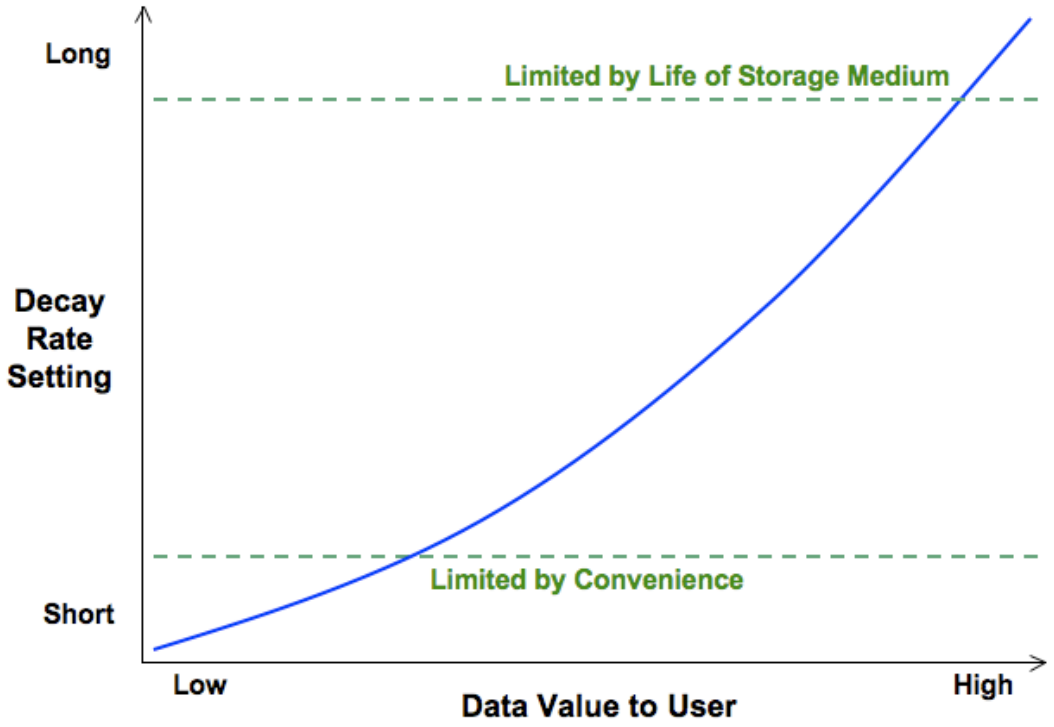


Figure 1. Value of Data to User. Depicts decay rate setting as a function of value to the user. The thresholds at top and bottom represent limitations of convenience and reliability

#### 3.1.1.1 Regulatory

In most organizations, specific data must be stored for regulatory purposes. For publicly traded companies, the Sarbanes-Oxley act (SOX) states that applicable business records must be retained in full for a period of no less than five years [SOX]. In the medical profession, regulatory acts such as the Health Insurance Portability and Accounting Act (HIPAA) dictate the methods by which sensitive patient insurance and medical records must be securely stored, yet be accessible to the parties involved [HIPAA]. In the event that the organizations bound by regulatory practices fail to preserve data, they may face serious penalties and potential criminal charges as a result.

This would limit the *maximum decay rate* so that necessary files are retained for at least the required length of time.

#### 3.1.1.2 Productivity

Stored data is often valued for its correlation to productivity, and the need to have certain information available for the purposes of conducting day-to-day business. Examples include Microsoft Word, Excel, and PowerPoint documents used by employees of an organization to carry out their duties. In most cases, this data is initially highly valued by a user, as it carries a significant weight to the success or failure of the individual in terms of their profession. It is not uncommon to see data of this type stored redundantly, having a backup copy located on a secure storage point such as a company file server. With this in mind, productivity data fits well into the scope of the data decay

framework, as a master copy of the file would be stored on a central server for employees to access when required to do so.

#### 3.1.1.3 Personal

Personal information is also highly valued by the user and may contain such information as financial/medical records, credit card/social security numbers, private communications, etc. A portion of this data consists of confidential information about the individual that he/she would not want others to have access to, unless absolutely necessary. The exposure of this information could directly harm the privacy and well being of the individual, therefore the confidentiality aspect of security is strongly in play when safeguarding this data type. The benefit of the data decay framework is that the user can be assured that copies of personal data that are floating about in storage without recent access will be removed before one forgets about their existence and risks unintentional disclosure.

#### 3.1.1.4 Value Mix

*High value* data typically represents a *small* portion of the total stored. Examples include personal/sensitive information (SSNs, bank records, medical records, private records), data with nostalgic/historical value (family photos, letters), and important business information (competition-sensitive information, personnel performance/salary, etc.).

*Low value* data typically represents a *large* portion of the total stored. Examples of low value data include non-sensitive files that could be easily reconstructed from source data, such as licensed software, extra copies of records, output of programs, etc.

#### 3.1.1.5 Value Known in Advance

In many cases, the value and decay time constant can be anticipated in advance, before the data is created. Such data may only be needed for near-term, and could be reconstructed easily. Other data in this category may be kept for purposes of nostalgia, and therefore be hard to reconstruct. In cases of knowing the time constant in advance, it may be kept for only a short period of time, measurable in hours or days, or potentially kept for much longer (months, years, decades). Additionally, data may be stored for a compulsory period of time, due to regulatory obligations or other compliance standards. For example, regulatory requirements may impose a minimum 5-year retaining period in such cases as Sarbanes-Oxley.

Next, we'll look at cases where the value of the data is either decreasing or increasing.

##### 3.1.1.5.1 Rapidly Decreasing

The highest percentage of data falls into the category where it is anticipated in advance to rapidly decrease in value over time. Examples are commercial advertisements, news releases, past project notes, and web browsing history. It is essentially data that had a short period of usefulness and immediately falls into a lesser value state.

#### 3.1.1.5.2 Slowly Decreasing

A lesser percentage of data is expected to slowly decrease in value over time. Examples include music files, books, and research data.

#### 3.1.1.5.3 Increasing

An even smaller percentage is expected in advance to *increase* in value over time. Examples include nostalgic photos, letters, and various data that have a certain level of emotional memories attached to them. Interestingly enough, this data may not be of any value to an adversary, much as a photo scrapbook does not hold the same information security risk as a collection of credit card numbers (nostalgic value vs. financial gain motive).

### 3.1.1.6 Value Not Known in Advance

The last category is data where the value and time constant *can't* be predicted in advance (data that you don't know you need until you need it). Examples include logs needed for forensics, or data that increases in value after an unexpected event. It may be necessary to use an average value for the decay constant and poll the system to determine if the constant should be modified based on usage habits or querying the importance from the user directly. One issue is that users may simply set the time constant to the maximum value for all files they are unsure of. It may be necessary to restrict this ability to administrators only.

#### 3.1.1.7 Frequency of use

The frequency of use also affects the value time constant. Each time the data is actually used, its value returns to 100%. A high frequency of use indicates that the particular dataset is important to the user; therefore the value does not decline as quickly as forgotten or unattended data. If usage activity begins to show a pattern of decline, so too will its associated value to the user.

### 3.1.2 Value of Data to an Adversary

The value of data to an *adversary* depends on the extent to which it helps them accomplish their objectives. Factors may include the importance to the organization, amount of personal data, cost of controlling that exposure, impact of exposure, dollar value, etc. High value data is defined as data such that the loss of it would cause immediate significant impact, and the effort to restore it is also significant. Low value data is defined as data such that the loss of it would *not* cause significant impact and/or the data can be easily restored.

#### 3.1.2.1 Loss of Data Confidentiality

One of the most severe consequences of data breach incidents is the loss of confidentiality that can result. Disclosure of sensitive information can result in serious financial, personal, and safety issues. All too often we hear about laptop thefts that result in the disclosure of personal records such as social security numbers and credit card information. The confidentiality loss of these records can jeopardize the well being of innocent citizens who had no idea that a particular organization was storing this type of

information. The converse is true that the organization storing the data can suffer serious damage to its business continuity and reputation.

#### 3.1.2.2 Loss of Data Integrity

Loss of data integrity is a serious issue, and one that can be attributed to such factors as hardware error. In the case of a secure storage system, the point of failure would most likely reside at the disk drive itself. Researchers at Google conducted a recent study to determine the failure trends in large disk drive populations [Pinheiro]. They found that, in 2002, an estimated 90% of all new information created was stored on magnetic media.

The focus of their study was to determine the failure rates and cause of failure in magnetic storage devices such as hard disk drives. Some surprising results of their research were that there was no direct correlation between increased failure rates as a result of high temperature and activity levels. Additionally, the researchers found that there were only a few Self-Monitoring Analysis Reporting Technology (SMART) parameters that contributed to probabilistic determination of drive errors (i.e. scan errors, reallocation counts, probational counts), and that on a large portion of drives that had failed, no SMART indicator had accurately predicted the event to occur.

The researchers collected rate of failure over a period of five years. They found the traditional reliability "bathtub curve", with a noticeably high early failure rate of approximately 3% in just the first 3 months of use. This failure rate sharply decreases to 2% for the first two years, until it dramatically increases to nearly 9% after 3 years of use [Pinheiro]. It is the 3-year mark that appears to be the defining point of failure in current

disk drives, and something to consider when constructing the data decay framework. Storage redundancy and error correction must be implemented to safeguard integrity of data, or information can be lost to an event such as irreparable disk drive failure.

#### 3.1.2.3 Loss of Non-Repudiation

Non-repudiation refers to the assurance of integrity for stored data on a system. Repudiation, or the ability to refute the claims of integrity, is jeopardized when an adversary has the capacity to alter or modify data without the consent of the owner. There are tools and technologies such as checksums and digital certificates available to verify that information is genuine. In addition, cryptographic transformations can verify that the information is unmodified through the act of a successful decryption (any small change to an encrypted file may change the outcome of decryption operations).

### 3.1.3 Data Pollution

Data pollution is defined as information that has no value to the user or an adversary. There is no longer a need to have this information stored on a system, yet it persists. The data may have held value at some point in time, but in its current form it only reduces the total available storage space for useful data. Additionally, this data pollution can grow to such a point that it reduces the efficiency of a system by increasing the time required to search for relevant data. An example of this frustration is the growing number of unrelated search hits when looking for a particular item online via search engines such as Google, Yahoo, Bing, etc.

An additional concern is the advent of social networking and websites such as Twitter, where recent statistics have shown that 10% of users are responsible for 90% of the data generated on the site [Hickins]. This figure tells us that only 10% of users will likely find the majority of data valuable, and the value of the data may be extremely short-lived. After this period of usefulness comes perpetual storage and the generation of additional data pollution. As in the days of industrialization and rapid progress, our ancestors had left a legacy of environmental pollution that we are now currently working to rectify.

It is without a doubt that we will see acceleration in the growth of data pollution, especially due to the increase of hard disk storage capacity. A recent paper estimated that in approximately 250 exabytes of new data ($2x10^{21}$ bits) were created in 2008, and that the figure will increase four-fold to 100 exabytes in 2010 [Wood]. Combined with an average growth rate of 44% per annum in hard disk storage space, there will be a serious concern in the near future regarding the amount of data clutter that exists. Drive space in 1956 was a mere 2 kb/in$^2$ compared to the 2008 density at 0.25 Terabits/in$^2$, and continual projected growth size in the years to come.

#### 3.1.3.1 Corollaries to Environmental Pollution

There are several corollaries that exist between data pollution and conventional environmental pollution. Based on recent discussions with a professional environmental engineer, Table 1 was developed to show examples of these corollaries [Dixon2009]. In both cases, the objective is to reduce the amount of unwanted material before it can cause significant problems. If left unaddressed, these effects can have wide-reaching consequences.

Table 1. Corollaries to Environmental Pollution. The causes, cleanup, and prevention of environmental pollution have corollaries in the information field.

| Environmental Arena | Data Corollary |
|---|---|
| Recycle | Clear storage medium for use by other data |
| Use less hazardous alternatives | Sanitize/obfuscate |
| Punishment | Negative audit results, press releases. |
| Incentives | Positive reputation |
| Cost of Control | Human/hardware resources and training |
| Release | Data breach |
| Classification of hazards (corrosive, toxic, flammable, etc.) | Data classification |
| Threshold limits | Partial dataset |
| Level and Length of exposure | Amount of data, time available to adversary |
| Legal/regulatory Requirements | Sarbanes-Oxley, HIPPA, DoD |
| Proper disposal | Secure erasure |
| Abatement | Limit generation of new data |
| Treatment | Encryption |
| Side effects | Usability, performance, user circumvention |
| Advanced knowledge of materials | Information security and networking |
| Sanitization | Data sanitization |

Next, the causes, cleanup, and prevention of data pollution will be discussed in further detail.

#### 3.1.3.2 Causes and Cleanup

Data pollution can be caused by an abundance of information, lack of practices to control the growth, and negative user actions such as policy non-compliance and creation of redundant datasets. What may make tasks easy in the short run for users could inevitably create a dangerous growth of data pollution that can block future productivity. There are serious consequences to allowing data to exist forever and not taking appropriate

actions to remove it after its period of usefulness has expired. To rectify the issue of the growth of excess data, there must be mechanisms in place to isolate useful information and securely erase the rest. Useful data would be measured by metrics such as frequency of use, time since last modification, among others.

#### 3.1.3.3 Prevention

Prevention consists of putting in place appropriate security practices, making users aware of the consequences of releasing sensitive information, and setting controls on what should be stored. Additionally, determinations for how long data should be stored, where it should be stored, and how it should be stored (encrypted, etc.) should be in place by the governing system. Other prevention activities include educating personnel with regards to combating data pollution, separation of user levels, auto deletion scripts/disk cleanup to remove non-sensitive data clutter (would require a secure removal strategy for sensitive information).

With respect to the future, those looking to prevent data pollution should question the benefit of having so many files in the first place. If one were to eliminate the root cause, then it would not be necessary to act after the fact with a decay framework to remove pollutant data. One emerging concept is to move away from standalone client systems and return to a mainframe system, where thin clients with no local storage connect and manipulate data stored in a central location. This trend is growing in popularity with collaborative products such as Google Docs. If there was only one point of storage, it would make the prevention of data pollution easier than policing multiple client machines with their own storage devices.

### 3.1.4 Half-Life

In this paper, the term "half-life" is meant to be the time it takes for the user data value or the adversary exposure risk to be reduced to half its initial value. We assume this follows an exponential curve in the same way as radioactive decay:

$$N_t = N_0 e^{-t/\tau} \qquad \text{[Eq. 3-1]}$$

Where:

$N_t$ = new value at time t

$N_o$ = initial value at time zero

$\tau$ = half life

## 3.2. Potential Decay Frameworks

The proposed solution is to desynchronize sensitive data between client and server in such a way that the only working copy over time will be on the server. The copy on the client will become unusable. As you disconnect and time elapses, the client files are gradually degraded such that increasing "effort" will be needed to restore them.

We'll first look at two methods of decay, involving file corruption and key corruption.

### 3.2.1 File Corruption

In file corruption, the file is gradually changed such that increasing effort is needed to reconstruct or restore it. This could be done by several methods, including breaking it into pieces, overwriting it, or encrypting/altering it.

#### 3.2.1.1 Break into increasing pieces

Mills and Znati offer insight into a secure storage system that utilizes a distributed hash table to break up a file and encrypt the pieces using a symmetric key [Mills2008]. It may be possible to continually repeat this process and break the file into smaller fragments, encrypting them each time with a new key to create multiple levels of obfuscation and increase the time required to reverse the process.

As time progresses, the file would require an increasing level of effort to be restored, as the layers of complexity are built. An additional point to consider is that the original file now exists as a collection of smaller files, including the metadata responsible for successfully piecing the fragments together. If a fragment is lost, the ability for the adversary to obtain any useful information is severely impacted. Additionally, some or all of the decryption keys may be stored off-site at the central server, which are re-obtained only through the next successful authentication by the client.

#### 3.2.1.2 Overwrite

Another method could be to gradually overwrite (“zeroize”) random segments of the file so it becomes more and more unavailable. This method would eventually overwrite

the entire contents of the file, making it completely unreadable and in a state similar to zero-pass secure erasure. A downside to this method is that a system implementing randomized overwrites must have some manner of correcting the "errors" introduced to the file, otherwise this becomes a one-way operation. There may exist a specific error tolerance in the decay framework to allow corrections to a file for restoration before a point of no return is reached. The benefit of a system such as this is an eventual guaranteed removal of a file if the decay operation is allowed to run beyond the threshold.

#### 3.2.1.3 Encrypt/alter increasing segments

Another method could be to gradually encrypt or alter segments of a file in a predetermined way. This would require later decryption in order to access it properly. Consideration is given to the careful placement of file markers to denote which segments have been encrypted/altered, in order to restore the file to its original state without issue. The markers would consist of a bit signature appended to the encrypted segment, so that the framework can identify the length of the transformation and undo it later on.

#### 3.2.1.4 Reduce Redundancy

Another method could be to start with multiple levels of redundancy and strip off one layer every "half-life". In the final step, the file would be completely deleted. During restoration, you'd rebuild these layers over time.

### 3.2.2 Key Corruption

Alternatively, a file's encryption key could be altered. Researchers at Lockheed Martin have proposed one possible solution, where the encryption key is stored in volatile memory only [Lee2005]. In the event of a perceived threat, the encryption key can be purged from memory and made unavailable to an adversary. It is possible to recover the encryption key after an authorized user, who has knowledge of the initial password to re-generate the key, provides this piece of information when the system receives power once again. This is an effective means for quickly dumping sensitive data in the event of an immediate threat, and provides the means for restoring access to the information at a time when it is once again safe to do so.

In the data decay framework, key storage in volatile memory may serve as an effective means of allowing access to sensitive information when the machine is powered on and entrusted to an authorized individual. The idea proposed by Lee et. al utilizes a "weight on wheels" switch that triggers the destruction of the encryption keys in volatile memory when a helicopter that uses their system takes off. This concept could be applied to a device such as a computer, triggering a key wipe in memory when the machine detects the closing of the laptop lid.

Another patent by IBM suggests a system of data encryption and storage that purges keys after a period of inactivity [Blakely1997]. When applied to the data decay framework, this inactivity could be measured as periods of time in which the stored data on a client has not been used. After a predefined threshold of time that the file has not been accessed by the user, it would trigger the corruption of the encryption key.

#### 3.2.2.1 Break into increasing pieces

Dodis and Yung suggest a hierarchical identity-based encryption with continual refreshing that gradually transforms the key, shifting it out of phase [Dodis2003]. The advantage of this system is that it is significantly harder for an adversary to determine the encryption key as it undergoes phase transformations over time. An obvious pitfall to this scenario is that the user of the data may potentially lose the original encryption key and be blocked from accessing the encrypted data. The key could be lost due to a malformed transformation that disallows proper reconstruction.

#### 3.2.2.2 Overwrite

Another method is to gradually overwrite ("zeroize") random segments of the key so it becomes more and more unavailable. As with the phase shift idea, the benefit is that the adversary will be missing critical information necessary to utilize the encryption key. In order for the encryption key to function properly for decrypt/encrypt activities, it must be in a whole state, which introducing overwrites to random sections of the file over time would prevent. An obvious downside is that the system must be able to discern where it performed the overwrite process in order to restore the key later on. Additionally, the system has to know the previous value that it wrote over, which means that this information might be stored somewhere on the system that can be retrieved by an adversary.

#### 3.2.2.3 Encrypt/alter increasing segments

Another method could be to gradually encrypt or alter segments of a key in a predetermined way. This would require later decryption in order to access it properly. The benefit of this method is that only particular segments need to be encrypted, rather than performing encryption on the entire key, in order to mask its value. A patent by researchers at Microsoft specifically outlines this concept of breaking up a file and storing both encrypted and unencrypted portions of it on a single storage device [Yasukawa1999]. As with previously discussed methods, the obvious concern is the proper reassembly and decryption of the file in order to return to the intact original copy.

## 3.3. Time Constants for Decay

The data decay initiation and rate of degradation depends on the values set in advance or dynamically for: 1) the initial grace period; 2) the value of the data to the user; 3) the value of the data to an adversary; 4) the ability to protect the data; and 5) the impact of the data on data pollution. When the value of the data to an adversary is greater than the value to the user, the ability to protect it is low, and the impact of the data on data pollution is high, the data decay should be performed more rapidly.

### 3.3.1 Grace Period Before Decay Starts

It might be advantageous to have a period of time in which there is *no decay* to allow casual users to continue work on their datasets when disconnected. This would accommodate unexpected disruption of service (power outages, network outages, in a region where you can't connect, etc.).

The length of the grace period could be determined by usage, work schedule, equipment availability, or other factors. It could be either set in advance or be self-adjusting based on patterns of usage.

### 3.3.2 Maximum Decay Rate

The *maximum* decay rate setting would determine the *fastest* time in which files would be degraded and/or eliminated. For some files, this would be limited by regulatory or other factors (e.g. must keep current week timecard data for at least 7 days, etc.)

### 3.3.3 Minimum Decay Rate

The *minimum* decay rate setting would determine the *slowest* time in which files would be degraded and/or eliminated. At the limit, this would be bounded by the life of the computer or storage medium itself (e.g. the disk drive life may only be three to ten years).

### 3.3.4 Example Decay Formula

The following is an example of how a combination of the above factors could be used to determine the data decay rate. During data decay, the remaining amount of data D as a function of time would be:

for $t < t_g$ (*during* grace period)

$$D(t) = D_i \qquad \text{[Eq. 3-2]}$$

for $t > t_g$ (*after* grace period)

$$D(t) = D_i (1-\lambda)^t \quad \text{[Eq. 3-3]}$$

where:

$$\lambda = \lambda_{min} + (2+V_a+I_{dp}-V_u-A_p)( \lambda_{max}-\lambda_{min})/4 \quad \text{[Eq. 3-4]}$$

and

$D_i$ = initial amount of data

$\lambda_{max}$ = maximum decay rate (quick)

$\lambda_{min}$ = minimum decay rate (slow)

$t_g$ = grace period

$V_u$ = value of the data to the user (dimensionless number between 0 and 1)

$V_a$ = value of the data to an adversary (dimensionless number between 0 and 1)

$A_p$ = ability to protect the data (dimensionless number between 0 and 1)

$I_{dp}$ = impact of the data on data pollution (dimensionless number between 0 and 1)

If $V_u$, $V_a$, $A_p$, and $I_{dp}$ are all set at their midpoint of 0.5, the decay rate would be *halfway* between the minimum and maximum. If $V_a$ and $I_{dp}$ are high, the decay rate would *increase* to eliminate the data more quickly. If $V_u$ and $A_p$ are high, the decay rate would *decrease* to allow the data to exist longer.

Figure 2 shows a data decay plot for two different scenarios using Eq. 3-3. In the first scenario (blue line), the data has more value to the user than to an adversary ($V_u > V_a$), so the decay rate is slow. In the second scenario (green line), the value of the data is greater to the adversary than to the user ($V_a > V_u$), so the decay rate is faster. The horizontal line at the beginning of the plot represents the grace period set to four days.

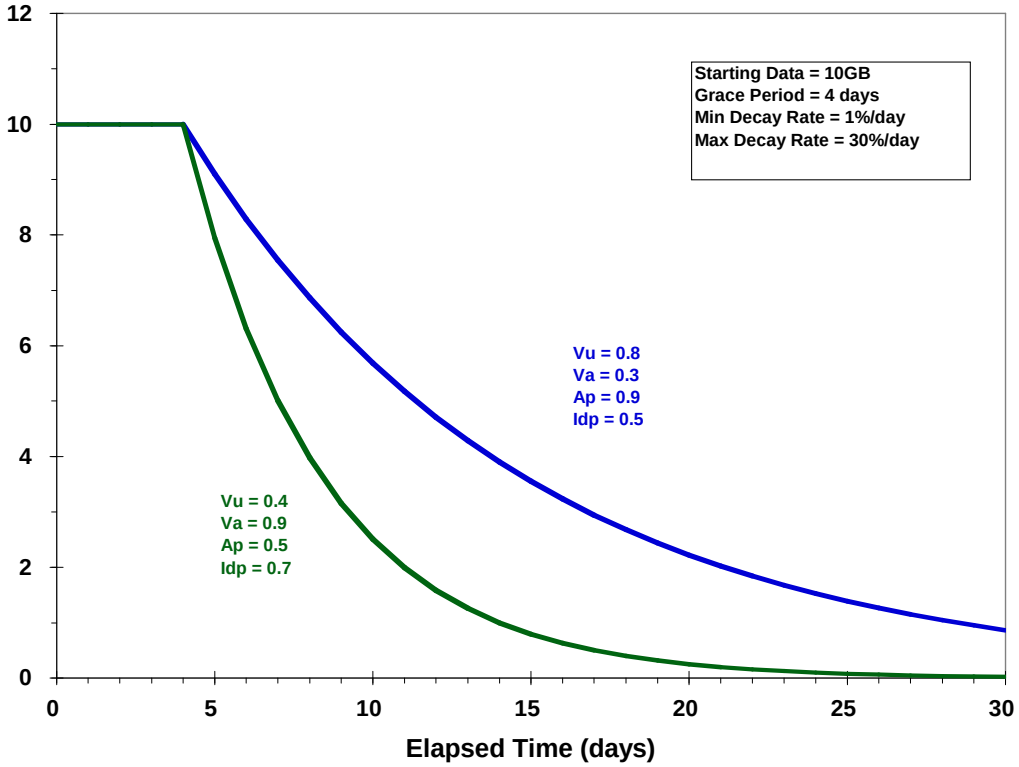


Figure 2. Data Decay Plot. Depiction of data decay process, where the blue line represents a scenario in which data has more value to the user than to an adversary. The green line represents a scenario where the value of the data is greater to the adversary than to the user, so the decay rate increases. The horizontal line at the beginning of the plot represents the grace period set to four days.

Next, we'll look at how the type and size of the data, along with other factors, can determine the best decay characteristics.

Table 2 lists describes four data storage scenarios: 1) television DVR recordings; 2) personal pictures; 3) PDA backup files (encrypted); and 4) sensitive military data (local copy). The amount of data stored would range from 0.64GB (low impact on data pollution) to 35GB (high impact on data pollution). The decay rate limits would range from 0.01%/day (~10-15 year half life) to 50%/day (1 day half life). The grace period would range from 0.005 days (~6 minutes) to 30 days (~1 month). The value of the data to a user

or adversary, the ability to protect the data, and the impact on data pollution would range from 0.1 (very low) to 1.0 (very high).

Table 2. Data Decay Scenarios. Four distinct examples, ranging from low-value to high-value data, were examined to determine how decay settings could be applied.

| Parameter | Symbol | Television DVR | | Personal Pictures | | PDA Backup Data (Encrypted) | | Sensitive Military Data (Local Copy) | |
|---|---|---|---|---|---|---|---|---|---|
| | | Value | Rationale | Value | Rationale | Value | Rationale | Value | Rationale |
| **Initial amount of data (GB)** | Di | 35 | 50 shows at 0.7GB each | 10 | 10,000 images at 0.001GB each | 0.64 | 64 files at 0.01GB each | 2 | 400 files at 0.005GB each |
| **Max (fast) Decay Rate (%/day)** | Lambda max | 50 | Will watch show within 1 day | 0.6 | Keep at least 4 years | 2 | Will re-sync PDA within 1 month | 10 | Keep at least 1 week |
| **Min (slow) Decay Rate (%/day)** | Lambda min | 0.01 | Limited by disk drive life of 10-15 years | 0.01 | Limited by disk drive life of 10-15 years | 0.8 | Limited by PDA life of 3 years | 0.1 | Limited by development period (18 months) |
| **Grace Period (days)** | tg | 0.005 | Length of typical power outage | 30 | Start decay if PC not powered on for 1 month | 10 | Start decay if PC not powered on for 10 days | 1 | Start decay if PC not connected to server for 1 day |
| **Value to user** | Vu | 0.1 | Low | 0.9 | High | 0.4 | Medium (can be restored from PDA) | 0.5 | Medium (can be restored from server) |
| **Value to adversary** | Va | 0.1 | Low | 0.2 | Low | 0.7 | High | 1.0 | High |
| **Ability to protect the data** | Ap | 0.1 | Low | 0.3 | Low | 0.5 | Medium | 1.0 | High |
| **Impact on data pollution** | Idp | 1.0 | High | 0.8 | High | 0.1 | Low | 0.3 | Low-Medium |

Figure 3 shows the resulting decay curves for the above scenarios. The television DVR data takes up significant storage space with only short-term value, so it is set to decay *rapidly*. The personal pictures take up almost as much space, but have longer-term user value and are set to decay more *slowly*. Both the PDA and military data are sensitive, but they are reasonably protected from unintentional disclosure and don't contribute significantly to data pollution, so they are set to decay at a *medium* rate. The effect of a *grace period* can also be seen, as on the personal pictures, where decay doesn't start for 30 days.

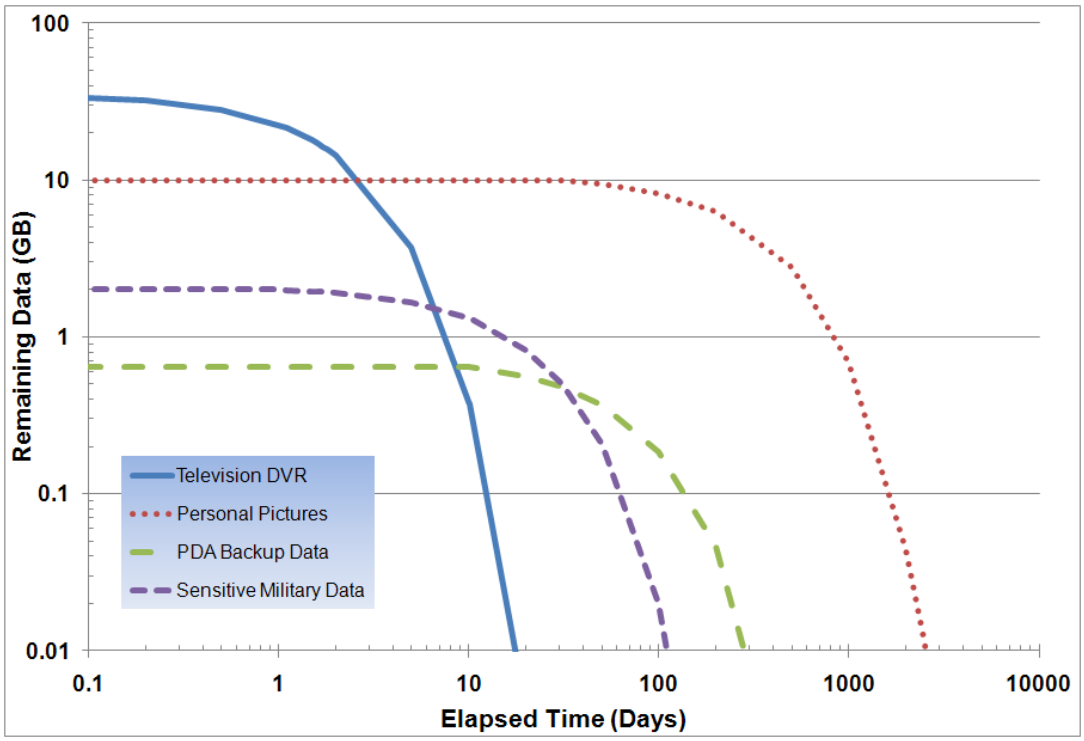


Figure 3. Data Decay Scenarios. Large size, low-user-value television DVR data would have a short grace period and decay rapidly, whereas large-size, high-user-value personal picture data would have a longer grace period and decay more slowly. Medium-size sensitive personal or military data would have a short to medium grace period and the decay rate would depend on the level of protection and value to an adversary.

### 3.3.5 Time Constant Override

In adverse situations it should be possible for the user of the system to initiate an immediate forced data wipe on command. This may also serve as a means to securely dispose of old client systems as well. A procedure similar to the Unix 'shred' command would begin a zero-pass write over data flagged as belonging to the decay framework's scope. A related feature exists on Blackberry handheld devices where one can issue a remote wipe of the device. A signal is received by the handheld on the cell network and it executes a zero-pass erasure on demand. The issue with this method is the difficulty in retrieving data once the operation is performed. It is assumed that the data is unrecoverable, or would require significant effort to reconstruct for use.

### 3.3.6 Data Recovery Example

Figure 3 depicts data recovery after a decay operation has been applied. It is shown that recovery commences after an activity spike, indicating the user's desire to access the information. The effort and time to recover a file should equate to the total time that decay had occurred, discouraging access by an adversary and ensuring that the contents of the data remain confidential.

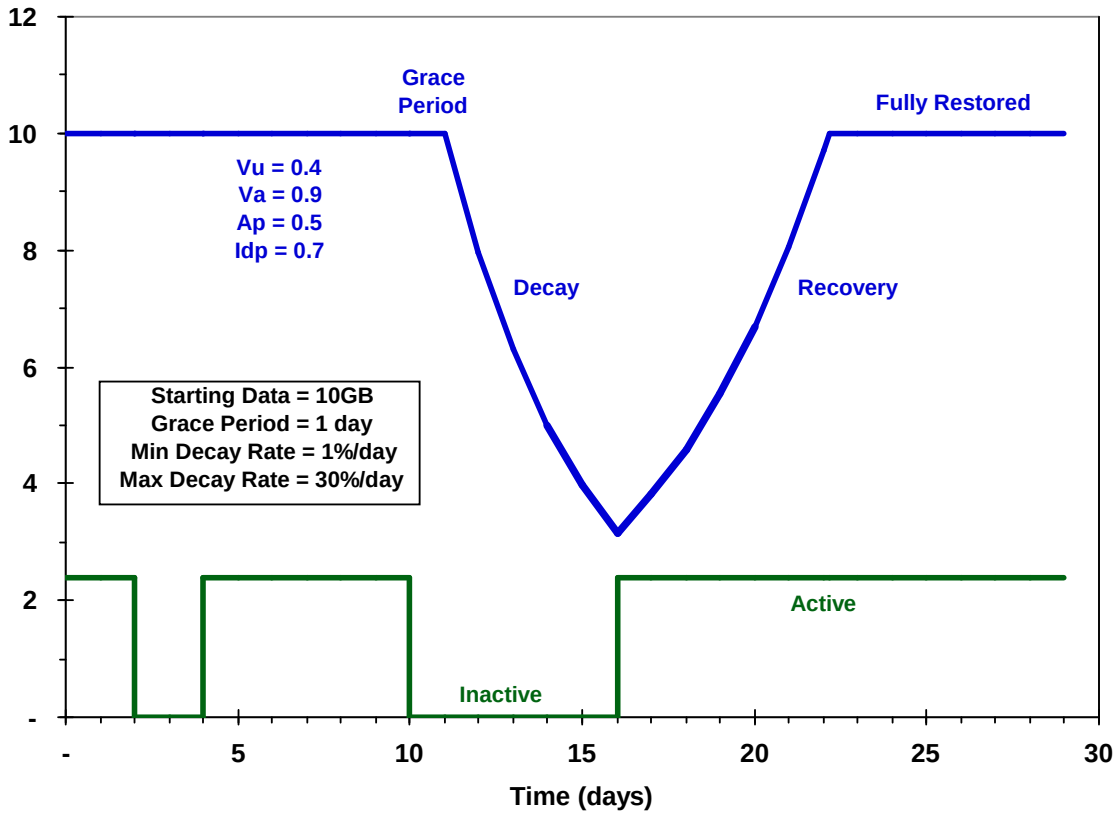


Figure 3. Data Recovery Plot. Depiction of data recovery process. The green lines indicate periods of activity by the user of the data, while the blue line represents the decay and recovery state as charted through time in days. The time to recover data equates to the time that the decay operation was allowed to run.

## 3.4. Determining Elapsed Time

There are several ways of determining the elapsed time using internal or external sources of information.

### 3.4.1 BIOS clock

The BIOS clock acts as a hardware timekeeper for a computing system, and often supplies higher-level applications and software with a benchmark for the current time value. The benefit of a BIOS clock is that it continues to keep time even when a computer is shut off, which is especially beneficial in client systems such as laptops, that may be powered off while in transit. However, like other timekeeping systems, it is still possible to manually alter the BIOS clock value and falsely report the date and time to querying applications.

### 3.4.2 OS clock

The OS clock is often used by running software on a computing system to determine the current date and time. This is, unfortunately one of the easier timekeepers to fool, and often the first place that an adversary would go to disrupt security applications that are tied into time reporting systems. On many systems, the ability to change the date and time on a system is reserved for administrators only, but there exists the possibility of system-level exploits to bypass this security feature.

### 3.4.3 Network time server

Network timeservers can be utilized to inform the client machine of the current date and time, offering an up-to-date and accurate depiction of the passage of time. However, the immediate drawback is that the client must be connected to the Internet in order to receive Network Time Protocol (NTP) messages.

### 3.4.4 Anti-spoofing

It is assumed that one of the first attempts by an adversary to thwart the protection of a time-based decay framework would be to exploit the system's notion of time and spoof its value. This is a serious concern that could potentially block the protection of time-based data decay, if the system believes that no time has passed at all. The framework must identify the ways in which an attacker could spoof the time (e.g. modify BIOS or OS clock, unverified NTP updates from a masquerading server, block network connectivity altogether to prevent time notifications).

### 3.4.5 Correlation from Multiple Sources

The best method of preventing spoofing of the elapsed time by an adversary is to correlate inputs from multiple sources. These sources could include NTP, OS and BIOS clocks, and update messages from the decay framework server. In the event that the system is cut off from network communication and has fewer options to evaluate the current time, it may be possible to look at internal activities.

It has been suggested that low level processes, such as system I/O operations, can be used to help estimate the passage of time [Border2009]. A patent exists describing a system of time coordination through multiple sources, such as GPS, radio waves, and television signals [Kihara]. Additionally, they suggest using unconventional mediums such as magnetic fields, AC power, and vibration/pressure in the environment to measure changes in time. Overall, the more sources there are to corroborate the current time, the higher confidence one has in determining the level of decay to implement for the files. Siegel et al. describe a method using the speed of CPU instruction cycles in relation to the number of executing cycles versus system clock transitions [Siegel95]. Low-level activities such as data read/writes on a system may provide the capacity to determine the passage of time without reliance on metrics that can be spoofed.

Another independent approach to measuring time is defined in a second patent [Kuczynski], where a small source of ionizing radiation is mounted next to a radiation sensitive material that collects the emissions. In this manner, a time interval is developed based on the consistent pattern of radioactive emission and the changes observed in the collector. One benefit of this method is that it does not require an external power source to perform the action, so theoretically one could place this system in a client machine that undergoes periods where it is turned off, yet have a reliable time source.

## 3.5. Security Drawbacks

Unfortunately, even with well-designed systems, there exists the potential for security to be compromised. The following sections detail possible attack vectors against the data decay framework and how to avoid their effects.

### 3.5.1 Inference Attacks

If the data decay method lacks complexity, an adversary may be able to predict the outcome and reconstruct data. In choosing an acceptable decay operation, an important question to ask is how easy it would be for an outside observer to estimate the outcome of the data transformation. Dodis and Yung suggest utilizing secret sharing schemes across storage nodes to protect against partial exposure [Dodis2003]. In the event that some information is gathered, the overall security of the system is not easily threatened.

### 3.5.2 Eavesdropping

As a system relies on communication across a network to coordinate instructions from the server to the client, it may be possible for an adversary to monitor communications and gain valuable information regarding how to exploit the data decay framework. It is crucial that the communication link between server and client utilizes strong encryption to encode messages and add layers of complexity to thwart eavesdropping attempts. Additionally, the framework should strive to make it difficult for an adversary to obtain any useful information from simply watching the flow of messages between the players involved in the framework. In a research paper by Blanchet, a method of secure file storage on untrustworthy networks using a "ProVerif" utility to assess the security of participating systems is suggested [Blanchet2008]. If the confidence level were questionable, operations would be deferred.

### 3.5.3 User Non-Compliance

The human element remains one of the most critical links in information security, and non-compliance by users of the data decay framework can circumvent the protection it offers. Users may set the decay time constant too long (e.g. 100yrs), thus the data resides on the system for an unacceptably long timeframe. Additionally, users may attempt to make unauthorized copies of the files, hoping to remove the constraints of the decay framework's security.

### 3.5.4 Man-in-the-Middle Attacks

A very dangerous scenario involves a man-in-the-middle attack, where the adversary masquerades as a participant in communications between client and server. When the identity of the endpoint that the client and/or server is communicating with cannot be verified, one must suspect that an adversary may be sending and receiving messages across the network to fool the recipient into thinking that they are the trusted party. This can result in a number of security issues, such as a client receiving incorrect instructions from a false server and unknowingly releasing sensitive information or reduce security strength on the client (e.g. a server update instructing the client to increase the time necessary to decay files, or not at all).

Fortunately, security researchers have examined this issue and developed means to protect systems against man-in-the-middle attacks. The use of digital signed certificates by commercial and enterprise servers allows the confirmation of identity to assure connecting clients that they are interacting with a trusted party. In the data decay framework, it is

important to realize the threat that MITM attacks pose, and implement countermeasures such as certificate verification to thwart adversaries.

## 3.6. Usage Drawbacks

As with any security framework, there exists a tradeoff between usability and protection. Some issues that may arise include the inability to work with the data while it is in the process of being restored from the effects of decay. The time required to restore a file renders it unusable until the operation is completed, which is an obvious drawback in terms of usability. In this section we will look at various types of drawbacks that one may encounter when using a system that incorporates the data decay framework.

### 3.6.1 Unintentional Loss

There are a number of events that can contribute to the unintentional loss of data. For example, a power failure or network outage can render the client and server unable to communicate with each other, or suddenly halt an operation such as storage and retrieval between the two. Accidental file corruption can render good data unusable on both client and server side, therefore data storage redundancy is an important consideration in developing a successful data decay framework. Lastly, software bugs created by programming errors can occur and are highly likely without careful quality control and application testing.

### 3.6.2 Server Availability

The ability to obtain new files, archive revisions of existing data, and receive administrative updates to the framework rests on the ability for the client system to communicate with the server. In the event that the client cannot establish a secure connection, it must rely on its own capabilities in managing the timed decay of data and operate based on past instructions from the last server connect. An adversary may intentionally block access to the server in hopes that they can defeat the security of the data decay framework, or a general networking/server uptime issue may block access as well. A client that has the capacity to communicate frequently with the server will provide a better user experience, but the decay framework should still be able to operate under conditions where it has not had a recent session.

### 3.6.3 Time to Restore

Loss of productivity can result from the time required to wait for decayed files to be fully restored on the client. This is an inconvenience that directly affects usability, and is an unfortunate tradeoff in the quest for security. Size of the file, duration of decay, and speed of the machine are all factors that will determine the time required to return to complete data restoration. In the event that the time to restore is too lengthy, a user would be able to pull a new copy of the file from the server instead. The amount of recent data lost as a result of this method would factor into the frequency of connections made to the server in which client and server files were synchronized.

## 3.7. Comparison with Existing Methods

In the following section we will examine current methods for achieving the goals set out in the data decay framework. How they operate and a discussion on their pros and cons will be presented.

### 3.7.1 Software Methods

There exist auto-deletion programs that will remove files based on metrics such as age (e.g. MS Outlook auto-archive), usage (e.g. MS Windows unused desktop icons), and file properties (e.g. MS Windows disk cleanup). These are relatively simple programs that perform basic file maintenance, and coincidentally help to address the issue of data pollution. For many of these applications, user intervention is required to initiate and confirm the files designated for removal.

A recent announcement has been made regarding security software that appears to be the most closely related program in existence to the goals of data decay. The product is called "Vanish," and it is designed to set a specific lifespan for files, at which point they are no longer usable [Geambasu2009]. Unlike the data decay framework, the files are *permanently unreadable*, with no capability to restore the data for use later on. The system uses Vanishing Data Objects (VDOs) to encapsulate data that is designated for destruction after a period of time.

The scope of the Vanish software is past archived data, such as copies of email correspondences that can result in the loss of repudiation later on if they are allowed to

persist. Temporary encryption keys are stored across a P2P network in distributed hash tables (DHTs). Unlike the decay framework, there is no central management system; rather the process occurs over a decentralized network of nodes. Vanish shares goals very much in line with the data decay framework, and it will certainly be exciting to see further research into the method proposed by Geambasu et al.

### 3.7.2 Manual Methods

Manual methods for removal may consist of activities such as a user sorting files by date and selecting items for deletion that are older than a specific point in time. This is normally not a fast or efficient means for removing extraneous data, and often consists of repetitive actions by the user to complete the task. In some instances the user may write a simple script to carry out the removal of particular datasets with less need for constant interaction with the system.

### 3.7.3 Hardware Methods

Methods of decaying data via hardware implementation are evaluated in the following section. The benefit of utilizing a hardware solution is that it is rudimentarily simple in practice and less prone to bugs and reliability issues observed in software counterparts.

#### 3.7.3.1 Volatile Memory

One hardware method to immediately remove any trace of stored data is to make use of volatile memory such as DRAM. When power is cut to these devices, the stored contents are erased (assuming the devices are at room temperature). Volatile memory requires a

constant source of power in order to retain the stored information for any length of time. Unlike magnetic storage hard disk drives, there is no trace of the information after removal, adding an additional level of protection.

#### 3.7.3.2 Ionizing Radiation

Another hardware method is to use an ionizing radiation source to flip random bits continuously (eventually overwhelming the ECC). This could be accomplished with certain DRAMs or SRAMs by building them with high-uranium/thorium content ceramic, lead, or other packaging materials. These package-induced alpha particle soft errors were first noticed in DRAM devices in the late 1970s [MayWoods].

Another natural source of ionizing radiation that can flip memory bits is terrestrial cosmic rays. Terrestrial cosmic ray effects were first predicted in 1979 by researchers from IBM [Ziegler79]. Although these effects are small at sea level, one could further enhance DRAM sensitivity to soft errors induced by atmospheric neutrons (by a factor of 1,000 or more) with boron doping [Phillips79]. This would create "volatile" memories whose contents decay randomly over time (would follow the half-life curve), independent of software or user control. The contents could be kept valid by periodic hardware or software controlled "scrubbing" every few days (or on bootup). Once the scrubbing stopped (client not connected/used), the number of flipped bits would eventually accumulate and overwhelm the ECC. Note that the atmospheric neutrons (aka "cosmic rays") don't cause permanent damage to the memories, only random bit flips.

This technique could also be applied to lower-density static random access memories (SRAMs), which are sometimes used in a battery-backed non-volatile mode. It's not applicable to present-day "flash memory", since the floating gate non-volatile memories aren't as susceptible to soft errors.

#### 3.7.3.3 No Local Memory

Most "dumb terminals" that exist today have some form of local memory attached or embedded in them. If this is the case, then it is necessary to have a secure erasure method for the attached memory, such as overwriting the existing data or cutting the power to the memory module before the information can be compromised.

#### 3.7.3.4 Hardware Encryption

Hard drive manufacturers are looking into the development of encryption chips that reside on the drive itself and perform the encrypt/decrypt operations independently of the operating system. This offers an unprecedented level of security, but there is still the risk that adversaries may reverse engineer the hardware or discover circumvention techniques to bypass the hardware encryption systems.

### 3.7.4 Comparison Summary

In summary, existing methods offer quick and efficient means to erase specific types of information. With applications like Windows disk cleanup, the user must manually initiate the operation, and files are not securely erased (still recoverable from the hard disk using forensic tools). On the opposite side of the spectrum, DRAM erasure quickly removes data,

however it is assumed to be lost forever. The data decay framework offers the ability to securely remove sensitive information over time, while retaining the capacity to restore information when it is needed again.

### 3.7.5 Anticipated User Reaction

Initial reaction to the data decay framework will most likely be irritation at the fact that the users have lost a degree of control over their data on client machines. Having to wait a period of time to reclaim a file if it has been decayed, instead of having immediate access to the data, will most likely initially frustrate users. Additionally, some users may immediately attempt to circumvent the controls put forth by the framework, in order to retain files indefinitely. However, many will soon find the value and necessity of protecting sensitive information while in transit with their client machine, as well as the reduction in useless junk data residing on the system. This framework is designed to benefit those who travel often and to areas where they cannot guarantee the safety of their stored information. Early adopters of the framework will more than likely include international business travelers and government entities who seek to protect their data without requiring complete vigilance by the user at all times. The data decay framework offers a simple and automated system for ensuring that sensitive data does not fall into the wrong hands.

### 3.7.6 Future User Needs

Users of the framework will most likely want to see a more robust control interface in future releases, with the ability to fine tune settings regarding the files slated for decay. Additionally, they may want enhanced statistics and feedback regarding the percentage of

decay, as well as the time required to restore files to their original state. Visual notifications with graphics and real-time statistics could be made available to the user, for an overall nicer experience (instead of not knowing when/how far along their data has decayed, or when it will be restored).

# Chapter 4 Summary

## 4.1. Conclusions

In summary, data decay is the process by which information is degraded over time. With the increase in the occurrence and severity of data breach incidents, we must look at new ways to safeguard sensitive information. This framework aims to directly combat the threat faced by data breaches, in addition to the growing concern of data pollution. When information is no longer valuable to a user, it exists as nothing more than a waste product of the digital world. Containment of sensitive information and the reduction of pollutant data will create a safer and more efficient technology-focused world.

There are a number of considerations when deploying the framework for data decay, such as user acceptance and non-compliance, spoofing attacks, and the risk that methods used to decay the information may render it unrecoverable later on. Determination of the rate of decay for files is based on a value system in determining the worth of information. To better explain how this evaluation works, a mathematical formula was designed to compute the rate of decay based on factors such as user and adversary value, ability to protect the data, and the potential for the data to be pollution.

In regards to the hypotheses developed earlier, the need to limit the lifespan of data was confirmed through the selection of reviewed literature, describing the growth of data in relation to the increase in reported vulnerabilities, as well as corollaries to environmental pollution. Development of a well-defined decay framework with suggested decay and

restoration schemes aided in confirming the earlier hypothesis that a rules-based decay scheme in a client/server system can solve the problem of data breaches and pollution.

## 4.2. Recommendations

The use of a well-formed grace period for data under the scope of the decay framework should be implemented to help new users adjust to the system. On the flip-side, if decay rates are too generous, the protection offered by the framework is diminished. Adopters of the system must determine a balance between usability and security that meets the objectives of their organization. A small degree of frustration caused by the system can more than make up for the consequences of a data breach incident's cost to recover.

When possible, users of the framework should make sure that they push relevant updates to the server to avoid any potential for loss of information when client-side data undergoes decay. The one location where the integrity of a file is guaranteed is at the server, and it is assumed that the client will be exposed to situations where the data could be compromised without safeguards in place.

# Chapter 5 Future Work

The information presented in this thesis offers a variety of ideas regarding the future development of secure storage systems. Suggested strategies, evaluation of existing systems, and concerns that should be addressed in the implementation of data decay have been presented to assist future research into the topic. Several suggestions for future research are presented in this chapter to provide worthwhile ideas to pursue.

## 5.1. Refined Data Decay Time Constants

In future iterations of the framework, it would be worthwhile to refine the use of Vu, Va, Ap, and Idp factors in determining the decay rate so they're not just equally weighted.

One may also consider implementing *dynamic* factors, so decay rate can change in response to real-world conditions. For example:

- $I_{dp}$ could change based on the current amount of data or available drive space
- $A_p$ could change based on the current security "score"
- $V_a$ could change based on the recent frequency of breach attempts
- $V_u$ could change based on the recent frequency of user data access
- Grace period could change based on recent user habits (work schedule, etc.)

One should also re-consider whether data recovery should be designed to take time, so as to discourage retrieving unnecessary data.

## 5.2. Improved Hardware Methods

Investigation of tamper-resistant hardware methods for automatic data decay may prove extremely useful in advancing the decay framework. Utilization of hardware such as volatile memory that is closer to true volatility, or even using memory with intentional imperfections to introduce corruption in the data over time may become a useful paradigm.

## 5.3. Impact of increasing distributed computing

Looking ahead, some trends to consider will be the shift to cloud computing and distributed systems. There will be a bigger focus on collaborative web-based software where datasets are concurrently worked on and shared amongst users. The design of the data decay framework addresses this by fitting the client-server model that we are actively moving towards in the future. At the heart of the framework is the central repository located on one or more servers that act as the collective point of access by clients. The shift towards distributed systems and cloud architecture empowers the framework by helping people adjust to the paradigm of storing their data on a remote system instead of keeping files locally.

It is certainly an exciting time in computing, and one of the most important concepts in designing future systems is anticipating security issues that may arise, and to select appropriate mitigation strategies, while remaining aware of tradeoffs that follow. The data decay framework aims to remain flexible and ready to adapt to changes in computing in the years to come.

## 5.4. Spin-offs From Neurological Studies

Improvements to the decay framework may benefit from understanding how the human brain handles information overload, scrubs old, less-important memories, and reinforces important ones. There may exist a tie-in to the reduction of data pollution by treating lower value data as short-term memories and important data as long-term. Advanced studies may be able to treat data similar to memories only decipherable in the context of the owner (linked to total experience), and not necessarily transportable to others (adversaries).

## 5.5. Artificial Intelligence

To solve the problem of accurately determining the passage of time, it may be advantageous to examine the problem from an artificial intelligence approach. Future systems will have increased computing power and eventually take advantage of neural networks and other designs that mimic human neural pathways. The determination of time may one day approach a near-sentient ability by the client.

# References


MW Merriam-Webster online dictionary

WP Wikipedia online encyclopedia

Privacyrights.org *A Chronology of Data Breaches*. (2005). Retrieved October 13, 2008, from Privacy Rights Clearinghouse Web site: http://www.privacyrights.org/ar/ChronDataBreaches.htm

Rosenberg Rosenberg, B. (2007, February 1). *Chronology of Data Breaches 2006: Analysis*. Retrieved October 14, 2008, from http://www.privacyrights.org/ar/DataBreaches2006-Analysis.htm

Schneier Schneier, B. (2009, January/February). Architecture of Security. IEEE Security and Privacy.

Mulligan-Schwartz Mulligan, D., & Schwartz, A. (2000). Your Place or Mine?: Privacy Concerns and Solutions for Server and Client-Side Storage of Personal Information. Retrieved October 14, 2008, from http://portal.acm.org/citation.cfm?id=332255

Birme Birme, J. (2005, January 24). Document Management System Security. Retrieved October 15, 2008, from http://www.cs.umu.se/education/examina/Rapporter/JonasBirme.pdf

Riedel Riedel, E., Kallahalla, M., & Swaminathan, R. (n.d.). A Framework for Evaluating Storage System Security. In FAST '02 Conference on File and Storage Technologies (pp. 15-30). Retrieved May 15, 2009, from https://www.usenix.org/publications/library/proceedings/fast02/full_papers/riedel/riedel_html/

Wylie Wylie, J. J., Bigrigg, M. W., Strunk, J. D., Ganger, G. R., Kiliccote, H., & Khosla, P. K. (2000, August). Survivable Information Storage Systems. Retrieved May 15, 2009, from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=863969

Pasiewicz Pasiewicz, M. (2008, April). On People, the Death of Privacy, and Data Pollution. Retrieved May 10, 2009, from EDUCASE Review Web site: http://www.schneier.com/news-055.html

SOX Sarbanes-Oxley Act. (n.d.). Wikipedia. Retrieved July 3, 2009, from http://en.wikipedia.org/wiki/Sarbanes-Oxley_Act

HIPAA Health Insurance Portability and Accountability Act. (n.d.). Wikipedia. Retrieved July 3, 2009, from http://en.wikipedia.org/wiki/HIPPA

Pinheiro Pinheiro, E., Weber, W. D., & Barroso, L. A. (2007, February). Failure Trends in Large Disk Drive Population. Retrieved June, 2009, from Google Inc. Web site: https://www.usenix.org/events/fast07/tech/full_papers/pinheiro/pinheiro.pdf

Hickins Hickins, M. (2009, June 11). Is Twitter The New Wikipedia? Message posted to http://www.informationweek.com/blog/main/archives/2009/06/istwitter_the.html

Wood Wood, R. (2008, July 29). Future Hard Disk Drive Systems. Retrieved July 9,2009, from Hitachi Global Storage Technologies Web site: http://www.sciencedirect.com/

Dixon2009 Private communication with Mark Dixon, Environmental Engineer, on June 27th, 2009

PGP Grenier, C., Rice, T., & Spinney, M. (2009, February 2). Press Release: Ponemon Study Shows Data Breach Costs Continue to Rise. Retrieved July 18, 2009, from http://www.pgp.com/insight/newsroom/press_releases/2008_annual_study_cost_of_data_breach.htm

Mills2008 B. Mills and T. Znati, "Increasing DHT Data Security by Scattering Data," *IEEE*, 2008.

Lee2005 Patent No.: US 6,928,551 Bl (45) Date of Patent: Aug. 9,2005 (54) METHOD AND APPARATUS FOR SELECTIVELY DENYING ACCESS TO ENCODED DATA. (75) Inventors: Larry A. Lee, Vestal, NY (US); Robert L. Kilmer, Jr., Endicott, NY (US); David R. Menigoz, Apalachin, NY (US) (73) Assignee: Lockheed Martin Corporation, Bethesda, MD (US)

Blakely1997 US005677952A [ii] Patent Number: 5,677,952 [45] Date of Patent: Oct. 14, 1997. [54] METHOD TO PROTECT INFORMATION ON A COMPUTER STORAGE DEVICE. [75] Inventors: George R. Blakley, ID. Austin. Tex.; Phillip W. Rogaway, Davis, Calif. [73] Assignee: International Business Machines Corporation. Austin, Tex.

Dodis2003 Y. Dodis and M. Yung, "Exposure-Resilience for Free: The Hierachical ID-based Encryption Case," *Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)*, 2003.

Yasukawa1999 Yasukawa, H., & Kurosawa, T. (Dec. 7th, 1999). U.S. Patent No. 5999622. Washington, DC: U.S. Patent and Trademark Office.

Kihara US006298014B1. United States Patent, Kihara et al. Patent No.: US 6,298,014 Bl. Date of Patent: Oct. 2,2001. TIME INFORMATION MANAGEMENT SYSTEM. Inventors: Hiroyuki Kihara; Toshio Umemoto; Tomomi Murakami; Masahiro Sase, all of Tokyo (JP). Assignee: Citizen Watch Co., Ltd., Tokyo (JP).

Border2009 Private communication with Dr. Charles Border of RIT on July 2, 2009.

Siegel95 United States Patent. Siegel US005467463A [ii] Patent Number: Date of Patent: 5,467,463 Nov. 14, 1995. SYSTEM INDEPENDENT TIMING REFERENCE FOR COMPUTER. Inventor: Mark D. Siegel, Fort Worth, Tex. [73] Assignee: Tandy Corporation, Fort Worth, Tex.

Kuczynski US 20070058493A1. United States. Patent Application Publication Go) Pub. No.: US 2007/0058493 Al Kuczynski et al. (43) Pub. Date: Mar. 15,2007. METHODS AND APPARATUS CAPABLE OF INDICATING ELAPSED TIME INTERVALS. Inventors: Joseph Kuczynski, Rochester, MN (US); David Otto Lewis, Rochester, MN (US). Correspondence Address: IBM CORPORATION, ROCHESTER IP LAW DEPT. 917

Blanchet2008 B. Blanchet and A. Chaudhuri, "Automated Formal Analysis of a Protocol for Secure File Sharing on Untrusted Storage," *IEEE Symposium on Security and Privacy*, pp. 417-431, 2008.

MayWoods May, T.C. and Woods, M.H. , "Alpha-particle-induced soft errors in dynamic memories", IEEE Transactions on Electron Devices, Jan 1979, Volume 26, Issue 1, pp. 2-9.

Ziegler79 J. F. Ziegler and W. A. Lanford, "The effect of cosmic rays on computer memories," Sci., vol. 206, p. 776, 1979.

Phillips02 G. W. Phillips, R. A. August, A. B. Campbell, M. E. Nelson, N. A. Guardala, J. L. Price and M. Moscovitch, "Feasibility of a Neutron Detector-dosemeter based on Single-event Upsets in Dynamic Random-access Memories," Radiation Protection Dosimetry, Volume 101, pp. 129-132, 2002.

Geambasu2009 Geambasu, R., Kohno, T., Levy, A. A., & Levy, H. M. (2009). Vanish: Increasing Data Privacy with Self-Destructing Data. Retrieved July 23, 2009, from http://vanish.cs.washington.edu/