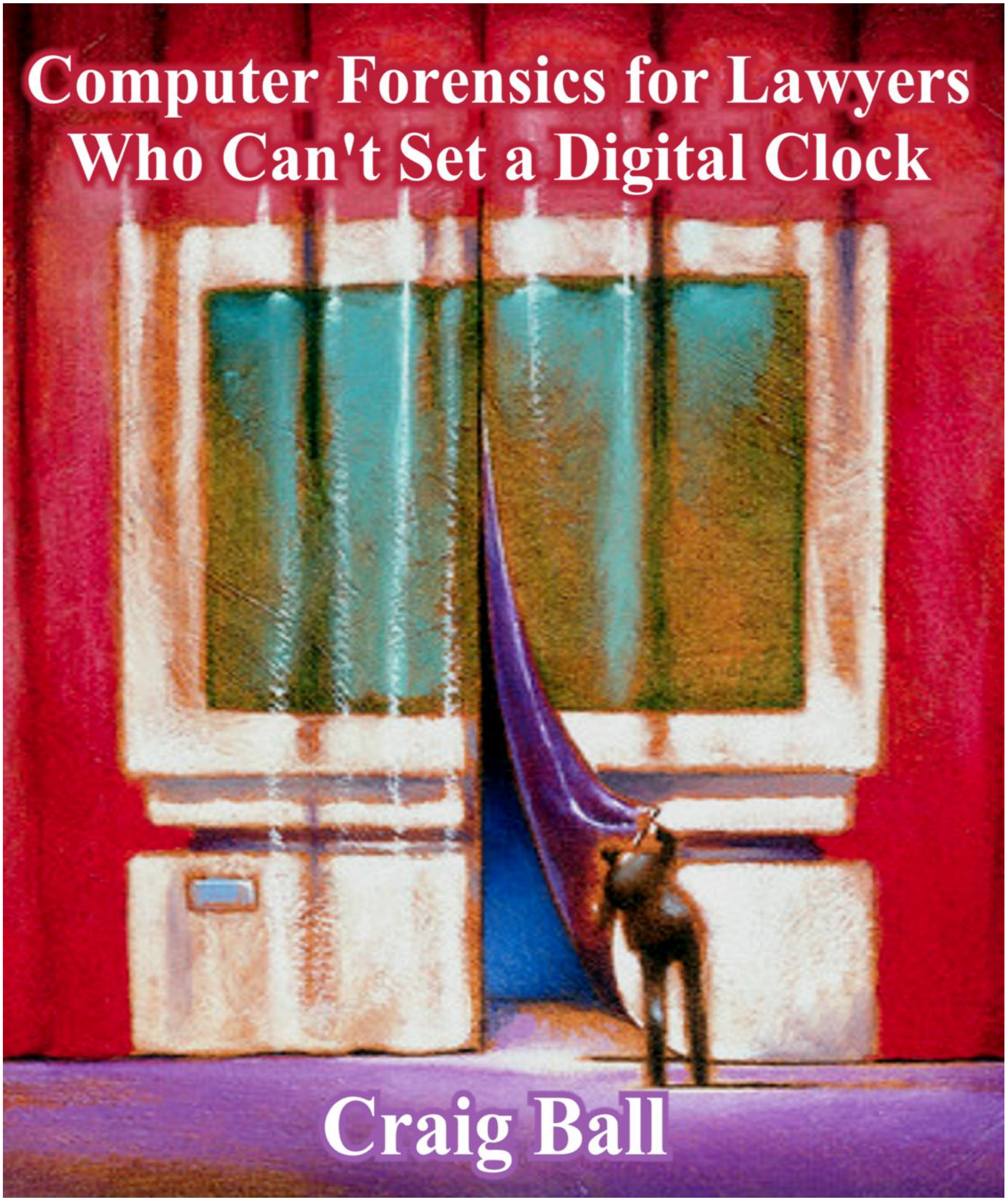

Computer Forensics for Lawyers Who Can't Set a Digital Clock
Craig Ball

Computer Forensics for Lawyers Who Can't Set a Digital Clock

Table of Contents

Note to Readers:

This article focuses on technical matters impacting the cost, complexity and scope of e-discovery, rather than the burgeoning case law. For extensive resources on electronic discovery law, please look at other materials available at www.craigball.com and visit the following helpful sites:

K&L Gates Electronic Discovery Law Site
http://www.ediscoverylaw.com/
Discovery Resources
http://discoveryresources.org/
Electronic Discovery Reference Model
http://www.edrm.net

For extensive links to further information about computer forensics, visit:
The Electronic Evidence Information Center
http://www.e-evidence.info/index.html

Computer Forensics for Lawyers Who Can't Set a Digital Clock

"When you go looking for something specific, your chances of finding it are very bad. Because of all the things in the world, you're only looking for one of them. When you go looking for anything at all, your chances of finding it are very good. Because of all the things in the world, you're sure to find some of them."
Movie Detective Daryl Zero, from the film "The Zero Effect"

The Smoking Gun

Lawyers love the smoking gun. We adore the study that shows it's cheaper to pay off the burn victim than fix the flawed fuel system, the directive that staff needs to work all night to implement the new document "retention" policy, the employment review with the racist remark and the letter between competitors agreeing to "respect" each other's pricing. Each case has its smoking gun. It may be a peashooter with the faintest whiff of cordite or a Howitzer with a red-hot muzzle, but it's there *somewhere*. Searching for the smoking gun once meant poring over great forests felled, turned to oceans of paper captured in folders, boxes, cabinets, rooms and warehouses. Today, fewer and fewer business communications and records find their way into paper form, so your smoking gun is likely smoking on someone's hard drive.

What's more, not only is the smoking gun more likely to be stored electronically, the informal and immediate nature of electronic communications makes them more likely to be smoking guns. People aren't as guarded in what they say via e-mail as when writing a letter. Electronic communication is so frictionless that a damning e-mail is just an improvident click away from dozens or hundreds or thousands of in boxes. Think also of the ease of digitally distributing attachments that would have consumed hours at a copier to send on paper.

Consider also the volume of electronic communications. On a given day, I might send out fifty to one hundred individual e-mails, but it's unlikely I've drafted and sent that many letters in any day of my entire career as an attorney. Put another way, I'm about fifty times more likely to put my foot in my mouth electronically than on paper. This is the norm in American business.

What You Don't Know *Can* Hurt You

Although lawyers are coming to appreciate that the smoking gun they seek may not be on paper, a pervasive lack of knowledge about electronic data, coupled with experience grounded exclusively on paper discovery, makes it hard for lawyers and judges to meet the challenge of digital data discovery.

Ten years ago, In a case involving a dispute over privileged documents on a shared laptop computer, the parties entered into an agreed order respecting the data on the computer, and the court appointed me as Special Master to carry out the tasks ordered. The instructions I received were simple…and daunting. Among other tasks, I was to reduce all "documents" on the computer to written form, including all scans, program files, deleted records and data from Internet surfing. Using round numbers, the hard drive in question had some ten gigabytes of data spread across 18,000 files. The way the assignment was structured, each file constituted a document and file sizes ran the gamut from virtually nothing to massive programs. Because of the sensitive nature of the information, I was expected to personally handle all aspects of the task, including monitoring the printing.

Estimates of how digital data convert to printed pages are notoriously misleading because of the wide variance in how applications format the printed page: a tiny Word file can consume dozens of printed pages while a large graphic file may result in a small image. However, a commonly cited estimate suggests the following correlation:

Data	**Printed Pages**
One megabyte	= 1,000-1,400
One gigabyte	= 100,000-140,000
One terabyte	=100,000,000-140,000,000

By this measure, the ten gigabytes of data on the hard drive would print out to something over a million pages, and I could get the job done in under a year of forty-hour weeks, chained to the printer. Problem was, even if I were willing to abandon my practice and babysit a printer, the files were not formatted so as to efficiently fill the printed pages (because they were data files and not designed to be printed). Instead, I was probably looking at several million printed pages, the vast majority of them containing meaningless strings of gibberish. Did I mention I'd have to make three copy sets? The paper and toner alone would cost $120,000, not to mention the printers and Prozac. It was ridiculous.

Clearly, a global order that the contents of a computer be printed out is a disaster. The solution in this case was to revise the order to permit production of the data in its native electronic format and to eliminate the production of software applications and other data that did not, in any manner, reflect activities by users of the computer. This is a much more time- and cost-efficient technique, and it spared a couple of acres of forest to boot.

A Little Knowledge is a Wonderful Thing

Errors like the potentially costly one just described can be avoided in the first place if lawyers gain a fundamental understanding of how a computer stores data and the many nooks and crannies where data can hide despite efforts to make it disappear. This knowledge is valuable whether you are combing an employee's computer to find out if they have engaged in on-the-job shenanigans with firm property or framing discovery requests; but be advised that it is no substitute for the services of a qualified and experienced computer forensics expert. If you don't know what you are doing, your efforts to resurrect deleted data may end up permanently deleting the smoking gun or, at the very least, imperiling its admissibility in court.

Reading this article isn't going to make you a computer forensics expert. Many topics are oversimplified or explained with metaphors that would make a computer engineer wince, but you will get enough of the basics to impress opposing counsel and make yourself wholly unattractive to members of the opposite sex. You might even find yourself casting admiring glances at short sleeve shirts and vinyl pocket protectors.

This article will focus on the WinTel platform (geek speak for a computer with an Intel processor running the Microsoft Windows operating system, though the material covered applies equally to Windows computers running AMD chips). All of the concepts and many of the specifics apply to other computing environments as well.

Magnetic Storage

A variety of technologies have to come together to create a computer, but the most important of these with respect to forensics has to be magnetic storage. Nearly all of the smoking gun data you seek to discover or shield from disclosure takes the forms of trillions upon trillions of faint and impossibly tiny magnetic charges that coat the surface of a rapidly spinning disc. A Lilliputian device, called a *read/write head,* interacts with these particles, imparting a magnetic charge or reading a charge already there. No matter what form information takes when it goes into a computer—video, sound, word, number, or photograph—it is all stored magnetically in a sequence of magnetic polarity changes customarily represented by ones and zeros. These "on" and "off" states are like the Morse code used by telegraphers one hundred fifty years ago, but now transmitted so quickly that an encyclopedia of information can be communicated in seconds.

It's Time

Can a lawyer be a damn good litigator without knowing much about the inner workings of a computer? Ten years ago, the answer would have been, "sure;" but we've reached the point where not understanding computer forensics and not having digital discovery skills is no laughing matter. It's a ticking time bomb in your practice. You *know* how important discovery is to winning your case. You know the value of the smoking gun document, the doctored record, and the too-candid memo. Products liability cases, wrongful discharge claims and antitrust actions, just to name a few, are won and lost in discovery. Try this fact on for size:

More than Ninety-five percent **of the world's information is being generated and stored in digital form and few business documents created today *(less than one-tenth of one percent)* ever become paper records. They *never* get printed out. They *never* leave the digital domain. Most *never* find their way into the printed material produced to you in discovery.**

Now ponder these questions:

Are you willing to accept an assurance of "we didn't find anything" from the other side when you know they haven't looked everywhere and they don't know how to find what they are supposed to be looking for?

Can you effectively cross-examine a computer expert if you know almost nothing about their area of expertise? How will you know when they are wrong? How can you expose their weaknesses?

Are you content to have to hire an expert in every case where computer records are at issue? And isn't that almost every case nowadays?

If the answer to any of these questions is "no," it's time to stop leaving the geek stuff to the geeks. It's time to learn the basics of computer forensics.

How Much Information?
The world produces between 1 and 2 exabytes of unique information per year, which is roughly 250 megabytes for every man, woman, and child on earth. An exabyte is a billion gigabytes, or 10^{18} bytes, *equivalent to the textual content of a trillion books*. Printed documents of all kinds comprise only .003% of the total. Magnetic storage is by far the largest medium for storing information and is the most rapidly growing, with shipped hard drive capacity *doubling every year.*

Single hard drives now hold three gigabytes of data and sell for as little as five cents per gigabyte. By way of comparison, if the automobile industry were as efficient, you could buy a new car for less than you paid for your last haircut!

Computer Forensics
Computer forensics is the identification, preservation, extraction, interpretation and presentation of computer-related evidence. It sounds like something anyone who knows his way around a computer might be able to do, and in fact, many who offer their services as computer forensic specialists have no formal forensic training or certification--which is not to say they can't do the job well, but it certainly makes it hard to be confident they can! There are compelling reasons to hire a formally trained and experienced computer forensic specialist. Far more information is retained by a computer than most people realize, and without using the right tools and techniques to preserve, examine and extract data, you run the risk of losing something important, rendering what you do find inadmissible, or even being charged with spoliation of the evidence.

The cardinal rules of computer forensics can be expressed as the five **A**s:

1. **A**dmissibility must guide actions: document everything that is done;

2. **A**cquire the evidence without altering or damaging the original;

3. **A**uthenticate your copy to be certain it is identical to the source data;

4. **A**nalyze the data while retaining its integrity; and,

5. **A**nticipate the unexpected.

These cardinal rules are designed to facilitate a forensically sound examination of computer media and enable a forensic examiner to testify in court as to their handling of a particular piece of evidence. A forensically sound examination is conducted under controlled conditions, such that it is fully documented, replicable and verifiable. A forensically sound methodology changes no data on the original evidence, preserving it in pristine condition. The results must be replicable such that any qualified expert who completes an examination of the media employing the same tools and methods employed will secure the same results.

After reading this paper, you may know enough of the basics of computer forensics to conduct a rudimentary investigation; but recognize that conducting a computer forensic investigation without the assistance of a qualified expert is a terrible idea. Experiment on an old system if you'd like, but leave real evidence to the experts.

Computer forensics focuses on three categories of data:

Active Data: These are the current files on the computer, still visible in directories and available to applications. Active data may be readily comprehensible using simple translation techniques (i.e., plain text files), but will more often need to be viewed within an application (computer program) to be useful. Such applications range from e-mail clients like Outlook, to database programs like Access or Excel, to word processors like Word or WordPerfect. Active data may also be password protected or encrypted, requiring further forensic activity to be accessed. Active data includes system data residing within the recycle bin, history files, temporary Internet directory, cookie "jar," system registry files, logs and other obscure but oft-revealing data caches. One important evidentiary point about data on a hard drive is that no matter what it may represent, whether simple text or convoluted spreadsheets, it exists only as infinitesimal magnetic flux reversals representing ones and zeroes *which must be processed by software to be intelligible*. Put another way, only the physical level with the magnetic domains is real; this level is also the least accessible. Words, pages, files, and directories are abstractions—illusions if you prefer--created by software that may or may not be reliable. The more levels of abstraction, the more likely evidence will not be, and should not be, admitted without scrutiny.

Latent Data: Latent data (also called "**ambient data**") are deleted files and other data, including memory "dumps" that have "lodged in the digital cracks" but can still be retrieved. This data resides on the hard drive or other storage media in, e.g., unallocated clusters (areas marked available for data storage but not yet overwritten by other data) and slack space. Latent data also includes information not readily understood absent special techniques and tools, like swap files, temporary files, printer spool files, metadata and shadow data (all discussed herein). The recovery of latent data is the art most often associated with computer forensics, but the identification, preservation, interpretation and management of active data is no less demanding of a forensic expert's skill.

Archival Data: This is data that's been transferred or backed up to peripheral media, like tapes, CDs, DVDs, ZIP disks, floppy disks, network servers or the Internet. Archival data can be staggeringly voluminous, particularly in a large organization employing frequent, regular back up procedures. It is critically important to recognize that an archival record of a source media never reflects all of the data that can be identified and extracted from the source media because such backups don't carry forward latent data. Accordingly, an opponent's offer to furnish copies of backup tapes is, while valuable, no substitute for a forensic examination of a true bit-by-bit copy of the source disk drive.

"And just what was that little window you clicked off when I came in?"

© Cartoonbank.com

Tell It to the Judge
Imagine that a case comes in where the content of a personal computer is critically important. Perhaps your client's marriage is on the rocks and infidelity and hidden assets are at issue. If you represent the wife, do you think that the philandering husband is going to agree to make his personal computer available to you; handing over the chat room transcripts, cyber-sex sessions, incriminating e-mails, Quicken balances, Internet history files, brokerage account records, digital photographs of the fluff on the side, business trip expense records, overseas account passwords and business correspondence? Chances are Hubby is going to fight you tooth and nail and, when finally ordered to make the machine available, he will clumsily seek to delete anything deemed compromising. But even if Hubby isn't trying to cover his tracks, know that every time he saves a file, or starts a program—*in fact every time he simply boots the machine*—some latent data is altered or overwritten to the point it can never be retrieved. By way of example, Windows accesses (and thus modifies metadata for) about a thousand files every time it boots up (and you wondered why booting took so long)!

You must persuade the court that conventional paper discovery is inadequate and that your client's interests will be irreparably harmed if she isn't granted access to Hubby's computer and afforded the right to conduct a complete forensic examination of same, starting with the creation of a sector-by-sector bit stream copy of the hard drive. Because Hubby has hired a savvy advocate, the judge is being assured that all reasonable steps have been taken to identify and protect computer data and that print outs of discoverable material will be furnished, subject to claims of privilege and other objections. If you can't articulate why your opponent's proposal is hogwash and thoroughly educate the judge about the existence and ongoing destruction of latent data, Missus is out-of-luck.

To be prepared to educate the Court, evaluate and select a computer forensics effort or simply better understand and advise your clients about "safe" data practices, you need a working knowledge of how a computer stores data and, more to the point, where and how data lives on after it's supposed to be gone.

To get that working knowledge, this section explains (as simply and painlessly as possible) the nuts and bolts of computer storage, beginning with the bits and bytes that are the argot of all digital computing, then on to the mechanics of hard drive operation and finally to the nooks and crannies where data hides when it doesn't want to be dispatched to that big CPU in the sky.

Bits and Bytes
You can become very facile with computers never knowing the nitty-gritty about bits and bytes, but when it comes to building a fundamental understanding of computer forensics, you've got to begin with the building blocks of computer data: bits and bytes. You know something of bits and bytes because every computer ad you've seen uses them in some impressive-sounding way. The capacity of computer memory (RAM), size of computer storage (disks), and the data throughput speed of modems and networks are all customarily expressed in bits and bytes.

This Little Piggy went to Market
When we were children starting to count, we had to *learn* the decimal system. We had to *think* about what numbers *meant*. When our first grade selves tackled a big number like 9,465, we

were acutely aware that each digit represented a decimal multiple. The nine was in the thousands place, the four in the hundreds, the six in the tens place and so on. We might even have parsed 9,465 as: *(9 x 1000) + (4 x 100) + (6 x 10) + (5 x 1)*.

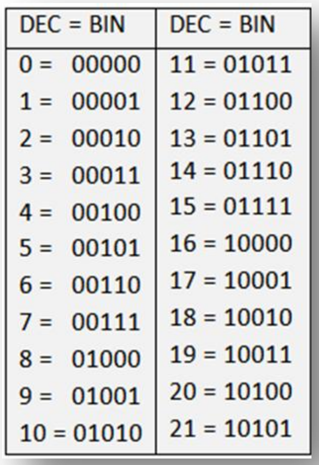
But soon, it became second nature to us. We'd unconsciously process 9,465 as nine thousand four hundred sixty-five. As we matured, we learned about powers of ten and now saw 9,465 as: *(9 x 10^3) + (4 x 10^2) + (6 x 10^1) + (5 x 10^0).* This was exponential or "base-ten" notation. We flushed it from our adolescent brains as fast as life (and the SAT) allowed.

Mankind probably uses base ten to count because we evolved with ten fingers. But, had we slithered from the ooze with eight or twelve digits, we'd have gotten on splendidly using a base eight or base twelve number system. It really wouldn't matter because any number--and consequently any data--can be expressed in any number system. So, it happens that computers use the base two or binary system, and computer programmers are partial to base sixteen or hexadecimal. It's all just counting.

A Bit about the Bit

Computers use **binary digits** in place of decimal digits. The word **bit** is even a shortening of the words "Binary digIT." Unlike the decimal system, where any number is represented by some combination of ten possible digits (0-9), the bit has only two possible values: zero or one. This is not as limiting as one might expect when you consider that a digital circuit—essentially an unfathomably complex array of switches—hasn't got any fingers to count on, but is very good and very fast at being "on" or "off."

In the binary system, each binary digit—"bit"—holds the value of a power of two. Therefore, a binary number is composed of only zeroes and ones, like this: 10101. How do you figure out what the value of the binary number 10101 is? You do it in the same way we did it above for 9,465, but you use a base of 2 instead of a base of 10. Hence: (1 x 2^4) + (0 x 2^3) + (1 x 2^2) + (0 x 2^1) + (1 x 2^0) = 16 + 0 + 4 + 0 + 1 = **21**.



DEC = BIN	DEC = BIN
0 = 00000	11 = 01011
1 = 00001	12 = 01100
2 = 00010	13 = 01101
3 = 00011	14 = 01110
4 = 00100	15 = 01111
5 = 00101	16 = 10000
6 = 00110	17 = 10001
7 = 00111	18 = 10010
8 = 01000	19 = 10011
9 = 01001	20 = 10100
10 = 01010	21 = 10101

Moving from right to left, each bit you encounter represents the value of increasing powers of 2, standing in for zero, two, four, eight, sixteen, thirty-two, sixty-four and so on. That makes counting in binary pretty easy. Starting at zero and going through 21, decimal and binary equivalents look like the table at right.

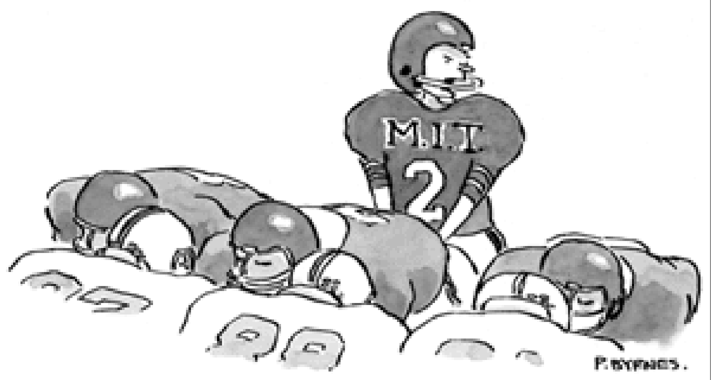
Bytes

A byte is a string of eight bits. The biggest number that can be stored as one byte of information is 11111111, equal to 255 in the decimal system. The smallest number is zero or 00000000. Thus, there are 256 different numbers that can be stored as one byte of information. So, what do you do if you need to store a number larger than 256? Simple! You use a second byte. This affords you all the combinations that can be achieved with 16 bits, being the product of all the variations of the first byte and all of the second byte (256 x 256 or 65,536). So, using bytes to express values, any number that is greater than 256 needs at least two bytes to be expressed (called a "word" in geek speak), and any number above 65,536 requires at least three. A value greater than 16,777,216 (256^3, exactly the same as 2^{24}) needs four bytes (called a "long word") and so on.

Let's try it: Suppose we want to represent the number 51,975. It's 1100101100000111, *viz*:

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
32768	16384	8192	4096	2048	1024	512	256		128	64	32	16	8	4	2	1
1	**1**	**0**	**0**	**1**	**0**	**1**	**1**	**+**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**
(32768+16384+2048+512+256) or **51,968**								**+**	(4+2+1) or **7**							

Why is an eight-bit string the fundamental building block of computing? It just sort of happened that way. In this time of cheap memory, expansive storage and lightning-fast processors, it's easy to forget how scarce and costly these resources were at the dawn of the computing era. Seven bits (with a bit reserved) was basically the smallest block of data that would suffice to represent the minimum complement of alphabetic characters, decimal digits, punctuation and control instructions needed by the pioneers in computer engineering. It was, in another sense, about all the data early processors could chew on at a time, perhaps explaining the name "byte" (coined by IBM scientist, Dr. Werner Buchholz, in 1956).



"01101001, 00111011, 00011010, but, but!"

ASCII

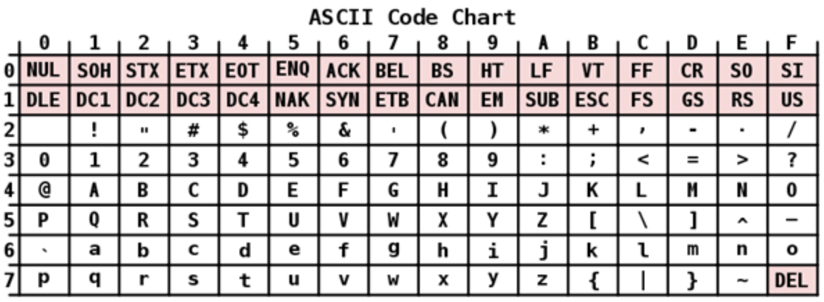
Computers need to work with text as well as numbers, so computers use binary numbers to stand for the upper and lower case English alphabet, as well as punctuation marks and special characters. The most widely deployed U.S. encoding mechanism is known as the **ASCII** code (for **American Standard Code for Information Interchange,** pronounced "ask-key"). By limiting the ASCII character set to just 128 characters, any character can be expressed in just seven bits (2^7) and so occupies less than one byte in the computer's storage and memory. The eighth bit in the byte was tasked to do other work (parity) in the earliest version of ASCII.

Hex

Long sequences of ones and zeroes are very confusing for people, so **hexadecimal notation** emerged as an accessible shorthand for binary sequences. Considering the prior discussion of base-ten (decimal) and base-two (binary) notation, it might be sufficient just to say that hexadecimal is base-sixteen. In hexadecimal notation (hex for short), each digit can be any value from zero to fifteen. Accordingly, four binary digits can be replaced by just one hexadecimal digit, and more to the point, a byte can be expressed as just two hex characters.

The decimal system supplies only 10 symbols (0-9) to represent numbers. Hexadecimal characters need 16, leaving us without enough single character, numeric values to stand in for all the values in each column. How will we cram 16 values into each column? The solution was to substitute the letters A through F for the numbers 10 through 15. So, we can represent 10110101 (the decimal number 181) as "B5" in hexadecimal notation.
It's hard to tell if a number is decimal or hexadecimal just by looking at it: if you see "37", does that mean 37 ("37" in decimal) or 55 ("37" in hexadecimal)? To get around this problem, two common notations are used to indicate hexadecimal numbers. The first is the suffix of a lower-case "h". The second is the prefix of "0x". So "B5 in hexadecimal", "B5h" and "0xB5" all mean the same thing.

The ASCII Code Chart at right can be used to express ASCII text in hex. The capital letter "T" has the hex value of 54 (i.e., row 5, column 4), so "Keep Austin Weird" hex encodes as 4B 65 65 70 20 41 75 73 74 69 6E 20 57 65 69 72 64.



ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

ASCII was introduced in the pre-Internet world of 1963. That was back before the world was flat—an era when the West dominated commerce and personal computing was the stuff of science fiction. Because a seven bit byte (septet) encoding scheme like ASCII can represent only 128 characters (2^7), it couldn't encode languages employing different characters sets like Cyrillic, Hebrew, Arabic or Greek. Extending the ASCII sets to an eight bit (octet byte) scheme--also called the ANSI or LATIN-1 character set--supported only 256 (2^8) characters, unsuited to East Asian languages, like Chinese, Japanese and Korean, which employ thousands of pictograms and ideograms.

Though various *ad hoc* approaches to foreign language encodings were developed, a universal, systematic encoding mechanism was needed to serve an increasingly interconnected world. These methods used more than one byte to represent each character. The most widely adopted such system is called **Unicode**. In its latest incarnation (version 8), Unicode standardizes the encoding of 93 written languages comprising 109,449 characters.

Unicode was designed to co-exist with the longstanding ASCII and ANSI character sets by emulating the ASCII character set in corresponding byte values within the more extensible Unicode counterpart, **UTF-8**. Because of its backward compatibility and multilingual adaptability, UTF-8 has become a widely-used encoding standard, especially on the Internet and within e-mail systems.

Now it may seem that you've asked for the time and been told the history of clock making, but computer forensics is all about recorded data, and <u>all</u> computer data exists as bits and bytes. What's more, you can't tear open a computer's hard drive and find tiny strings of ones and zeros written on the disk, let alone words and pictures. The billions of bits and bytes on the hard drive exist only as faint vestiges of magnetism, microscopic in size and entirely invisible. It's down here--way, way down where a dust mote is the size of Everest and a human hair looks like a giant sequoia--where all the fun begins.

Information Storage

We store information by translating it into a physical manifestation: cave drawings, Gutenberg bibles, musical notes, Braille dots or undulating grooves in a phonograph record. Because binary data is no more than a long, long sequence of ones and zeros, it can be recorded by any number of alternate physical phenomena. You could build a computer that stored data as a row of beads (the abacus), holes punched in paper (a piano roll), black and white vertical lines (bar codes) or bottles of beer on the wall (still waiting for this one!).

"I should have had him put into a more manageable format years ago."

But if we build our computer to store data using bottles of beer on the wall, we'd better be plenty thirsty because we will need something like 99,999,999 bottles of beer to get up and running. And we will need a whole lot of time to set those bottles up, count them and replace them as data changes. Oh, and we will need something like the Great Wall of China to set them on. Needless to say, despite the impressive efforts ongoing at major universities and bowling alleys to assemble the raw materials, our beer bottle data storage system isn't very practical. Instead, we need something compact, lightweight and efficient—a leading edge technology--in short, a refrigerator magnet.

Magnetic Storage

Okay, maybe not a refrigerator magnet exactly, but the principles are the same. If you take a magnet off your refrigerator and rub it a few times against a metal paperclip, you will transfer some magnetic properties to the paperclip. Suppose you lined up about a zillion paper clips and magnetized some but not others. You could go down the row with a piece of ferrous metal (or, better yet, a compass) and distinguish the magnetized clips from the non-magnetized clips. Chances are this can be done with less space and energy than beer bottles, and if you call the magnetized clips "ones" and the non-magnetized clips "zeroes," you've got yourself a system that can record binary data. Were you to glue all those paper clips in concentric circles onto a spinning phonograph record and substitute an electromagnet for the refrigerator magnet, you wouldn't be too far afield of what goes on inside the hard and floppy disk drives of a computer, albeit at a much smaller scale. In case you wondered, this is also how we record sounds on magnetic tape, except that instead of just determining that a spot on the tape is magnetized or not as it rolls by, we gauge varying degrees of magnetism which corresponding to variations in the recorded sounds. This is called **analog** recording—the variations in the recording are analogous to the variations in the music.

Since computers process electrical signals much more effectively than magnetized paper clips jumping onto a knife blade, what is needed is a device that transforms magnetic signals to electrical signals and vice-versa—an energy converter. Inside every floppy and hard disk drive is a gadget called a disk head or read/write head. The read/write heads are in essence tiny electromagnets that perform this conversion from electrical information to magnetic and back again. Each bit of data is written to the disk using an encoding method that translates zeros and ones into patterns of magnetic flux reversals. Don't be put off by Star Wars lingo like "magnetic flux reversal"--it just means flipping the magnet around to the other side or "pole."

Older hard disk heads make use of the two main principles of electromagnetic force. The first is that applying an electrical current through a coil produces a magnetic field; this is used when writing to the disk. The direction of the magnetic field produced depends on the direction that the current is flowing through the coil. The converse principle is that applying a magnetic field to a coil will cause an electrical current to flow. This is used when reading back previously written information. Newer disk heads use different physics and are more efficient, but the basic approach hasn't changed: electricity to magnetism and magnetism to electricity.

Fantastic Voyage

Other than computer chip fabrication, there's probably no technology that has moved forward as rapidly or with such stunning success as the hard disk drive. Increases in capacity and reliability, precision tolerances and reduction in cost per megabyte all defy description without superlatives. The same changes account for the emergence of electronic media as the predominant medium for information storage (it's big—it's cheap—it's reliable), with commensurate implications and complications for civil discovery.

Since you now understand the form of the information being stored and something of the physical principles underlying that storage, it's time to get inside the hard drive and draw closer to appreciating where and why data can be deleted but still hang around. In 1966, Hollywood gave us the movie "Fantastic Voyage" about a group of scientists in a submarine shrunken down to microscopic dimensions and injected into the bloodstream. A generation later, the Magic School Bus along with the Honey I Shrunk the Kids series offered similar fantastic voyages. Let's follow the lead of Raquel Welch, Mrs. Frizzle and Wayne Szalinski and descend deep within the inner workings of a hard drive.

Caveat: At this point, we start talking about the innards of a personal computer. Should you be tempted to actually open one up and monkey around inside, please be advised that there is a significant risk of damage to the computer, your data and, most importantly, *to you*. Before you open the case of any PC, pull the plug and disconnect <u>all</u> cables, especially the power, modem, monitor and printer cables. Resist all temptation to poke around inside the power supply. There's little worth seeing in there and you can electrocute yourself. Seriously! If you experiment on a hard drive, be sure it contains no data that you care to retain. Note also that the technical term for a hard drive that has been opened up is "toast."

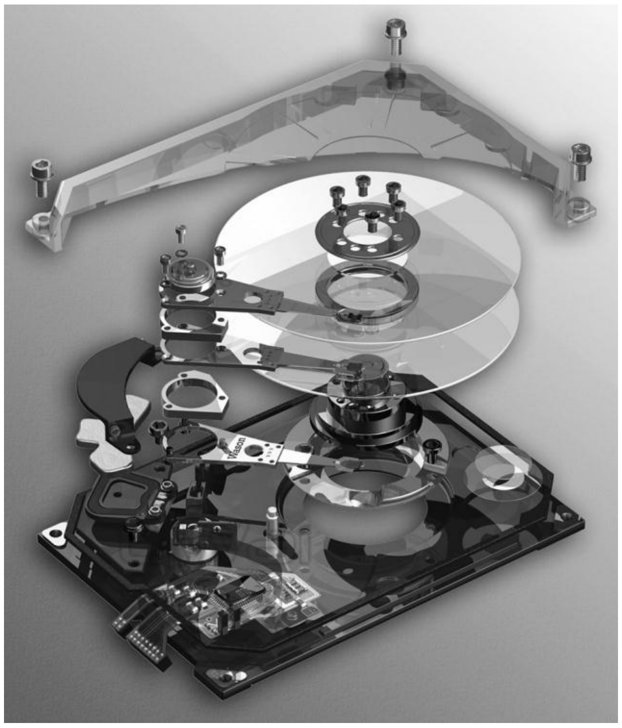
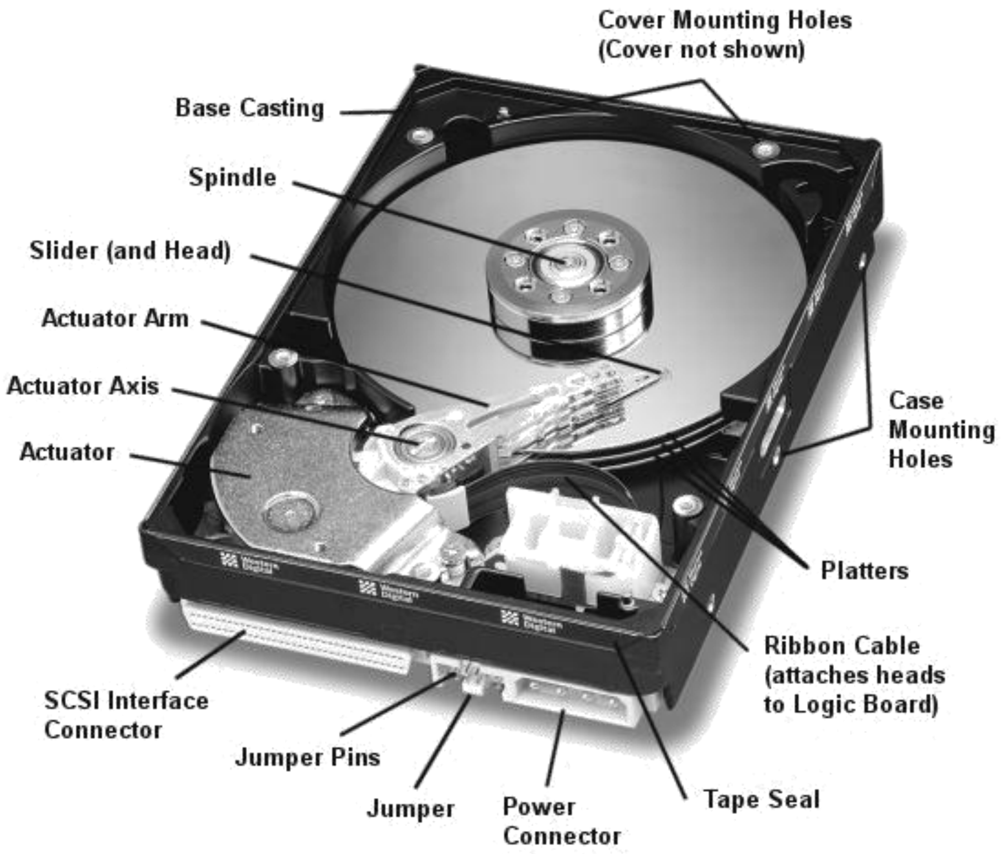


Figure 1

(Above) This is an exploded view of a typical personal computer hard drive.
The splendid illustration above is the work of Griff Wason.
Note the stack of discs (platters) and the ganged read/write heads.
(Below) A photo of a hard drive's interior with cover removed.



Disc Anatomy 101

A personal computer hard drive is a sealed aluminum box measuring (for a desktop system) roughly 4" x 6" x 1" in height. Though often mounted above or below the optical (CD/DVD) drives, it is not uncommon to encounter the hard drive located almost anywhere within the case, customarily secured by several screws attached to any of six or more pre-threaded mounting holes along the edges of the case. One face of the case will be labeled to reflect the drive specifications as in Fig. 2, while a printed circuit board containing logic and controller circuits will cover the opposite face (shown removed in Fig. 3).

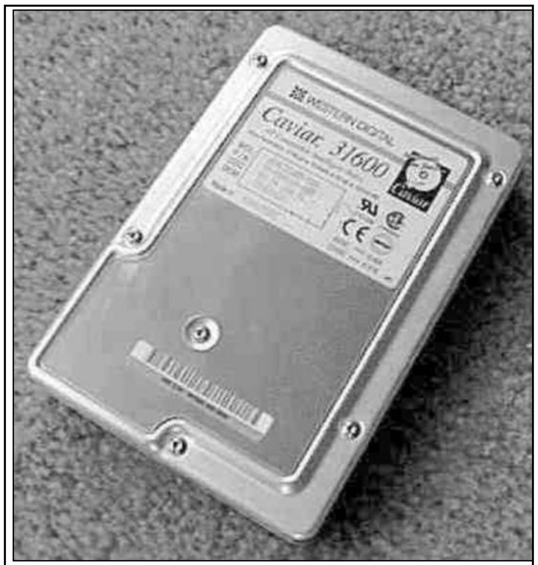


Figure 2

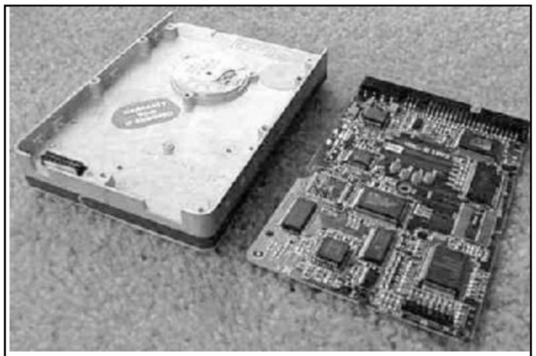


Figure 3

Hard disk drives principally use one of five common interfaces: IDE/ATA, SCSI and SATA. You can tell immediately by looking at the back of the hard disk which interface is being used by the drive:

- PATA (Parallel ATA, sometimes called EIDE for Extended Integrated Drive Electronics): A 40-pin rectangular connector (Figs. 4 and 5).
- SATA (Serial ATA): A 7-pin flat connector, less than a third the size of its IDE counterpart (Fig 5)
- SCSI **(**Small Computer System Interface): A 50-pin, 68-pin, or 80-pin D-shaped connector (see fig. 1).
- SAS for Serial Attached SCSI
- FC for Fibre Channel

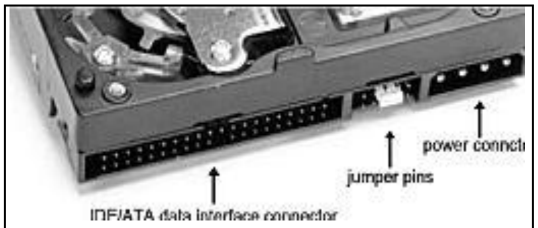


Figure 4

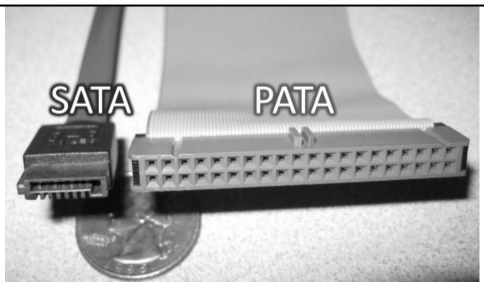


Figure 5

A hard disk contains round, flat discs called **platters**, coated on both sides with a special material able to store data as magnetic patterns. Much like a record player, the platters have a hole in the center allowing them to be stacked on a spindle. The platters rotate at high speed—typically 5,400, 7,200 or 10,000 rotations per minute--driven by a special motor. The read/write heads are mounted onto sliders and used to write data to the disk or read data from it. The sliders are, in turn, attached to arms, all of which are joined as a single assembly oddly reminiscent of a record player's tone arm and steered across the surface of the disk by a device called an actuator. (Fig. 6). Each platter has two heads, one on the top of the platter and one on the bottom, so a hard disk with three platters (normally) has six surfaces and six total heads.

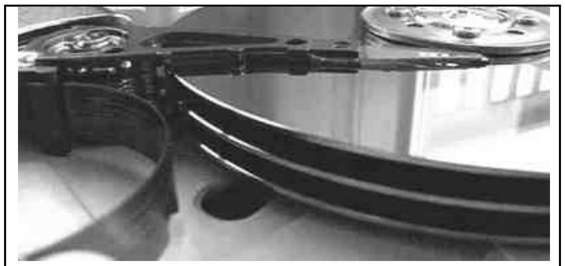
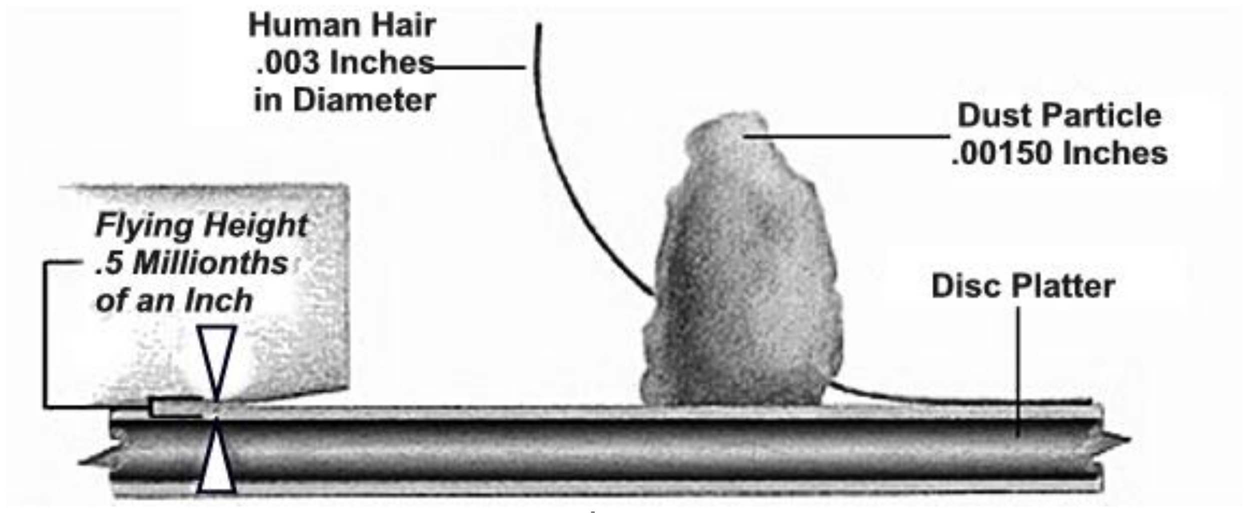


Figure 6

When the discs spin up to operating speed, the rapid rotation causes air to flow under the sliders and lift them off the surface of the disk--the same principle of lift that operates on aircraft wings and enables them to fly. The head then reads the flux patterns on the disc while flying just *.5 millionths of an inch* above the surface. At this speed, if the head bounces against the surface, there is a good chance that the heads or sliders would burrow into the media, obliterating data and frequently rendering the hard drive inoperable ("head crash"). Surprisingly, head crashes are increasingly rare events even as the tolerances have become more exacting. To appreciate the fantastic tolerances required for achieving this miracle, consider Fig. 7. A human hair is some *6,000 times thicker* than the flying height of a modern hard drive read/write head! No wonder hard drives must be assembled in "clean rooms" with specially filtered air supplies.



Perspective: Woody Monroy, head of corporate communications for hard drive maker Seagate Technology, L.L.C., points out that, in terms of speed and tolerances, a hard drive's operation is equivalent to *an F-16 jet fighter plane flying at 813 times the speed of sound and one-sixty second of an inch off the ground…while counting every blade of grass as it goes!*

Sectors, and Clusters and Tracks, Oh My!
Now it starts to get a little complicated, but stay with me because we've nearly unraveled the mystery of latent data. At the factory, platters are organized into specific structures to enable the organized storage and retrieval of data. This is **low level formatting,** dividing each platter into tens of thousands of densely packed concentric circles called **tracks**. If you could see them (and you can't because they are nothing more than microscopic magnetic traces), they might resemble the growth rings of the world's oldest tree. It's tempting to compare platter tracks to a phonograph record, but you can't because a phonograph record's track is a single spiraling groove, not concentric circles. A track holds far too much information to serve as the smallest unit of storage on a disk, so each one is further broken down into **sectors**. A sector is normally the smallest individually addressable unit of information stored on a hard disk, and holds **512 bytes** of information. The first PC hard disks typically held 17 sectors per track. Figure 8 shows a very simplified representation of a platter divided into tracks and sectors. In reality, the number of tracks and sectors is far, far greater. Additionally, the layout of sectors is no longer symmetrical, to allow the inclusion of more sectors per track as the tracks enlarge away from the spindle. Today's hard disks can have thousands of sectors in a single track and make use of a space allocation technique called zoned recording to allow more sectors on the larger outer tracks of the disk than on the smaller tracks nearer the spindle.

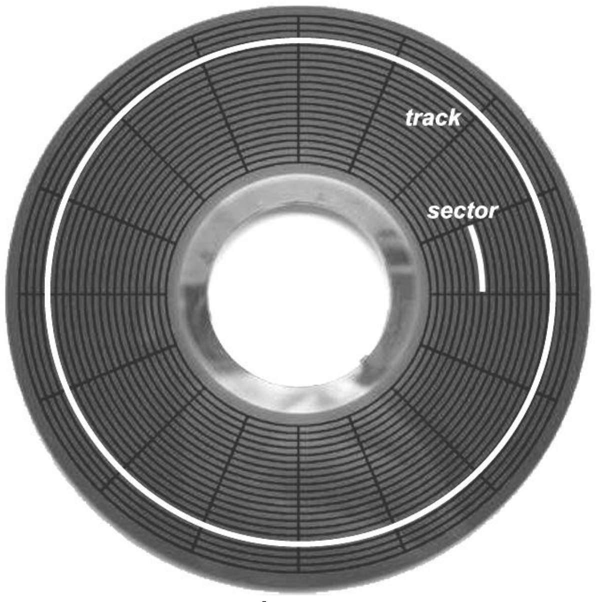


Figure 8
Image used courtesy
Charles Kozierok of pcguide.com

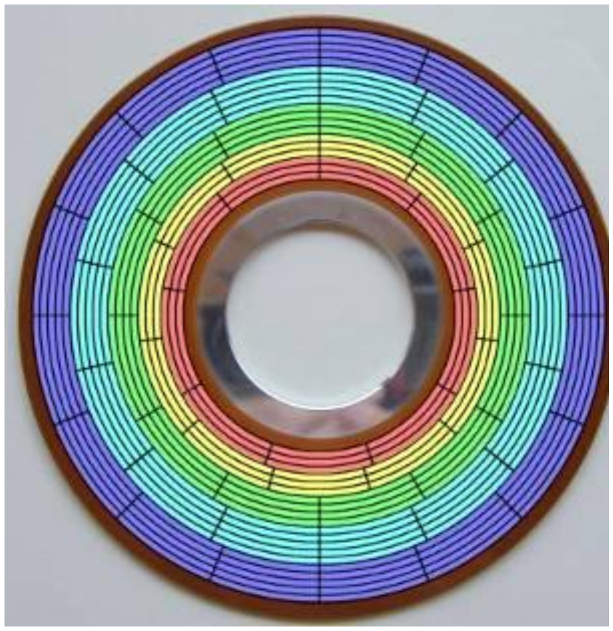


Figure 9
Image used courtesy
Charles Kozierok of pcguide.com

Figure 9 is an illustration of zoned recording. This model hard disk has 20 tracks divided into five zones, each shown as a different color (or shade of gray, if not printed in color). The outermost zone has 5 tracks of 16 sectors; followed by 5 tracks of 14 sectors, 4 tracks of 12 sectors, 3 tracks of 11 sectors, and 3 tracks of 9 sectors. Note that the size (length) of a sector remains fairly constant over the entire surface of the disk, unlike the non-zoned disk representation in Fig 8. Absent zoned recording, if the innermost zone were nine sectors, every track on this hard disk would be limited to only 9 sectors, greatly reducing capacity. Again, this is just an illustration; drives actually have thousands of tracks and sectors.

To this point, we have described only *physical* units of storage. That is, platters, tracks, sectors and even bits and bytes exist as discrete *physical* manifestations written to the media. If you erase or overwrite data at the physical level, it's pretty much gone forever. It's fortunate, indeed, for forensic investigators, that personal computers manage data not physically but *logically*. Because it would be impractical to gather the megabytes of data that comprise most programs by assembling it from 512 byte sectors, the PC's operating system speeds up the process by grouping sectors into continuous chunks of data called **clusters**.

A cluster is the smallest amount of disk space that can be allocated to hold a file. Windows and DOS organize hard disks based on clusters, which consist of one or more contiguous sectors. The smaller the cluster size, the more efficiently a disk stores information. A cluster is also called an **allocation unit**.

Operating Systems and File Systems

Having finally gotten to clusters, the temptation to jump right into latent data is almost irresistible, but it's important that we take a moment to get up to speed with the DOS and Windows operating systems, and their file systems, or at least pick up a smattering of the lingo surrounding same so you won't be bamboozled deposing the opposition's expert.

As hard disks have grown exponentially in size, using them efficiently is increasingly more difficult. A library with thirty books runs much differently than one with 30 million. The **file system** is the name given to the logical structures and software routines used to control access to the storage on a hard disk system and the overall structure in which files are named, stored and organized. An **operating system** is a large and complex collection of functions, including the user interface and control of peripherals like printers. Operating systems build on file systems. If the operating system is the car, then the file system is its engine. Operating systems are known by familiar household names, like **MS-DOS**, **Windows or Vista**. In contrast, file systems go by obscure (and unflattering) monikers like **FAT, FAT32, VFAT** and **NFTS**. Rarely in day-to-day computer use must we be concerned with the file system, but it plays a critical role in computer forensics because the file system determines the logical structure of the hard drive, including its cluster size. The file system also determines what happens to data when the user deletes a file or subdirectory.

The FAT and NTFS File Systems

To simplify a complex subject, this topic will focus on the two file systems used in the Windows environment:, being the **FAT family of file systems** used by DOS, Windows 95-98 and Windows ME, as well as the **NTFS file system** at the heart of Windows NT, 2000, XP, Vista and Windows 7. Be advised that, although these file systems account for the vast majority (85+%) of personal computers in the world, there are non-Microsoft operating systems out there, such as Unix, Linux, MacOS, OS/2 and BeOS. Though similarities abound, these other operating systems use different file systems, and the Unix or Linux operating systems often lie at the heart of corporate and web file servers—today's "big iron" systems--making them increasingly important forensically. Perhaps not today, but within a few years, chances are you'll be seeking discovery of data residing on a machine running a flavor of Linux or MacOS.

The FAT Family

The FAT family refers not to the epidemic of obesity in America but to a lineage of file systems that organize the major disk structures of the hard drive, including **FAT12, FAT16, VFAT** and **FAT32**. FAT is short for **File Allocation Table**, referring to the table of contents that serves as a road map and card catalogue of every bit of data on the drive. The numbers refer to the number of bits used to label the clusters. Since more bits equals a longer address number and a longer address number equals the ability to store more clusters, using 2^{16} bits allowed the cataloguing of 65,536 clusters versus the parsimonious 4,096 clusters (2^{12}) permitted by a twelve bit cluster number.

As with so many aspects of the personal computer, the file system has undergone an evolutionary process spurred by limitations that didn't seem much like limitations at the time each system was designed. For example, the MS-DOS/Windows 3.X file system, known simply as FAT (and also, over time, called FAT12 and FAT16) was originally designed to manage floppy disks (DOS was, after all, short for **D**isk **O**perating **S**ystem). Its greatest virtue was simplicity, but a lack of security, reliability and support for larger hard discs proved its Achilles' heel. Not even the most prescient among us could have anticipated personal computer users would have access to inexpensive one terabyte hard drives. It was simply inconceivable as little as ten years ago. Accordingly, the DOS and Windows 3.X file systems used so limited a cluster numbering system that they were unable to create a disk partition (volume) larger than two gigabytes, and then only if large clusters were used, wasting a lot of disk space (something we will return to later). This limitation lasted right up through the first version of Windows 95! The need to address larger and larger hard drives was a prime mover driving the evolution of the FAT file system.

NTFS

If you spent much time using Microsoft operating systems built on the FAT file system, you don't have to be told how quirky and unreliable the computing experience can be. By the early 1990s, as the networking of personal computers was increasingly common and hard drives were growing by leaps and bounds, the limitations of the FAT family of file systems were all too obvious, and those limitations were keeping Microsoft from selling its operating systems in the lucrative corporate arena. Microsoft realized that if it was going to gain a foothold in the world of networked computers, it would need to retool its operating system "from the ground up."

The New Technology File System (NTFS) was Microsoft's stab at a more reliable, secure and adaptable file system that would serve to meet the needs of business users. The new system offered greater protection against data loss, security features at both the user and file levels (limiting who can view and what can be viewed in the networked environment) and support for both long file names and gargantuan hard drives. The NTFS also makes more efficient use of those larger hard drives.

The NTFS file system is at the center of Windows NT, 2000, XP, Vista and windows 7. Windows XP has been around since 2001 and Windows Seven is now the only entry-level operating system sold by Microsoft; consequently, virtually every PC entering the marketplace today uses the NTFS file system.

NTFS has had a significant impact upon computer forensics as a consequence of the more detailed information stored about system usage. NTFS uses a very powerful and fairly complex database to manage file storage. One unique aspect of NTFS that sets it apart from FAT is that, if a file is small enough in size (less than about 1,500 bytes), NTFS actually stores the file in the Master File Table to increase performance. Rather than moving the read/write heads to the beginning of the disk to read the Master File Table entry, and then to the middle or end of the disk to read the actual file, the heads simply move to the beginning of the disk, and read both at the same time. This can account for a considerable increase in speed when reading lots of small files. It also means that forensic examiners need to carefully analyze the contents of the Master File Table for revealing information. Lists of account numbers, passwords, e-mails and smoking gun memos tend to be small files.

To illustrate this critical difference a different way, if both FAT and NTFS were card catalogues at the library, FAT would direct you to books of all sizes out in the stacks, and NTFS would have all volumes small enough to fit tucked right into the card drawer.

Understanding the file system is key to appreciating why deleted data doesn't necessarily go away. It's the file system that marks a data cluster as deleted though it leaves the data on the drive. It's the file system that enables the creation of multiple partitions where data can be hidden from prying eyes. Finally, it's the file system that determines the size of a disk cluster with the attendant persistence of data within the slack space. Exactly what all this means will be clear shortly, so read on.

Formatting and Partitioning

There is a fair amount of confusion—even among experienced PC users—concerning formatting and partitioning of hard drives. Some of this confusion grows out of the way certain things were done in "the old days" of computing, i.e., fifteen years ago. Take something called "low level formatting." Once upon a time, a computer user adding a new hard drive had to low-level format, partition, and then high-level format the drive. Low level formatting was the initial "carving out" of the tracks and sectors on a pristine drive. Back when hard drives were pretty small, their data density modest and their platter geometries simple, low level formatting by a user was possible. Today, low level formatting is done at the factory and no user ever low-level formats a modern drive. Never. You couldn't do it if you tried; yet, you will hear veteran PC users talk about it still.

For Windows users, your new hard drive comes with its low level formatting set in stone. You need only be concerned about the disk's partitioning into **volumes,** which users customarily see as drive letters (e.g., C:, E:, F: and so on) and its high level formatting, which defines the logical structures on the partition and places at the start of the disk any necessary operating system files. For the majority of users, their computer comes with their hard drive partitioned as a single volume (universally called C:) and already high level formatted. Some users will find (or will cause) their hard drive to be partitioned into multiple volumes, each appearing to the user as if it were an independent disk drive. From the standpoint of computer forensics, perhaps the most important point to remember about FAT partitions is that they come in three different "flavors" called **primary, extended DOS** and **logical.** Additionally, the primary partition can be designated "**active"** and **"inactive.** Only one partition may be designated as active at any given time, and that partition is the one that boots the computer. The forensic significance is that inactive partitions are invisible to anyone using the computer, unless they know to look for them and how to find them. Inactive partitions, then, are a place where users with something to hide from prying eyes may choose to hide it. One simple way to find an inactive partition is to run the FDISK command if the system uses DOS or Windows 95/98/ME. If the system uses Windows Vista, XP, NT or Windows 2000 don't use FDISK. Instead, use Disk Management*,* an enhanced version of FDISK, but BE VERY CAREFUL! You can trash a hard drive in no time if you make a mistake with these utilities.

Cluster Size and Slack Space

By way of review, a computer's hard drive records data in bits, bytes and sectors, all physical units of storage established by the hard disk drive's internal geometry in much the same way as the size and number of drawers in a filing cabinet are fixed at the factory. Sticking with the file cabinet metaphor, bits and bytes are the letters and words that make up our documents.

Sectors (analogous to pages) are tiny segments of thousands of concentric rings of recorded data. A sector is 512 bytes, never more or less. A sector is the smallest individually addressable physical unit of information used by a computer. Computer hard drives can only "grab" data in sector-size chunks.

A common paper filing system uses labeled manila folders assembled into a "red rope file" or master file for a particular case, client or matter. A computer's file system stores information on the hard drive in batches of sectors called clusters. Clusters are the computer's manila folders and, like their real-world counterparts, collectively form files.
These files are the same ones that you create when you type a document or build a spreadsheet.

In a Windows computer, cluster size is set by the operating system when it is installed on the hard drive. Typically, Windows 98/ME ME clusters are 32 KB, while Win7/Vista/XP/NT clusters are 4 KBs. Remember that a cluster (also called an allocation unit) is the smallest unit of data storage in a file system. You might be wondering, "what about bits, bytes and sectors, aren't they smaller?" Certainly, but as discussed previously, in setting cluster size, the file system strikes a balance between storage efficiency and operating efficiency. The smaller the cluster, the more efficient the use of hard drive space; the larger the cluster, the easier it is to catalog and retrieve data.

This balance might be easier to understand if we suppose your office uses 500-page notebooks to store all documents. If you have just 10 pages to store, you must dedicate an entire notebook to the task. Once in use, you can add another 490 pages, until the notebook won't hold another sheet. For the 501st page and beyond, you have to use a second notebook. The difference between the capacity of the notebook and its contents is its "wasted" or "slack" space. Smaller notebooks would mean less slack, but you'd have to keep track of many more volumes.

In the physical realm, where the slack in the notebook holds empty air, slack space is merely inefficient. But on a hard drive, where magnetic data isn't erased until it's overwritten by new data, the slack space is far from empty. When Windows stores a file, it fills as many clusters as needed. Because a cluster is the smallest unit of storage, the amount of space a file occupies on a disk is "rounded up" to an integer multiple of the cluster size. If the file being stored is small, even just a few bytes, it will still "tie up" an entire cluster on the disc. The file can then grow in size without requiring further space allocation until it reaches the maximum size of a cluster, at which point the file system will allocate another full cluster for its use. For example, if a file system employs 32-kilobyte clusters, a file that is 96 kilobytes in size will fit perfectly into 3 clusters, but if that file were 97 kilobytes, then it would occupy four clusters, with 31 kilobytes idle. Except in the rare instance of a perfect fit, a portion of the final storage cluster will always be left unfilled with new data. This "wasted" space between the end of the file and the end of the last cluster is slack space (also variously called "file slack" or "drive slack," and it can significantly impact available storage (Fig. 10).

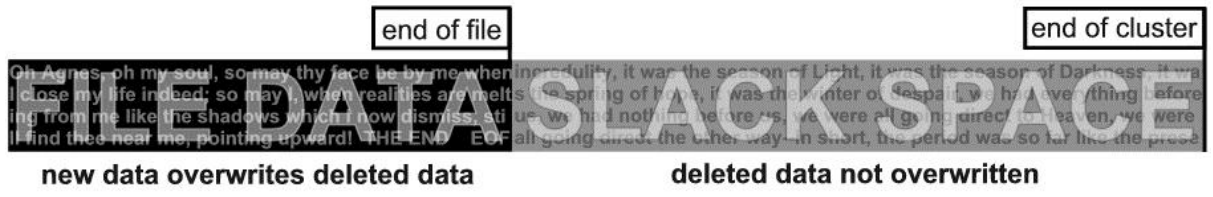

end of file
end of cluster
FILE DATA
SLACK SPACE
new data overwrites deleted data
deleted data not overwritten

Figure 10

When Windows deletes a file, it simply earmarks clusters as available for re-use. When deleted clusters are recycled, they retain their contents until and unless the entire cluster is overwritten by new data. If later written data occupies less space than the deleted data, some of the deleted data remains, as illustrated in Figure 10. It's as if in our notebook example, when you reused notebooks, you could only remove an old page when you replaced it with a new one.

"True, I can't take it with me, but I can take the access codes to it."

© Cartoonbank.com

Though it might seem that slack space should be insignificant —after all, it's just the leftover space at the end of a file— the reality is that slack space adds up. If file sizes were truly random then, on average, one half of a cluster would be slack space for every file stored. But, most files are pretty small--if you don't believe it, take a look at your web browser's temporary Internet storage space. The more small files you have, the more slack space on your drive. It's not unusual for 25-40% of a drive to be lost to slack. Over time, as a computer is used and files deleted, clusters containing deleted data are re-used and file slack increasingly includes fragments of deleted files.

A simple experiment you can do to better understand clusters and slack space is to open Windows Notepad (usually in the Programs>Accessories directory). Type the word “hello” and save the file to your desktop as “hello.txt.” Now, find the file you’ve just created, right click on it and select “properties.” Your file should have a size of just 5 bytes, but the size it occupies on disk will be much larger, ranging from as little as 4,032 bytes in Windows XP or Vista to as much as 32,768 bytes in Windows 95 or 98. Now, open the file and change “hello” to “hello there,” then save the file. Now, when you look at the file’s properties, it has more than doubled in size to 11 bytes (the space between the words requires a byte too), but the storage space occupied on disk is unchanged because you haven’t gone beyond the size of a single cluster

Cluster size can vary depending upon the size of the hard drive volume and the version of FAT in use. The older versions of FAT which you encounter on computers using the first release of Windows 95 or any older version of Windows or DOS will create drives with cluster sizes ranging from 2,048 bytes (2K) to 32,768 bytes (32K). With the introduction of FAT32, introduced with Release 2 of Windows 95 and found in Windows 98, 2000, and ME cluster sizes have tended to be 32,768 bytes, particularly as hard drive size has ballooned. Under the NTFS file system found on Windows 7, Vista, XP and NT, cluster size has dropped down to 4,032 bytes, resulting is less waste due to file slack.

Forensic Implications of Slack Space
In “Jurassic Park,” scientists clone genetic material harvested from petrified mosquitoes to bring back the dinosaurs. Like insects in amber, Windows traps deleted data and computer forensics resurrects it. Though a computer rich with data trapped in file slack can yield a mother lode of revealing information, mining this digital gold entails tedious digging, specialized tools and lots of good fortune and patience.

The Windows system is blind to all information in the slack space. Searching is accomplished using a forensically-sound copy of the drive and specialized examination software, a hex editor utility that permits an examiner to read the data in each cluster directly from the media (or another operating system, like Linux, that treats a drive like a file), permitting string searches of contents. File slack is, by its very nature, fragmented, and the information identifying file type is the first data overwritten.

The search for plain text information is typically the most fruitful avenue in file slack examination and an exercise often measured not in hours, but in days or weeks of review. Experienced computer forensic examiners are skilled in formulating search strategies likely to turn up revealing data, but the process is greatly aided if the examiner has a sense of what he or she is seeking before the search begins. Are there names, key words or parts of words likely to be found within a smoking gun document? If the issue is trade secrets, are there search terms uniquely associated with the proprietary data? If the focus is pornography, is there image data or Web site address information uniquely associated with prohibited content?

Because most lawyers and litigants are unaware of its existence, file slack and its potential for disgorging revealing information is usually overlooked by those seeking and responding to discovery. In fairness, a request for production demanding “the contents of your computer’s slack space” is unlikely to be productive. In practice, the hard drive must be examined by a computer forensics expert employed by one of the parties, a neutral expert agreed upon by all parties or a special master selected by the court.

Bear in mind that while the computer is running, computer data is constantly being overwritten by new data, creating a potential for spoliation. The most prudent course is to secure, either by agreement or court order, forensically-sound duplicates (clones or images) of potentially-relevant hard drives. Such specially created copies preserve both the live data and the information trapped in the slack space and other hiding places. Most importantly, they preserve the status-quo and afford litigants the ability to address issues of discoverability, confidentiality and privilege without fear that delay will result in destruction of data. There's more on this topic to follow.

How Windows Deletes a File
Increasingly, computer users have a vague awareness that when a file is deleted in Widows, it's not necessarily gone forever. In fact, Windows can be downright obstinate in its retention of data you don't want hanging around. Even actions like formatting a disk, long regarded as preemptive to data recovery, won't obliterate all your secrets—far from it (see "The BIG Lie" sidebar, next page). Think about *that* next time you sell an old computer or donate it to the local high school!

How is that deleting a file doesn't, well, *delete* it? The answer lies in how Windows stores and catalogues files. Remember that the Windows files system deposits files at various locations on your disc drive and then keeps track of where it has tucked those files away in its File Allocation Table or Master File Table--essentially a table of contents for the massive tome of data on your drive. This table keeps tabs on what parts of the hard drive contain files and what parts are available for storing new data. When you delete a file, Windows doesn't scurry around the hard drive vacuuming up ones and zeroes. Instead, all it does is add a special hexadecimal character (E5h) to replace the first letter of the filename in FAT systems or add an entry to the master file table in NTFS that tells the system "this file has been deleted" and, by so doing, makes the disk space containing the deleted data available for storage of new data (called "**unallocated space**"). But deciding that a file drawer can be used for new stuff and clearing out the old stuff are two very different things. The old stuff—the deleted data—stays on the drive until it is magnetically overwritten by new data (and can even survive overwriting to some extent—but we're getting ahead of ourselves).

If we return to our library card catalogue analogy, pulling an index card out of the card catalogue doesn't remove the book from the shelves, though consulting the card catalog, alone, you'd think it's gone. Deleting a computer file only removes the card. The file (the "book" in our analogy) hangs around until the librarian needs the shelf space for new titles.
Let's assume there is a text file called secrets.txt on your computer and it contains the account numbers and access passwords to your Cayman Islands numbered account. Let's assume that the bloom has gone off the rose for you, marriage-wise, and you decide that maybe it would be best to get this file out of the house. So, you copy it to a thumb drive and then delete the original. Now, you're aware that though the file no longer appears in its folder, it's still accessible in the Recycle Bin. Consequently, you open the Recycle Bin and execute the "Empty Recycle Bin" command, thinking you can now rest easy. In fact, the file is not gone. All

that has occurred is that Windows has flipped a bit in the Master File Table to signal that the space once occupied by the file is now available for reuse. The file, and all of the passwords and account numbers it holds, is still on the drive and, until the physical space the data occupies is overwritten by new data, it's not that hard to read the contents of the old file or undelete it. Even if the file's overwritten, there's a chance that part of its contents can be read if the new file is smaller in size than the file it replaces. This is true for your text files, financial files, images, Internet pages you've visited and your e-mail.

If a computer has been in use for a while, odds are that it contains a substantial volume of unallocated file space and slack space containing "deleted" data. To illustrate, the old laptop computer on which this paper was originally written in about 2004 had 1.8 gigabytes of free space available on its 30-gigabyte hard drive, and *98.56% of that space contained deleted files*: 474,457 clusters of "deleted" data. How long that data remains retrievable depends on many factors, but one thing is certain: unless the computer user has gone to extraordinary lengths to eradicate every trace of the deleted data, bits and pieces--or even giant chunks of it--can be found if you know where and how to look for it.

What's this Hex Stuff, Voodoo?

Binary numbers get very confusing for mere human beings, so common shorthand for binary numbers is **hexadecimal notation**. If you recall the prior discussion of base-ten (decimal) and base-two (binary) notation, then it might be sufficient just to say that hexadecimal is base-sixteen. In hexadecimal notation, each digit can be any value from zero to fifteen. Accordingly, four binary digits can be replaced by just one hexadecimal digit and, more to the point; a byte can be expressed in just two hexadecimal digits. So 10110101 in binary is divided into two 4-bit pairs: 1011 and 0101. These taken individually are 11 and 5 in hexadecimal, so 10110101 in binary can be expressed as (11)5 in hexadecimal notation.

It's apparent that once you start using two digit numbers and parentheses in a shorthand, the efficiency is all but lost; but what can you do since we ten-fingered types only have 10 different symbols to represent our decimal numbers? Hexadecimal needs 16. The solution was to use the letters A through F to represent 10 through 15 (0 to 9 are of course

The BIG Lie

Since the dawn of the personal computer, if you asked Microsoft, IBM, Compaq, Dell or others how to guard your privacy when selling or giving away a PC, chances are you'd be told to "delete the files and format your hard drive." If you followed this advice, DOS or Windows would solemnly warn you that formatting "will erase ALL data" on the disk." Trouble is, formatting doesn't erase all data. Not even close. This is the big lie. **Formatting *erases less than 1/10th of one percent* of the data on the disk**, such that anyone with rudimentary computer forensic skills can recover your private, privileged and confidential data. If it's not overwritten or physically destroyed, it's not gone. For a fine article on this issue, see the Jan/Feb 2003 issue of IEEE Security and Privacy Magazine or visit: **http://www.computer.org/security/garfinkel.pdf**

Finally, with the release of Windows 7, a full format of a drive WILL overwrite deleted data, but a fast format (used by most people) still will not.

represented by 0 to 9). So instead of saying (11)5, we say the decimal number 181 is "B5" in hexadecimal notation (or *hex* for short).

It's hard to tell if a number is decimal or hexadecimal just by looking at it: if you see "37", does that mean 37 ("37" in decimal) or 55 ("37" in hexadecimal)? To get around this problem, two common notations are used to indicate hexadecimal numbers. The first is the suffix of a lower-case "h". The second is the prefix of "0x". So "B5 in hexadecimal", "B5h" and "0xB5" all mean the same thing (as does the somewhat redundant "0xB5h"). Since a set of eight bits (two hexadecimal digits) is called a *byte,* the four bits of a single hexadecimal digit is called a "nybble" *(I'm not making this up!).*

The significance of hexadecimal notation in computer forensics goes beyond the use of hex byte E5h as a tag used in FAT to mark that the clusters occupied by a file as available for use, i.e., "deleted." Hexadecimal notation is also typically employed (alongside decimal and ASCII translations) in forensic software used for byte-by-byte and cluster-by-cluster examinations of hard drives.

RAM Slack

So far we've talked about recovering the remnants of files that a computer user purposefully stored and deleted. Suppose there were ways to gather bits and pieces of information the user deemed so secret he or she *never* knowingly stored it on the disk drive, perhaps a sensitive report read onscreen from floppy but not copied, a password or an online query. A ***now-defunct*** peculiarity in the DOS and earliest Windows file systems made this possible, but the contents of the data retained was as unpredictable as a pull on a slot machine. These digital lagniappes resided in regions of the drive called **"RAM slack."**

To understand RAM slack, we need to review part of our discussion of file slack. Computers work with data in fixed block lengths called sectors and clusters. Like Nature, a computer abhors a vacuum, so sectors and clusters are always full of *something.* Earlier, we focused on file slack, the data that filled the space remaining when a file couldn't fill the last cluster of

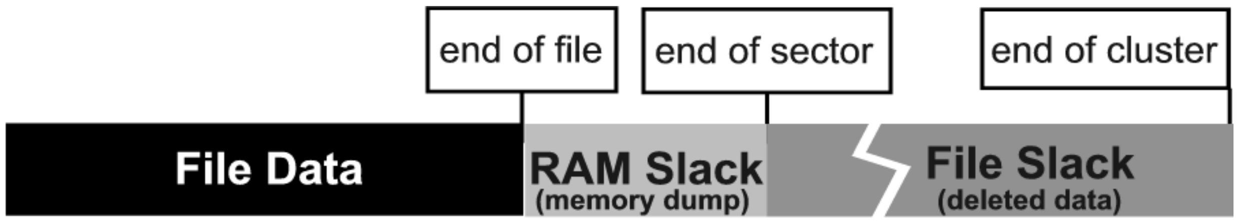


Figure 11

space allocated for its use, deleted data that remained behind for prying eyes to see. This data could range from as little as one byte to as much as 32,767 bytes of deleted material on a typical PC running Windows 98 (eight times less for Windows XP systems). This may not seem like much, but the entire text of the U.S. Constitution plus the Bill of Rights can be stored in less than 32,000 bytes!

Recall that file slack extends from the end of the file stored in the cluster until the end of the *cluster*, but what about the morsel of slack that exists between the end of the stored file and the end of the last *sector.* Remember that sectors are the smallest addressable unit of storage on a PC and are strung together to form clusters. Sectors are only 512 bytes in size and the computer, when it writes any data to disk, *will not write less than a full sector*. But what if the file data being written to the last sector can't fill 512 bytes and there is some slack remaining? If the sector has space remaining in its 512 bytes which it can't fill from the file being stored, older file systems once padded the remaining space with whatever happened to be in the computer's Random Access Memory (RAM) at that moment, hence the name "RAM slack" (see Fig. 11). Granted, we are not talking about a whole lot of data—always less than 512 bytes—but it was enough for a password, encryption key, paragraph of text, or a name, address and phone number. Everything you do on a computer filters through the computers RAM, even if you don't save it to disk; consequently, *RAM slack could contain anything, and there are at least as many instances of RAM slack on a computer that has been in use for any length of time as there are files on the hard drive.* But, though many forensic examiners still talk about RAM slack, the Windows operating system hasn't padded sectors with RAM contents for years, so RAM slack long ago ceased to be a fruitful source of forensic data.

Swap Files

Just like you and me, Windows needs to write things down as it works to keep from exceeding its memory capacity. Windows extends its memory capacity (RAM) by swapping data to and from a particular file called a **"swap file**." When a multitasking system such as Windows has too much information to hold in memory at once, some of it is stored in the swap file until needed. If you've ever wondered why Windows seems to always be accessing the hard drive, sometimes thrashing away for an extended period, chances are it's reading or writing information to its swap file. Windows Vista, XP, NT and 2000 use the term **"page file"** (because the blocks of memory swapped around are called *pages),* but it's essentially the same thing: a giant digital "scratch pad."

Like RAM slack of yore, the swap file still contains data from the system memory; consequently, it can contain information that the typical user never anticipates would reside on the hard drive. Moreover, we are talking about a considerable volume of information. How much varies from system-to-system, but it runs to *millions and millions of bytes*. For example, the page file on the windows machine used to write this article is currently 3.53 *gigabytes in size*. As to the contents of a swap file, it's pretty much a sizable swath of whatever kind of information exists (or used to exist) on a computer, running the gamut from word processing files, e-mail messages, Internet web pages, database entries, Quicken files, you name it. If the user used it, parts of it are probably floating around somewhere in the Windows swap file.

The Windows swap file sounds like a forensic treasure trove—and it is—but it's no picnic to examine. The data is usually in binary form—often without any corollary in plain text--and so must be painstakingly gone through, byte-by-tedious-byte. My 3.53 GB page file theoretically represents some *thirty million pages* of data (an absurd page equivalency, but everyone loves these crazy extrapolations). Although filtering software exists to help in locating, e.g., passwords, phone numbers, credit card numbers and fragments of English language text, it's still very much a needle-in-a-haystack effort (like so much of computer forensics in this day of vast hard drives).

Swap files have different names and may be either permanent or temporary on different versions of Windows. Users can adjust their system settings to vary the permanency, size or location of swap files. The table below lists the customary swap file name and location in each of the versions of Windows, but because these settings are user-configurable, there is no guarantee that the location will be the same on every system.

Because the memory swapping is (by default) managed dynamically in Windows 95, 98 and ME, the size of the swap file changes as needed, with the result that (barring custom settings by the user), the swap file in these versions tends to disappear each time the system is rebooted, its contents relegated to unallocated space and recoverable in the same manner as other deleted files.

Windows Version	Swap File Name	Typical Location(s)
Windows 3.1	386SPART.PAR	Root directory (C:\) Windows subdirectory Windows\System subdirectory
Windows 95, 98, ME	WIN386.SWP	Root directory (C:\)
Windows NT, 2000, XP, Vista, Win7	PAGEFILE.SYS	Root directory (C:\)

Windows NTFS Log File

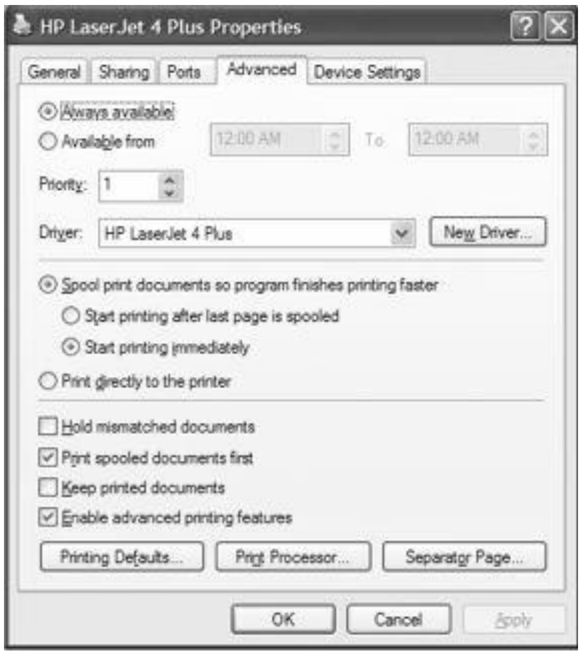
The NTFS file system increases system reliability by maintaining a log of system activity. The log is designed to allow the system to undo prior actions if they have caused the system to become unstable. While arguably less important forensically in the civil setting than in a criminal matter, the log file is a means to reconstruct aspects of computer usage. The log file is customarily named $LogFile, but it is not viewable in Windows Explorer, so don't become frustrated looking for it.

TMP, BAK and Spool Files

Every time you run Microsoft Word or WordPerfect, these programs create temporary files containing your work. The goal of temp files is often to save your work in the event of a system failure and then disappear when they are no longer needed. In fact, temp file do a pretty good job saving your work but, much to the good fortune of the forensic investigator, they often do a pretty lousy job of disappearing. Temp files are often abandoned, frequently as a consequence of a program lock up, power interruption or other atypical shut down. When the application is restarted, it creates new temp file, but rarely does away with the predecessor file. It just hangs around indefinitely. Even when the application does delete the temp file, the contents of the file tend to remain in unallocated space until overwritten, as with any other deleted file.

As an experiment, search your hard drive for all files with the .TMP extension. You can usually do this with the search query "*.TMP." You may have to adjust your system settings to allow viewing of system and hidden files. When you get the list, forget any with a current date and look for .TMP files from prior days. Open those in Notepad or WordPad and you may be shocked to see how much of your work hangs around without your knowledge. Word processing applications are by no means the only types which keep (and abandon) temp files.

Files with the .BAK extensions (or a variant) usually represent timed backups of work in progress maintained to protect a user in the event of a system crash or program lock up. Applications, in particular word processing software, create .BAK files at periodic intervals. These applications may also be configured to save earlier versions of documents that have been changed in a file with a .BAK extension. While .BAK files are supposed to be deleted by the system, they often linger on.



[✓ Always available]
Available from 12:00 AM To 12:00 AM
Priority: 1
Driver: HP LaserJet 4 Plus
[✓ Spool print documents so program finishes printing faster]
Start printing after last page is spooled
[✓ Start printing immediately]
Print directly to the printer
Hold mismatched documents
[✓ Print spooled documents first]
Keep printed documents
[✓ Enable advanced printing features]

Figure 12

If you've ever poked around your printer settings, you probably came across an option for spooling print jobs, promising faster performance. See Figure 12 for what the setting box looks like in Windows XP. The default Windows setting is to spool print jobs so, unless you've turned it off, your work is spooling to the printer. Spool sounds like your print job is winding itself onto a reel for release to the print queue, but it actually is an acronym which stands for (depending upon who you ask) "simultaneous peripheral operations on line" or "system print operations off-line." The forensic significance of spool files is that, when spooling is enabled, anything you print gets sent to the hard drive first, with the document stored there as a graphical representation of your print job. Spool files are usually deleted by the system when the print job has completed successfully but here again, once data gets on the hard drive, we know how tenacious it can be. Like temp files, spool files occasionally get left behind for prying eyes when the program crashes. You can't read spool files as plain text. They must either be decoded (typically from either Windows enhanced metafile format or a page description language) or they must be ported to a printer compatible with the one for which the documents were formatted.

Windows Registry

The Windows Registry is the central database of Windows that stores the system configuration information, essentially everything the operating system needs to "remember" to set itself up and manage hardware and software.

The registry can provide information of forensic value, including the identity of the computer's registered user, usage history data, program installation information, hardware information, file associations, serial numbers and some password data. The registry is also where you can access a list of recent websites visited and documents created, often even if the user has taken steps to delete those footprints. One benefit of the Registry in forensics is that it tracks the attachment of USB storage media like thumb drives and external hard drives, making it easier to track and prove data theft.

In a Windows 95/98/ME environment, the registry is a collective name for two files, USER.DAT and SYSTEM.DAT. In the Windows 7/Vista/XP/NT/2000 environment, the registry is not structured in the same way, but the entire registry can be exported, explored or edited using a program called REGEDIT that runs from the command line (i.e., DOS prompt) and is found on all versions of Windows. You may wish to invoke the REGEDIT application on your system just to get a sense of the structure and Gordian complexity of the registry, but be warned: since the registry is central to almost every function of the operating system, it should be explored with utmost care since its corruption can cause serious, i.e., *fatal,* system errors.

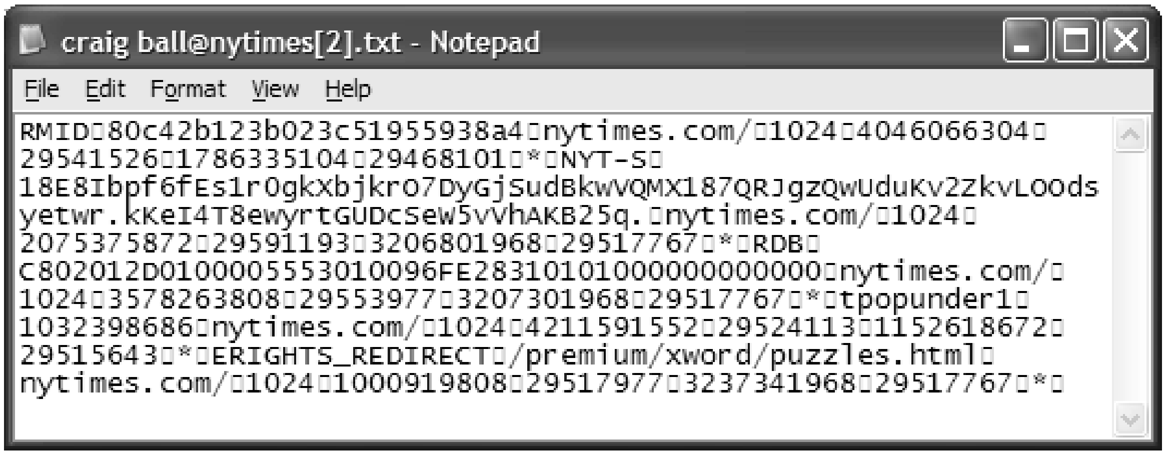
Cookies

Cookies are the most maligned and misunderstood feature of web browsing. So much criticism has been heaped on cookies, I expect many users lump them together with computer viruses, spam and hacking as a Four Horseman of the Digital Apocalypse. Cookies are not malevolent; in fact, they enable a fair amount of convenience and function during web browsing. They can also be abused.

A cookie is a small (<4kb) text file that is deposited in a reserved cookie directory on a user's computer by a website visited by the user. It is, in a sense, a small scratch pad that can be used by a website to store information about the user so that the information can be retrieved by the website in a subsequent visit. Cookies are a means by which websites can personalize the user's online experience or speed the user's authentication. When you go to Amazon.com and the site greets you by name as soon as you arrive, such recognition occurs because the Amazon site has deposited a cookie on your machine during a prior visit. Cookies can contain many things, including a designated user name, a password you've created to access the site, a log of prior visits, customized settings and other data that allows the site to adapt to the user. Cookies can also record the address of the website a user visited just prior to arriving at the site depositing the cookie. When used to enhance and streamline a user's webs surfing, cookies are very beneficial to both user and website operator. It's important to note that cookies are not programs. They are merely electronic Post-It notes, but unscrupulous web site operators who, by working in concert, can assemble data about a user that will facilitate tracking a user's web surfing habits can abuse cookies.

From the standpoint of computer forensics, cookies offer insight to a user's online behavior. Users that take steps to erase their browser history files often forget to dispose of their cookies, which are stored in the cookies subdirectory of the Windows directory on Windows 95/98/ME systems and within the individual user profile on Windows Vista/XP/NT/2000 systems. On my system, I found 5,731 cookies. Very few of them represent any effort by me to customize anything on a website, but one that does is the cookie associated with my online subscription to the New York Times crossword puzzle, shown in Figure 13. Cookies are not required to adhere to any fixed format so note that very little of the cookie's content is intelligible. Most of the data has no value beyond the operation of the website that created it. However, note that the name of the cookie indicates (in Windows XP) the identity under which the user was logged

in when the site was visited. The file's properties (not shown) will indicate the date the cookie was created and the date the web site was last accessed.
A file called INDEX.DAT contained within the Cookies subdirectory is worth examining since it contains a (partially) plain text listing of every site that dropped a cookie on the system, sort of a "super" history file. One provocative aspect of cookies is their ability to act as an authentication key. If the New York Times cookie from my system were copied to the Cookie subdirectory on your system, the New York Times website would see and admit you as me. This potential for extending an investigation using another person's cookie data raises many interesting—and potentially unsettling—possibilities.


craig ball@nytimes[2].txt - Notepad
File Edit Format View Help

```
RMID80c42b123b023c51955938a4nytimes.com/10244046066304
295415261786335104294681010*NYT-S
18E8Ibpf6fEs1r0gkXbjkrO7DyGjSudBkwVQMX187QRJgzQwUduKv2ZkvLOOds
yetwr.kKeI4T8ewyrtGUDcSeW5vVhAKB25q.nytimes.com/1024
207537587229591193320680196829517767*RDB
C802012D01000055530100096FE2831010100000000000nytimes.com/
1024357826380829553977320730196829517767*tpopunder1
1032398686nytimes.com/10244211591552295241131152618672
29515643*ERIGHTS_REDIRECT/premium/xword/puzzles.html
nytimes.com/1024100091980829517977323734196829517767*
```

Figure 13

Application Metadata

Metadata is "data about data." *Application* metadata is a level of information embedded in a file and more-or-less invisibly maintained by the application that created the file. Although application metadata data security issues affect many programs, the epicenter of the application metadata controversy has been Microsoft Word and other components of Microsoft Office. Application metadata grows not out of the *Secret Bill Gates Conspiracy to Take Over the World*, but out of efforts to add useful features to documents, such as information on who created or edited a document, the document's usage and distribution history and much more. The problem with application metadata, especially for lawyers, comes about when people share Word document files. When you send someone (opposing counsel, a client, the court) a Word file on disk or via the Internet, you send not only the text and formatting of the document; you also transmit its application metadata layer. The associated metadata might reveal the amount of time spent editing the document and identify others with whom the document was shared. The metadata might also include collaborative commentary, earlier versions of the document and even the fact that you merely recycled a document prepared in another matter or purloined from another lawyer! In short, application metadata can cause problems ranging from embarrassment to malpractice.

In its Knowledge Base Article Q223396, Microsoft details some examples of metadata that may be stored in documents created in all versions of Word, Excel and PowerPoint, including:

- Your name
- Your initials
- Your company or organization name
- The name of your computer
- The name of the network server or hard disk where you saved the document
- Other file properties and summary information
- Non-visible portions of embedded OLE objects
- The names of previous document authors
- Document revisions
- Document versions
- Template information
- Hidden text
- Comments

While some application metadata is readily accessible just by viewing in the Office application, other application metadata can only be seen using a low-level binary file editor. Microsoft offers a free "Hidden and Collaboration Data Removal" utility for download. You can locate it by running a search at www.microsoft.com for "rhdtool.exe". While most metadata can be removed from Word documents, without buying any software, a simple and effective way to identify and eliminate metadata from Word documents is an $80.00 program called the Metadata Assistant, sold by Payne Consulting Group (www.payneconsulting.com).

Hidden Data
Most of what we have discussed thus far centers on data that no one has sought to conceal, other than by deletion. But, there are techniques by which data can be concealed on a computer, ranging from the unsophisticated and retrievable to the sophisticated and (practically) irretrievable. For example, files can be given the attribute "hidden" so as not to appear in directory listings. This is easily overcome by, e.g., issuing the dir /ah command, but you have to know to do this in your search. Data can be hidden in functional sectors marked as "bad" in the file table such that the systems simply skip these sectors. Here, the characterization of the sector will need to be changed or the sectors themselves will need to be examined to extract their contents. Earlier, this article discusses the use of inactive partitions to hide data; that is, hiding data in areas "unseen" by the operating system. Encrypted data poses near-insurmountable challenges if the encryption is sufficiently strong and unencrypted data hasn't found its way into swap files and slack space. Finally, and perhaps most insidious because of its simplicity, is the hiding of data in plain sight by simply changing its filename and file extension to seem to be something it is not, such as by renaming pornographic jpeg files as something that would not normally garner any attention, like "format.exe." Unless one compares file sizes or examines the files' contents and attributes with care, there would be little reason for a casual investigator to find the wolf in sheep's clothing.

Shadow Data
As previously discussed, data on a hard drive is stored in thousands of concentric rings called tracks over which a tiny read/write head flies, reading and writing information as densely packed necklaces of magnetic fluctuations. This feat requires a mechanical precision unlike almost any other we encounter in our daily lives. But hard drives haven't always been as

precise as modern disks and, in the days before mind-boggling data densities, minute variations in track alignment and in the size and penetration of the recording field were common. As a consequence, each time a track was overwritten, the read/write head might not completely cover the pre-existing data. Some of the magnetic information containing overwritten data may have "swerved" out of the track path due to wobbling in the head or other misalignment. Earlier disk writes may have occurred with the read/write head a bit further away from the surface of the disc, widening (and deepening) the bands of recorded data.

The consequence of this infinitesimal 3-D variation is that a remnant of previously recorded data can exist just beyond the borders of each track or at different levels in the physical media. This fringe of potentially recoverable information data is called **"shadow data."** Shadow data can potentially exist on ancient hard drives, floppy discs, backup tapes and Zip disks. Figure 14 is a graphical representation of what shadow data might look like on a disc drive if it were visible.

Massive jumps in areal data density of hard drives means that shadow data is a creature of history and now the exclusive province of three-letter government agencies in high-tech spy novels and urban legend. No one has ever published an account of a successful recovery of any consequential volume of data employing magnetic remnance techniques on modern hard drives. In short, overwritten data on modern hard drives is *not recoverable*, even if overwritten just once.

Other Revealing Data

In addition to the latent data possibilities described above, a thorough forensic investigation will look at a user's hibernation file, browser cache files (also called Temporary Internet Files in Internet Explorer), browser history files, Link files, Prefetch data, web Bookmarks and Favorites, file dates and more. Of course, the user's e-mail and their Recycle Bin must also be explored. An alert investigator will also look at the nature of software installed on a computer and the timing of that software's installation; that is, contextual analysis.

Contextual Analysis

The complexity and interactive nature of a personal computer permits revealing information to be gleaned not only from the contents of discrete files but also from the presence or absence of certain files and programs, as well as the timing of their appearance or disappearance. For example, the recent appearance of encryption or steganography applications (the latter employed to conceal data by invisibly integrating it within other carriers, usually drawings or photographs) may be a red flag that the user has hidden or encrypted data on the drive. The presence of a user-installed copy of the Quicken financial management program coupled with the absence of any financial data files may suggest that data has been removed from the machine. Similarly, the presence of a user-installed facsimile software program should trigger a search for facsimile image files.

If you were to examine usage patterns for a typical Windows PC, you'd find that more than 90% of the programs and files on the drive are never used in any given year. Most of us access the same little neighborhood of files and programs and rarely stray from them. This near-universal trait has both positive and negative implications for computer forensics. The positive is that the vignette of files likely to contain revealing information is small relative to the giant canvas of the hard drive, but the down side is that these needles hide in a very large haystack. If the discovery plan requires combing through or, worse, printing out "everything" on the drive, then it will be a gargantuan exercise, more than 90% wasted. If we focus instead only on those files that have been accessed or modified within a specified look back period, we need to have some basis on which to treat each file's date attributes as reliable. In fact, changing file dates is child's play and, absent an ability to validate the system clock at the time the attributes were applied, even dates that haven't been fudged may be fanciful without contextual analysis.

Going, Going, Gone
So far, this paper has spoken primarily of what information is available to find on a Windows personal computer and where it might be found. Now, we turn briefly to a few practical considerations in dealing with that data. If you look back at what we have covered heretofore, you'll see that a large volume of the potentially revealing information to be found is latent data, and the bulk of that data resides within unallocated space on the hard drive (e.g., in the slack space). Similarly, key forensic data like the swap files, hibernation files, TEMP files, log files and so forth are dynamic. They change constantly as programs are run and documents created. The point of all this is that unallocated space gets allocated and dynamic files change as a computer is used. For that matter, latent data can be progressively destroyed even when the computer is not in use, so long as the power is connected and the operating system is running. As hard as it is to obliterate *specific* data from a computer, some latent data is being completely destroyed all the time a computer is in operation, overwritten by new data. Your smoking gun is gradually being destroyed or, worse, may soon be disrupted by disk maintenance utilities that defragment the disk. Considering that Windows accesses and changes hundreds of files each time it boots, you can appreciate that doing nothing may be tantamount to allowing evidence to be destroyed. Every time you boot windows you destroy or alter data. The creation of temp files, the updating of logs, the reading of configuration files may all seem benign acts, but they likely entail use of unallocated space, overwriting of latent data and alteration of metadata values.

Bit Stream Backup
Once latent data is overwritten, it's pretty much gone forever. If you want to preserve the status quo and retain access to latent data, the only practical way to do so is by making a bit stream copy of the hard drive. A bit stream copy is a sector-by-sector/byte-by-byte copy of a hard drive. A bit stream copy preserves not only the files and directory structures; it preserves all of the latent data, too. Anything less will leave potential evidence behind. It's critically important that you appreciate the difference between a bit stream copy and an archival copy of the type that people create to protect them in the event of a system crash. Archival backups copy and retain only the active files on a drive, and frequently not even all of those. If you can imagine a hard drive with all latent data stripped away, you'd have a pretty good picture of an archival back up. In short, an archival back up is simply no substitute for a bit stream back up when it comes to computer forensics.

Computer forensic specialists create bit stream images using dedicated hardware imaging tools or programs like EnCase, FTK Imager or X-Ways Forensics. These and other commercially available programs make the mirroring process easier, but an identical copy of every sector of a drive can also be made using a free utility called Linux DD (which runs under the also-free Linux operating system, but not on a machine running DOS or Windows). Whatever device or program is used, it is essential that the examiner be able to establish its reliability and acceptance within the forensic community. The examiner should be able to demonstrate that he or she has a valid license to own and use the software as the use of a bootleg copy could prove an embarrassing revelation in cross-examination.

The creation of a forensically competent bit stream copy entails a second step. It is not enough to simply make a faithful copy of the disk drive; a forensic examiner must be equipped to irrefutably demonstrate that the copy does not deviate from the original, both immediately after it is created and following analysis. This is typically accomplished using some mathematical sleight-of-hand called **"hashing."** Hashing a disc creates a digital fingerprint; that is, a small piece of data that can be used to positively identify a much larger object. Hashing is a form of cryptography that relies upon a concept called "computational infeasibility" to fashion unique digital signatures. Essentially, the entire contents of any stream of digital information is processed by a specialized mathematical equation called an "algorithm" that works in only one direction because it would be a gargantuan (i.e., "computationally infeasible") task--demanding hundreds of computers and thousands of years--to reverse engineer the computation. The bottom line is that if the bit stream copy of the data is truly identical to the original, they will have the same hash values; but, if they differ by so much as a comma (well, a byte), the hash values will differ markedly. The computational infeasibility means that someone trying to pass a doctored drive off as a bit stream copy can't make changes that will generate an identical hash value. There are a number of hash algorithms floating around, but the two most frequently employed in computer forensic work are called **MD5** and **SHA1**. Programs that create bit stream copies may also employ another form of authentication called **"Cyclic Redundancy Check" (CRC)**. CRC may be done before MD5 or SHA1 hashing or (less desirably) instead of it.

Computationally infeasible is not the same as computationally impossible, but it might as well be. From the standpoint of relative probabilities, before two hard drives with differing content could generate the same MD5 hash value ("hash collision"), you'd have won the lottery a *billion billion billion billion billion times*. That said, hash collisions have been contrived for the MD5 algorithm, but not in a manner that should give anyone pause in its near-term continued use to authenticate duplicate drives.

Now What?

But let's beam out of the digital domain and return to the practice of law on planet Earth. Either the opposition has computer data you want or you have computer data the other side may want. You now appreciate that evidence is potentially being destroyed as the computer is used. Now what?

When the government faces this dilemma, they have a pretty handy solution: get a warrant and seize everything. For the rest of us, getting, or even just preserving, computer evidence can be an uphill battle. If a computer is used to run a business, can you persuade the judge to order it be turned off and sequestered? If the computer is a mish-mash of personal, professional, private and privileged information, is it proper for the judge to order a wholesale copy of the hard drive to be turned over to the opposition? Where is the line between unwitting destruction of latent evidence and spoliation? These are not easy questions, but the law has generally recognized that the mere fact that the party opposing discovery has adopted a high tech filing system should not operate to deprive a party of access to discoverable material. If you would be entitled to inspect or copy the information were it on paper, why should that right be diminished because it's digitized?

When is Forensic Analysis Warranted?

"To a man with a hammer, everything looks like a nail," wrote Mark Twain. The same might be said of attorneys whose clients have benefited from the use of computer forensics in electronic discovery. Understandably, they want access to the other side's systems in every case. But, as powerful a tool as it is, computer forensic analysis probably has a place in less than one-in-ten litigated matters. The challenge for the court is identifying the issues and circumstances justifying forensic access, imposing appropriate safeguards and allocating the often-substantial cost.

It's long settled that evidence is discoverable whether it exists on paper or solely as a microscopic arc of magnetic data on a disk; but are we entitled to root around in another's computer hard drive when we couldn't do the same in their file room? The answer seems to be "occasionally." Absent a showing of abuse, the rules of procedure invest the responsibility to locate, preserve and produce discoverable material on the producing party. If the producing party responds "it's not there," the requesting party is largely bound to accept that representation unless there is some credible basis to suggest it's unreliable. But most people lack the skill and tools to identify, preserve and extract latent computer data; so the statement "it's not there" is, at best, "it's not where we looked, and we haven't looked thoroughly."

By the same token, it's not reasonable to expect a responding party to hire a computer forensic examiner and perform a thorough search for latent data in every case. It's too expensive, time-consuming and not always certain to lead to the discovery of relevant evidence. Neither can the requesting party's forensic expert be granted unfettered access to an opponent's computers absent steps to protect the confidentiality of proprietary, privileged or just-downright-embarrassing material. A balance must be struck between the potential for discovery of relevant evidence and the potential for unwarranted intrusion at great expense.

The most obvious instance where forensic examination is indicated is a matter involving a credible allegation of negligent or intentional spoliation, or concealment, of electronic information or its paper counterpart. Another is a circumstance where it appears likely that relevant and discoverable data exists, but is accessible only through the use of forensic restoration techniques. Other instances include matters where computers have allegedly been employed to perpetrate a crime, fraud or tort, such as theft of trade secrets, workplace harassment, concealment of assets, hacking, theft of service, electronic vandalism, identity fraud, copyright infringement, etc.

Forensic Imaging Should Be Routine
Since it's not always possible to ascertain the need for computer forensic analysis at the onset of a dispute and with computer data being so volatile and fluid, how can a litigant preserve the status quo and protect potentially discoverable data? The best answer seems to be to act decisively to enforce the obligation to preserve while deferring disputes concerning the obligation to produce. At least with respect to the computer systems used by key players, if an opponent is unwilling to immediately remove them from service and secure them against tampering, loss or damage, then it is imperative that the hard drives for each computer be duplicated in a *forensically-sound* fashion and secured. They may never be used but, if needed, there is no better mechanism to demonstrate diligence in the preservation of discoverable data. The same prudence applies to other media which may later be claimed to have contained relevant and discoverable data, including personal digital assistants, e-mail servers and online repositories. *Caveat:* Routine file back up to tape, floppy disks, recordable CDs, thumb drives or other media using virtually any off-the-shelf back up application will *not* produce a forensically sound clone of the data, rendering some or all latent data unrecoverable in the future, ripe for a charge of spoliation.

Answers to Frequently Asked Questions about Forensic Imaging

What is a "forensically-sound" duplicate of a drive?
A "forensically-sound" duplicate of a drive is, first and foremost, one created by a method which does not alter data on the drive being duplicated. Second, a forensically-sound duplicate must contain a copy of every bit, byte and sector of the source drive, including unallocated "empty" space and slack space, precisely as such data appears on the source drive relative to the other data on the drive. Finally, a forensically-sound duplicate will not contain any data (except known filler characters) other than which was copied from the source drive. All of this must be achieved in an authenticable way.

What's the difference between a "clone" and an "image" of a drive?
These terms are often used interchangeably, along with others like "bit stream copy," "mirror" and "ghost." So long as the duplicate is created in a forensically-sound way and can be reliably verified to be so, the name attached to the duplicate doesn't make much difference. However, the term "drive image" is most closely associated with a method of forensic duplication whereby all of the data structures on the source drive are stored in a file or series of files which, though structurally different from the source drive, can be reconstituted ("restored") in such a way as to be a forensically-sound duplicate of the source drive. A drive image is typically used with compression algorithms to store of the source drive data in a more compact fashion. Though a drive image is capable of being restored to create a clone drive, modern drive analysis software is designed to "virtually restore" the drive, reading directly from the image file and "seeing" the forensically-sound duplicate drive without the necessity for restoration.

How do you make a "forensically-sound" duplicate of a drive?
Although many forensic examiners use similar techniques and equipment, there is no one "recognized" or "approved" way to create a forensically-sound duplicate of a drive. There are a number of hardware and software tools well-suited to the task, each with their strengths and weaknesses, but all are capable of creating a forensically-sound duplicate of a typical PC hard

drive when used correctly. Keep in mind that there are many different types of digital media out there, and a tool well-suited to one may be incapable of duplicating another. You simply have to know what you are doing and select he correct tools for the job

Duplication tools fall into two camps: those which create a drive image (a file which can be restored to match the source) and those which create a clone drive (a target drive or other media that duplicates the source data without the need for data restoration). My favored approach was once to clone drives but that outmoded approach has entirely given way to drive imaging. Again, done right, either approach works. Just get everything (including unallocated clusters and file slack) and be sure you can authenticate the duplicate.

To create forensically sound copies of hard drives, I've variously used a host of approaches, ranging from generic software capable of producing a bit stream duplicate to custom-built applications exclusively for forensic drive duplication to handheld devices that automate nearly the entire process. One alternative is a hardware cloning devices like those from Voom Technogies (http://www.voomtech.com), CPR Tools (http://www.cprtools.net) Intelligent Computer Systems (www.ics-iq.com), Tableau (http://www.tableau.com/) or Logicube, Inc. (www.logicube.com). For high speed onsite acquisition, I use a Voom HardCopy 3P handheld drive duplication tool ($1,410.00) that allow me to simply hook up a source and target drive, push a few buttons and go. I've tested its accuracy using hash signature tools and, in every instance, the duplicate created by the Voom was forensically sound. Hardware-based write-blocking device is embedded in the device and intercepts any efforts to write to the source drive.

Other specialized duplication methods entail using forensic applications like Forensic Tool Kit Imager (nominally $89.00 but freely downloadable from Access Data; www.accessdata.com), EnCase Forensic Edition ($3,600.00 from Guidance Software, Inc.; www.guidancesoftware,com) or X-Ways Forensics ($1,270.00 from X-Ways Software Technology AG; www.x-ways.com) to create a drive image. These applications are designed expressly to support computer forensic examiners and are all excellent products. For a less-costly approach, consider Symantec's Norton Ghost ($69.95 from Symantec, Inc.; www.symantec.com) or the free Linux dd utility (included with any version of Linux). Ghost has been maligned as a forensic tool because, when used with its default commands and settings, it violates the cardinal rule of computer forensics—it changes data on the source drive. However, if Ghost is used with care—and the correct command line switches and settings are selected—it's capable of creating either a forensically-sound image or clone disk. If you're adept with the free Linux operating system, using Linux' dd (for disk dump) utility is surely the most cost effective solution. Here again, in untrained hands, dd is an unforgiving application and can destroy evidence; but, used with care by one who knows what they are doing, it's a gem.

There are many products on the market that claim to duplicate "everything" on a drive, but beware, as most are merely back up utilities and don't preserve the unallocated space. Unless the product carries over ever bit and sector of the source drive, without modification or corruption, it's wholly unsuited for computer forensics. Before settling on any duplication product, peruse the literature, solicit recommendations from computer forensic specialists and review the (woefully out-of-date) test results at the National Institute of Standards and Technology's (NIST) Computer Forensic Tool Testing program (http://www.cftt.nist.gov/index.html).

How can you prove the duplicate drive is forensically sound?
Considering the size of modern hard drives, one way you *can't* prove the validity of your duplicate is by manually comparing the data. It's just impossible. So, the process of verification has got to be automated and foolproof. To appreciate the solution, take a moment to ponder the problem: how can you examine perhaps forty, sixty, eighty *billion* entries on a duplicate drive and be certain that every one of them has precisely the same value and is in the exact same relative location as on the source drive? Not just be *certain*, but be more reliably certain than fingerprints and more than DNA evidence. This is where we say "thanks" to all the mathematical geniuses who gave up normal human interaction to dedicate their lives to algorithms, arrays and one-way computations. These are the brainiacs who thought up "hash functions" and "message digests."

A hash function accepts a value of any size as its input, performs a complex calculation on that input and returns a value of fixed length as its output. The output value functions as a unique representation of the input. Put in a complex "message" and out pops a long string of letters and number bearing no discernable relationship to the message but which can only be generated by the one input. Accordingly, the output is called a "message digest." The really amazing part of this is that the computation only works in one direction. It's considered "computationally infeasible" to decode the input from the output, which is a fancy way to say "Fuhgeddaboudit!" Since the input message can be anything, someone had the very bright idea to use the entire contents of a hard drive or thumb drive as the input and—voila!—the output becomes a fingerprint of that drive's contents and layout. Change so much as one single bit somewhere on the drive and the message digest changes dramatically. Since the fingerprint is unique to the inputted message (here, the data on the drive) only a forensically-sound duplicate of the drive could generate the same message digest.

Two widely-used hash functions are called MD5 and SHA-1. MD-5 generates a 32 character (128-bit) string that might look something like this: *9E2930D48131COFC9EE646AE2197A69C*. No matter how long or short the input, the MD5 output always is thirty-two characters in length. The chance of two different inputs producing the same MD5 message is greater than 1 in 340 undecillion. That's a staggering I in *340,000,000,000,000,000,000,000,000,000,000,000,000* chance! That beat's the pants off of DNA and fingerprints, and SHA-1 is even *more* reliable.

In 2004, four Chinese researchers, Xiaoyun Wang, Dengguo Feng, Xuejia Lai and Hongbo Yu, succeeded in using a supercomputer to fabricate slightly different files with identical MD-5 hash values. Though still an excellent tool for validation, experts expect a gradual move away from MD-5 to even more secure hash algorithms.

Steps to Preserve the Evidence
A thorough exploration of the legal issues and precedents concerning the duty to preserve and produce electronically stored information is beyond the scope of this paper, but the near-term goal must be to preserve the status quo lest, like the lawyers litigating *Jarndyce v. Jarndyce* in

Charles Dickens' "Bleak House," the lawyers keep squabbling until there is nothing left to fight over. Faced with a potential for forensic analysis, forensically sound duplication of potentially relevant media is key to preserving evidence.

As soon as it appears that computer data—and above all, latent data—may lead to the discovery of admissible evidence (or may meet whatever standard your jurisdiction applies to define what must be preserved), several things should be done:

1. The opposition should be expressly advised that the computer data is regarded as evidence and that immediate steps must be taken to preserve all such evidence until the court has an opportunity to address its discoverability. Because few people have a full appreciation of how much latent data exists on their machines or the adverse impact ongoing use can have on such data, you will need to be quite specific in your description of the actions to be taken or avoided, as well as in your identification of the target media. In some instances, you may be justifiably concerned that such a communiqué will serve as a road map to the destruction of evidence, but if you hope to have any chance of proving spoliation, you will need to be certain that ignorance won't serve as a defense. For further guidance in drafting a preservation notice, see the article entitled, "**The Perfect Preservation Letter"** at www.craigball.com.

2. Begin the process of educating the court about electronic evidence by moving for a protective order requiring that the party in possession of the computer refrain from any action that may impair the ability to recover latent or dynamic data. The goal initially is not to fight all the discovery battles, but only to preserve the status quo so that evidence doesn't disappear.

3. Secure two forensically sound duplicates of the evidence media. Once the accuracy has been established by hashing, you will want to leave one copy completely untouched and use the other for analysis to guard against any accusation that data was altered or corrupted during analysis. Hard drives are cheap. Sanctions are expensive. Preserve a chain of custody with respect to the copies or you will impair their usefulness. Be certain that the person selected to make the copies is fully qualified by training or experience to do so. You may be choosing a courtroom witness, so demeanor and appearance should play a role in your selection.

4. Seek an agreement with opposing counsel to engage, or get a court order to appoint, a special master to act as an impartial custodian of the original media and/or bit stream copies. Ideally, the special master should be both an attorney and skilled in computer forensics. It may not be necessary for the special master to be a computer forensics expert—he or she can hire skilled personnel as needed and supervise their work—but the master must be sufficiently conversant in all of the principal issues discussed in this article so as to be able to guide the court and communicate with technical personnel. Using a lawyer as the special master streamlines the identification and resolution of privilege, privacy, trade secret, relevance and discoverability issues. Some courts vest in the special master a limited authority to resolve discovery disputes within the ambit of the master's delegated responsibility. No matter how such matters are handled, the master's duty is to serve as an *impartial* custodian or arbiter, affording both sides a full and fair opportunity to have their concerns aired and their rights protected.

What's It Going to Cost?
Computer forensic analysis is exacting work requiring specialized knowledge, specialized tools, patience, tenacity, restraint, insight and no small measure of investigative talent. Analysts tend to come from the ranks of law enforcement or the military; but neither a working knowledge of forensic procedures nor an intimate acquaintance with computers alone suffice to qualify one as a computer forensic specialist. A competent forensic analyst needs both skill sets. That is, of course, a prelude to saying, "it's expensive."

Plan on paying from $150.00 to $500.00 per hour for forensic analysis and, while a quick-and-dirty, well-focused drive analysis might be completed in a day or two, a complex analysis can take much longer.

One area in which costs can never be cut is in the use of slipshod evidentiary procedures. No matter how convinced you might be that the information uncovered will never be offered in court, a competent forensic examiner won't do the job in a way that will taint the evidence. A competent examiner never boots from the original drive. A competent examiner never "just takes a quick peek" at the data. A competent examiner never uses the original media in the analysis. *Never!*

Using a computer special master with a law degree, litigation experience and computer forensic ability is going to cost $350.00 to $550.00 or more per hour depending upon training, experience and stature, but the additional cost should be offset by a quicker resolution of discovery disputes and a diminished reliance upon the court acting *in camera.* The use of an impartial master with computer skills can also free the parties from having to hire their own computer forensic experts. For more on this, read, "**Finding the Right Computer Forensics Expert," and E-Discovery: A Special Master's Perspective"** at www.craigball.com.

Who Pays?
With the advent of electronic discovery, the longstanding presumption that a producing party bears the cost to identify, collect and bring forward material sought in discovery is increasingly being challenged by litigants and re-examined by courts. Meeting an e-discovery demand in the 21st century can be substantially more costly than its 20th century paper-centric counterpart. The higher cost of electronic discovery is a function of the greater volume, depth and complexity of electronic recordkeeping and a problem exacerbated by fundamental flaws in the way computers and users create and store digital information. The good news is that it's not always going to be more expensive and, when we finally get our digital acts together, e-discovery will be the *only* cost-effective solution. Until then, lawyers can look forward to years of quarreling over who pays. As a general proposition, the party seeking forensic analysis pays for that work unless the need for the effort arose because of malfeasance on the part of the other side (e.g., data destruction or alteration).

Is Digital Different?
Faced with a demand for cost shifting, the party seeking electronic discovery might wonder, "Why should the courts depart from the longstanding practice that the producing party pays? Should a requesting party be disadvantaged simply because an opponent has adopted an

electronic mechanism for creating and storing information? *We're* not asking for more, *they're* just creating and keeping more of the stuff we seek! Didn't the producing party *choose* to computerize, voluntarily and for its own benefit? "

In fact, the stampede to computerization, with the attendant strain on discovery boundaries and budgets, was so broad and deep a sea change, why even call it a choice? We got where we are before anyone realized how far out on the limb we'd climbed. A device evolved from an electronic toy no one expected to succeed now sits on every desk and serves as the conduit for much of our communication, research, commerce, entertainment and misbehavior. Does the shift from paper to bits and bytes matter? Is digital different?

A business record born on paper (e.g., a handwritten form or a letter from a typewriter) is pretty much "what-you-see-is-what-you-get." There is no layer of information lurking within the fibers of the paper. You don't need special tools or techniques to glean the contents. A photocopy probably conveys about as much useful information as the original. Absent forgery, the author and addressee are there in black-and-white. But, digital is different. Computer-generated documents all have *metadata* associated with them. That is, data about data, information outside the body of the document that conveys such things as when the document was created or modified, its author, electronic format, file size and more. Moreover, the creation of an electronic record often engenders the creation of a host of other records, some, like back up files or prior versions, the users knows about and some, like log, spool and swap files, the user may never imagine exist. Computers have also facilitated the recording of communications that, not long ago, wouldn't have been reduced to writing. E-mail now stands in for conversations that would have been phone calls or water cooler chitchat twenty years ago. The end result is that discoverable information exists in new planes, not only a broader swath of data, but a deeper level as well.

An exponential increase in discoverable volume is not the only challenge, nor is it the most difficult to resolve. A greater hurdle stems from the manner in which computers retain and dispose of information. Can you imagine a business that managed all its records and transactions—personal, professional, intimate, recreational, confidential and privileged--by dumping one and all into a big bin? How about a lawyer dumping every scrap of paper in her life--wills, bills, stills, frills and thrills--in a giant folder labeled "Stuff?" It's hard to image that level of incompetence, but we'd certainly expect that such malfeasance--commingling client materials with personal and third party stuff--would hobble claims of privilege or confidentiality. Yet, that's what a computer routinely does in its management of swap files, e-mail folders and the web surfing cache, to name just a few problem areas.

If that's not bad enough, the computer is a trash can without a bottom! You try to tidy up by deleting files and the computer just hides them (or pieces of them) from you, squirreling data away like acorns, willy-nilly, across a vast expanse of hard drive! Is it any wonder that trying to makes sense of this mess is expensive?

Lawyers frequently approach e-discovery as they've always done with paper records. But we've had thousands, of years to master the management of paper records, and the innate physicality of a writing means it's easier to track, isolate and, ultimately, destroy. Digital is different, and, while the rules of procedure and evidence may prove sufficiently flexible to adapt to a virtual world, some in the bench and bar loathe straying far from their familiar, paper-based systems. Inflexibility boosts the cost of electronic discovery, through, *inter alia*, the use of tortured definitions of “document” in discovery requests, impossibly overbroad production demands, compulsory “blow back” of native digital data to paper printouts (with the attendant loss of the metadata layer). More costly still is the practice of reducing data to a page-like format to facilitate privilege review. When even a modestly-sized hard drive can easily generate a million “pages” of documents and a server, tens- or hundreds of millions of pages, there are simply not enough eyeballs that can be placed in front of enough desks to complete the job in the customary fashion. Because digital is different, we must change as well.

Shifting Costs: The Rowe and Zubulake Decisions

Though this discussion has steered wide of the burgeoning case law governing electronic discovery, one can’t talk about planning for the cost of computer forensics in e-discovery without at least touching on the two most important decisions on the topic: *Rowe Entertainment, Inc. v. The William Morris Agency, Inc.*, 205 F.R.D. 421 (S.D.N.Y. 2002) and *Zubulake v. UBS Warburg LLC*, F.R.D. 309 (S.D.N.Y. 2003).

The import of these decisions is that they articulate factors to be weighed by a court in determining whether the cost of responding to a discovery request should be shifted to the party seeking discovery. The *Rowe* opinion put forward eight factors, but proved to favor disproportionately large entities resisting discovery. Accordingly, the Court in *Zubulake*--a discrimination case where the plaintiff sought e-mail stored on backup tapes—re-visited the *Rowe* factors and derived a three-part approach to determining whether cost-shifting is appropriate.

Significantly, the *Zubulake* court makes clear that if the materials sought are "accessible" (e.g., active online data or readily available near-online data like optical disks), the responding party bears the cost of production absent undue burden or expense warranting protection. However, if the materials sought are inaccessible--such as e-mail on legacy backup tapes and most information developed through forensic examination--the Court may consider cost shifting and undertake a factual inquiry to identify what type of information is likely to reside on the “inaccessible” media. This inquiry may entail some sampling of the inaccessible media to gauge its relevance and the level of cost and effort in meeting the discovery request. Finally, as the third leg of the analysis, the Court set out seven factors to be used in balancing interests and burdens. In the order of importance which the Court ascribed to them, the seven considerations are:

(1) *Is the request specifically tailored to discover relevant information?*
(2) *Is the information available from other sources?*
(3) *How does cost of production compare to the amount in controversy?*
(4) *What are the relative positions of the parties in terms of resources?*
(5) *Who is best able to control costs and has an incentive to do so?*
(6) *Are the issues in discovery key to the issues at stake in the litigation?*
(7) *What are the relative benefits to the parties of obtaining the data?*

The Court's recognition of sampling as an appropriate means to gauge the likelihood that discovery will prove fruitful enough to justify the attendant burden is noteworthy. Though the Zubulake court set the sample size, it left the selection of the items sampled to the party seeking discovery. While this introduces an element of happenstance, unless the tools of discovery better tame the volume, sampling is probably as sound a splitting of the baby as any other. Another notable aspect of the decision is the Court's refusal to shift the cost of privilege review to the requesting party, reasoning that the producing party is better situated to control this cost and that, once inaccessible data is restored for review, it's really no different than any other discovery materials and, accordingly, review costs would ordinarily be borne by the producing party.

"Mister Jackson! You know how I feel about sampling."

The Court did not address cost shifting when forensic intervention is sought in response to a producing party's obstructive or destructive actions, such as failing to preserve electronic evidence or affirmative efforts to eliminate same. In those circumstances, Courts are likely to visit all the costs of discovery upon the producing party but intervene to protect the rights of third-parties and preserve privilege.

The Rough Road Ahead
The next decade will see the introduction of a wondrous array of new and sophisticated technology tools and toys. Hard drives will continue to grow in capacity and drop in price per gigabyte. Two terabyte hard drives are commonplace and cost less than $100.00. Wireless connectivity will be ubiquitous and online storage and services ("cloud computing") will grow in importance. Personal digital assistants will continue to converge with cellular phones, MP3 players and global positioning devices, exemplified by Apple's iPhone and the many Android-powered devices, Low cost/high capacity solid state media will find their way into a host of new gadgets, many with unique, proprietary operating systems. We will continue to see increased reliance on and integration of computers in our lives. These machines will look less and less like our current clunky PCs--nimbler and more specialized. Greater portions of our daily lives and labors will be recorded digitally and stored online in richer media formats incorporating sound, video and location data. Paper will not disappear, but little of what we deal with on paper today will remain in paper form. Paper will become the transient medium. Encryption will be easier to use and will be built into more applications that create and store information.

It sounds pretty exciting and positive—and it is--but the dark side for litigators is that discovery of electronic evidence is not only going to become a larger part of our practice, it's going to get harder and cost more. We will be seeking discovery of data stored in cell phones, automobile dashboards and deep in the Cloud. Cherished notions of personal privacy will continue to collide with our growing ability to track, record, analyze, communicate and compile personal information. It will be challenging, to say the least, and it requires lawyers to cultivate an understanding of technology as never before; but, if you've read this far and "get it" (or most of it), you're someone who can turn the coming challenges into opportunities.

CRAIG BALL

Trial Lawyer & Special Master
Computer Forensic Examiner
Author and Educator

3723 Lost Creek Blvd.
Austin, Texas 78735
E-mail: craig@ball.net
Web: craigball.com

Lab: 512-514-0182
Mobile: 713-320-6066

Craig Ball is a Board Certified trial lawyer, certified computer forensic examiner and electronic evidence expert He's dedicated his career to teaching the bench and bar about forensic technology and trial tactics. After decades trying lawsuits, Craig limits his practice to service as a court-appointed special master and consultant in computer forensics and e-discovery. A prolific contributor to educational programs worldwide--having delivered over 700 presentations and papers--Craig's articles on forensic technology and electronic discovery frequently appear in the national media. He writes a monthly award winning column on computer forensics and e-discovery for Law Technology News and Law.com called "Ball in your Court." Craig Ball has served as the Special Master or testifying expert on computer forensics and electronic discovery in some of the most challenging, front page cases in the U.S. Named as one of the Best Lawyers in America and a Texas Superlawyer, Craig is a recipient of the Presidents' Award, the State Bar of Texas' most esteemed recognition of service to the profession and of the Bar's Lifetime Achievement Award in Law and Technology.

EDUCATION

Rice University (B.A., 1979, triple major); University of Texas (J.D., with honors, 1982); Oregon State University (Computer Forensics certification, 2003); EnCase Intermediate Reporting and Analysis Course (Guidance Software 2004); WinHex Forensics Certification Course (X-Ways Software Technology 2005); Certified Data Recovery Specialist (Forensic Strategy Services 2009); numerous other classes on computer forensics and electronic discovery.

SELECTED PROFESSIONAL ACTIVITIES

Law Offices of Craig D. Ball, P.C.; Licensed in Texas since 1982.
Board Certified in Personal Injury Trial Law by the Texas Board of Legal Specialization
Certified Computer Forensic Examiner, Oregon State University and NTI
Certified Computer Examiner (CCE), International Society of Forensic Computer Examiners
Admitted to practice U.S. Court of Appeals, Fifth Circuit; U.S.D.C., Southern, Northern and Western Districts of Texas.
Member, Editorial Advisory Board, Law Technology News and Law.com (American Lawyer Media)
Board Member, Georgetown University Law School Advanced E-Discovery Institute and E-Discovery Academy
Board Member, International Society of Forensic Computer Examiners
Member, Sedona Conference WG1 on Electronic Document Retention and Production
Member, Educational Advisory Board for LegalTech (largest annual legal technology event)
Special Master, Electronic Discovery, numerous federal and state tribunals
Instructor in Computer Forensics and Electronic Discovery, United States Department of Justice
Lecturer/Author on Electronic Discovery for Federal Judicial Center and Texas Office of the Attorney General
Specialist Lecturer, Electronic Discovery, University of Texas School of Law faculty
Instructor, HTCIA Annual 2010, Cybercrime Summit, 2006, 2007; SANS Instructor 2009, PFIC 2010, CEIC 2011
Special Prosecutor, Texas Commission for Lawyer Discipline, 1995-96
Council Member, Computer and Technology Section of the State Bar of Texas, 2003-2010
Chairman: Technology Advisory Committee, State Bar of Texas, 2000-02
President, Houston Trial Lawyers Association (2000-01); President, Houston Trial Lawyers Foundation (2001-02)
Director, Texas Trial Lawyers Association (1995-2003); Chairman, Technology Task Force (1995-97)
Member, High Technology Crime Investigation Association and International Information Systems Forensics Assn.
Member, Texas State Bar College; Life Fellow, Texas and Houston Bar Foundations
Member, Continuing Legal Education Comm., 2000-04, Civil Pattern Jury Charge Comm., 1983-94, State Bar of Texas
Adjunct Professor, South Texas College of Law, 1983-88