RITSUMEIKAN

デジタル・フォレンジック

enPiT Proセキュリティ: ProSec

セキュリティリテラシー

サイバーセキュリティ

立命館大学情報理工学部　上原哲太郎

デジタルの世界は監視社会

- ITの世界になって、
  人々の行動記録は残りやすくなった
- 例えば昔、人が電話で会話していた頃、
  会話の記録は全く残らなかったが・・・・
- メールがアタリマエの時代になって、
  全ての会話の記録が残るようになった
- 特に企業内不正・組織不正では絶大な威力
  - エンロン事件など

しかしデジタル証拠は…

- 電磁的証拠が犯罪や不正と関わることは増えて集めやすくなったが・・・
- 電磁的証拠は改ざんが容易であることは誰もが知っている
- 実際，被疑者が
  『病的に用心深く』『技術があれば』
  不正調査を攪乱させることが出来る…
  特に怖いのが**冤罪**

典型的事件：遠隔操作ウィルス

- 「主な」事件で５つからなる（すべて2012年＝平成２４年）
  - 6月29日　横浜小学校襲撃予告事件（CSRFによる冤罪）
  - 7月30日　ヲタロード無差別殺人予告事件（逮捕→起訴→釈放）
  - 8月27日　福岡幼稚園・タレント事務所襲撃予告事件（逮捕→釈放→再逮捕→釈放）
  - 9月11日　伊勢神宮襲撃予告事件（逮捕→被疑者の主張によりiesys発見→9月21日釈放）
    - この他に2件ほど新聞報道されていない事件がある
    - 最初の横浜事件を除いては同一PCで複数回の犯罪予告がされている
- 遠隔操作ウィルス事件
  - 10月9日　落合弁護士に「真犯人メール」届く　翌日報道各社にも
  - 11月13日　自殺予告メール（まどか首つりメール）
  - 1月1日　謹賀新年クイズ（雲取山メール）
  - 1月5日　再チャレンジメール
  - 2月10日　容疑者逮捕
  - 5月18日　「真・真犯人メール」→自供へ

詳しくは…

- PC遠隔操作事件
  神保哲生著
  光文社（2017．5）
  ISBN978-4334979058

被疑者だって電磁的記録の「立証が難しい」ことを利用する

- 京都大学研究員不正アクセス事件公判でのやりとり
  - 検察官「事実関係については争わないんですね？」
  - 研究員「はい、争いません」
  - 「ではASKACCSのサーバログに残っているあなたのアクセス時間と回数については、起訴状別紙の通りで間違いないですか」
  - 「覚えてません」
  - 「でもサーバログに残ってるんですよ？　アクセスログが記録されてるのは知ってますか？」
  - 「それは知ってますが、**そのログが真正かどうかはわかりませんから**」
  - 「じゃあ**ログが真正だと認められれば**、内容を認めるんですね？」
  - 「それは……**真正だと立証されたら**、お答えしたいと思います」

佐々木俊尚の「ITジャーナル」『検察官を翻弄しまくったoffice氏の奮闘』
http://blog.goo.ne.jp/hwj-sasaki/e/cf7e7ee952cc871948b78f3a1206a6b4

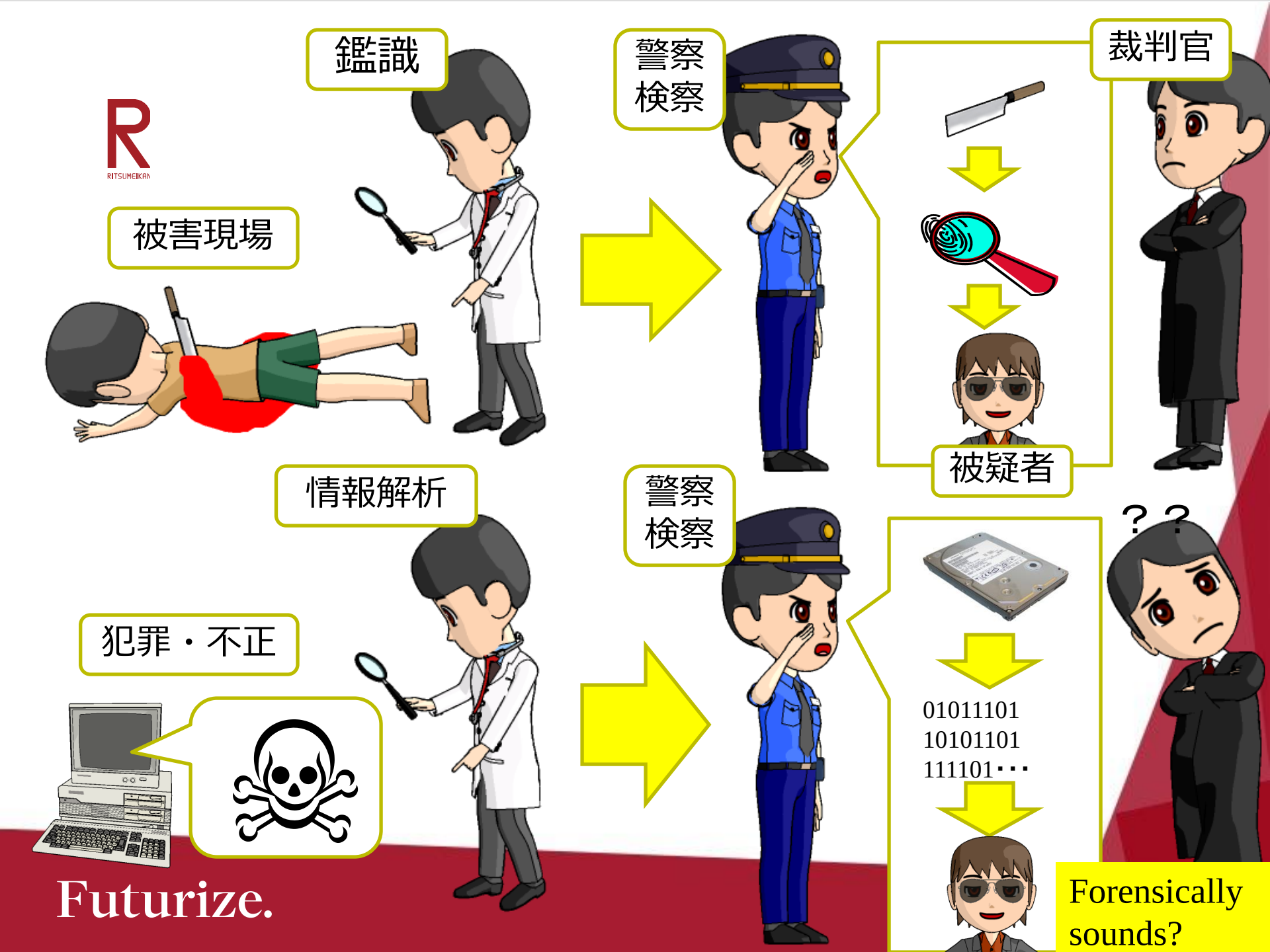
なぜこうなるのか？

- 他の証拠だってこういう争いは有り得る
- だが電磁的証拠は・・・
  - 偽造が圧倒的に容易であることが広く知られている
  - 証拠の「鑑識」にあたる部分が広く認知された職業ではない→そこに微妙に信用がない

- そこで・・・
  - 質と量で証拠が偽造される余地がなかったことを示す
    - 質：プロセスが適切であったことを示す
    - 量：複数の証拠の間に矛盾がないことを示す
  - その証拠の評価に耐える理論を構築するのが『デジタル・フォレンジック』という学問

デジタルフォレンジックとは何か

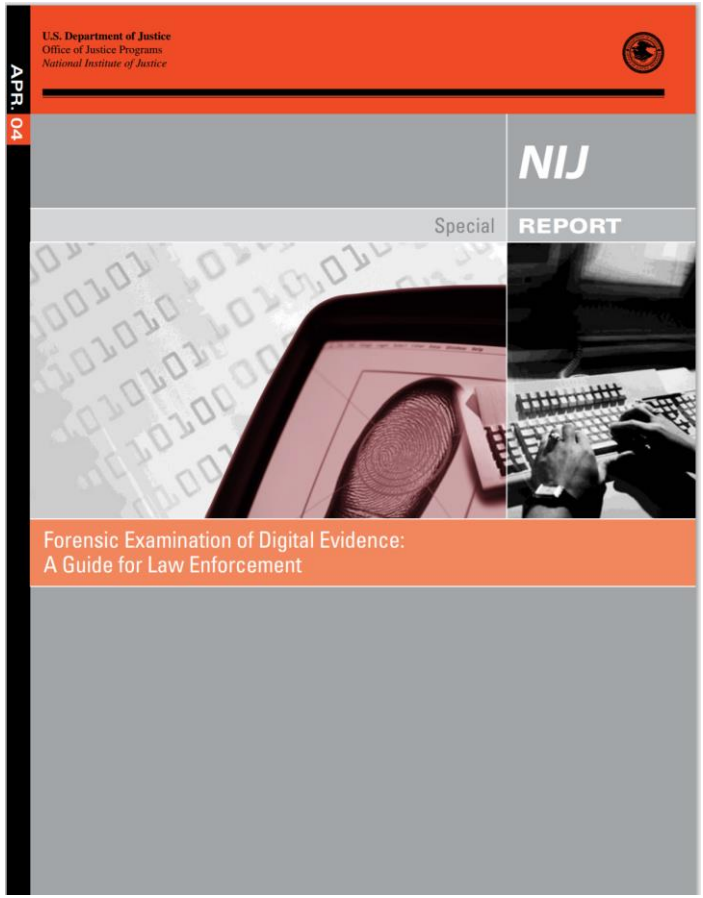
- Forensic Medicine＝法医学
  Forensic Chemistry＝法化学
  Digital Forensics＝情報法科学？
- 警察的な文脈ではいかにIT機器を調査して『電磁的証拠(e-Evidence)』を見つけ出し、いかに解析するか
  - 民間の文脈では組織内不正調査のため以外に、その予防策としての『記録保持』を含む


鑑識
被害現場
警察
検察
裁判官
被疑者
情報解析
犯罪・不正
警察
検察
01011101
10101101
111101・・・
？？
Futurize.
Forensically sounds?

デジタルフォレンジックの基本

- 「証跡収集」のプロセスを厳格にする
  - 手順を可能な限り「標準化」
- 「標準」の手順を経たことを客観的に検証可能にする
  - ドキュメンテーション・写真・ビデオ・録音…
- 「証拠」の評価を客観的に検証可能にする
  - システムとの関係・データ

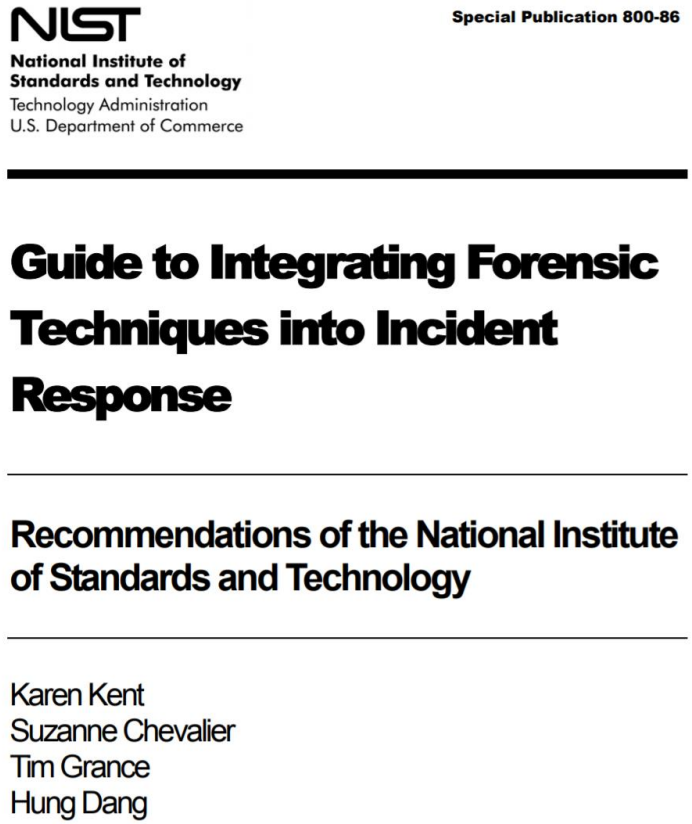
米国でのガイドライン



- 米国司法研究所(NIJ)の報告書 Forensic Examination of Digital Evicence: A Guide for Law Enforcement
- 初版は1994年 最新でも2004年

NIST SP800-86

Guide to Integrating Forensic Techniques into Incident Response

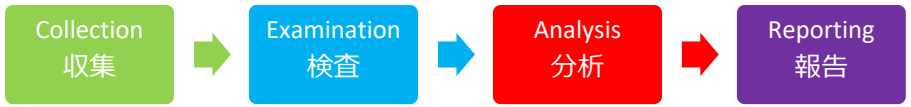

NIST
National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

Special Publication 800-86

Guide to Integrating Forensic Techniques into Incident Response

Recommendations of the National Institute of Standards and Technology

Karen Kent
Suzanne Chevalier
Tim Grance
Hung Dang



- 収集
- 検査
- 分析
- 報告

この4段階を定義

ISO/IEC 27037:2012

Information technology -- Security techniques -- Guidelines for identification, collection, acquisition and preservation of digital evidence

Abstract

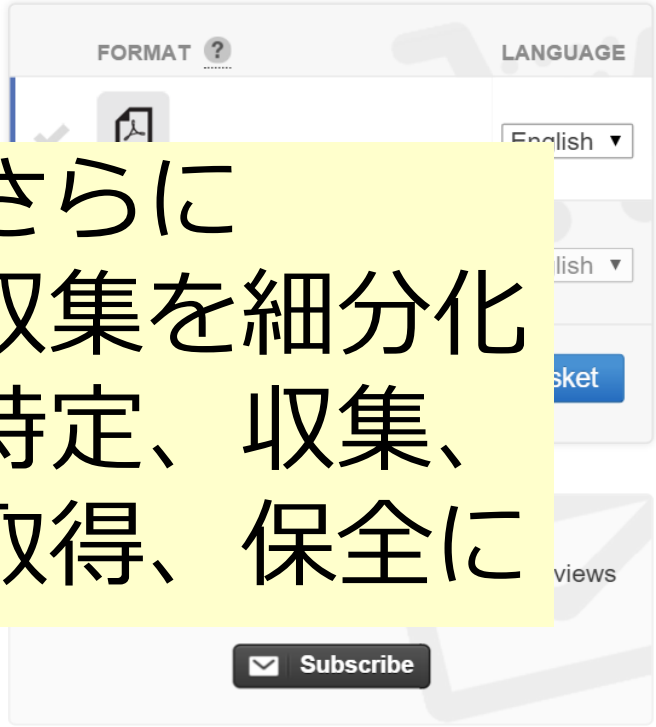
Preview ISO/IEC 27037:2012

ISO/IEC 27037:2012 provides guidelines for specific activities in the handling of digital evidence, which are identification, collection, acquisition and preservation of potential digital evidence that can be of evidential value.

It provides guidance to individuals with respect to common situations encountered throughout the digital evidence handling process and assists organizations in their disciplinary procedures and in facilitating the exchange of potential digital evidence between jurisdictions.

ISO/IEC 27037:2012 gives guidance for the following devices and circumstances:

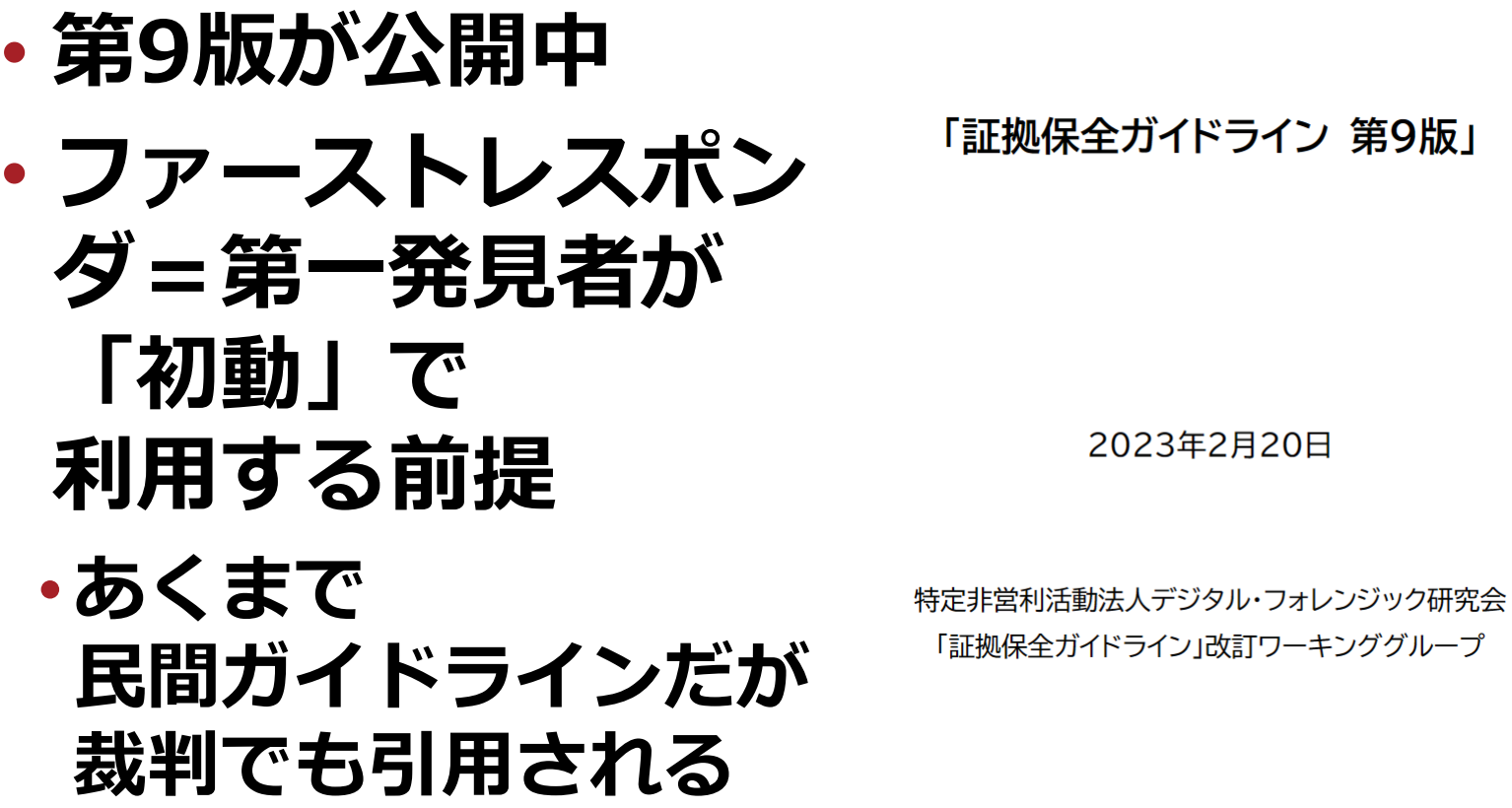
- Digital storage media used in standard computers like hard drives, floppy disks, optical and magneto optical disks, data devices with similar functions,
- Mobile phones, Personal Digital Assistants (PDAs), Personal Electronic Devices (PEDs), memory cards,
- Mobile navigation systems,
- Digital still and video cameras (including CCTV),
- Standard computer with network connections,
- Networks based on TCP/IP and other digital protocols, and
- Devices with similar functions as above.



さらに
収集を細分化
特定、収集、
取得、保全に

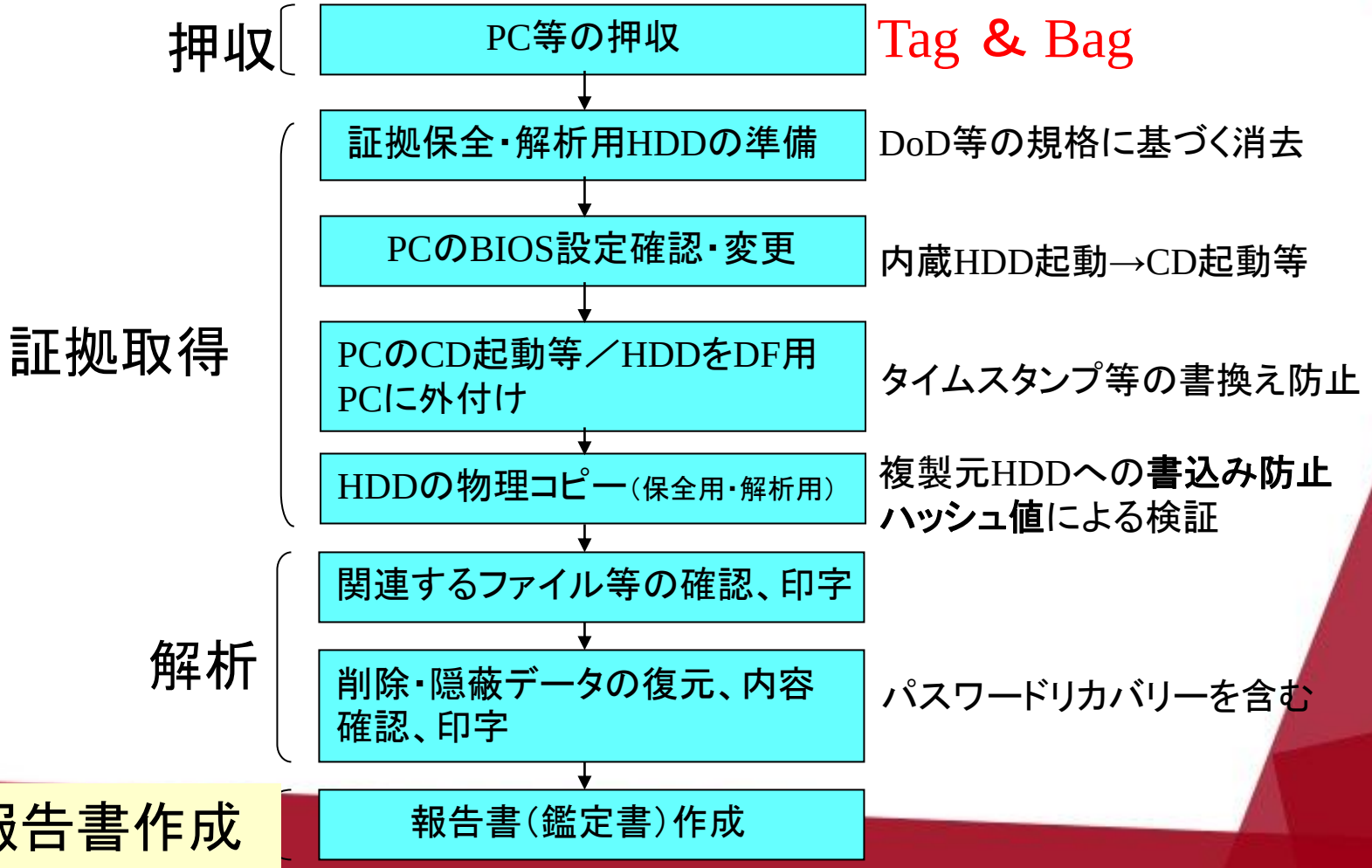
デジタルフォレンジック研究会「証拠保全ガイドライン」

- 第9版が公開中
- ファーストレスポンダ＝第一発見者が「初動」で利用する前提
  - あくまで民間ガイドラインだが裁判でも引用される



https://digitalforensic.jp/wp-content/uploads/2023/02/shokohoznGL9.pdf

基本的なプロセスは「現状保存」とハードディスクの複製、内容解析




09
05
06
10
07
11
Trident

問題は電源が入っていたときや「スリープモード」だったとき

- プログラムをはじめとする証拠が「メモリ上」にある
  - 具体的には「稼働中のプログラム」「閲覧中のWebページ」「作成中のデータ」などなど
  - 電源を切る/コンセントを抜くとこれが消える可能性
- なので（現状保存の原則には反するが）正しいプロセスでPCを操作し、記録しつつ電源を切る前に集められる証拠は集める必要がある→ここに議論の余地

証拠が固められたらどうする？

- 次は中で「何が起きていたのか」を探る
- 証拠に関わるファイルについて・・・
  - いつどのように作られたか
  - どう参照されたか　どう変更されたか
  - いつどう消されたか　→　ファイル復活が必要

を整理し，時系列にまとめる必要がある
これを「タイムライン解析」という

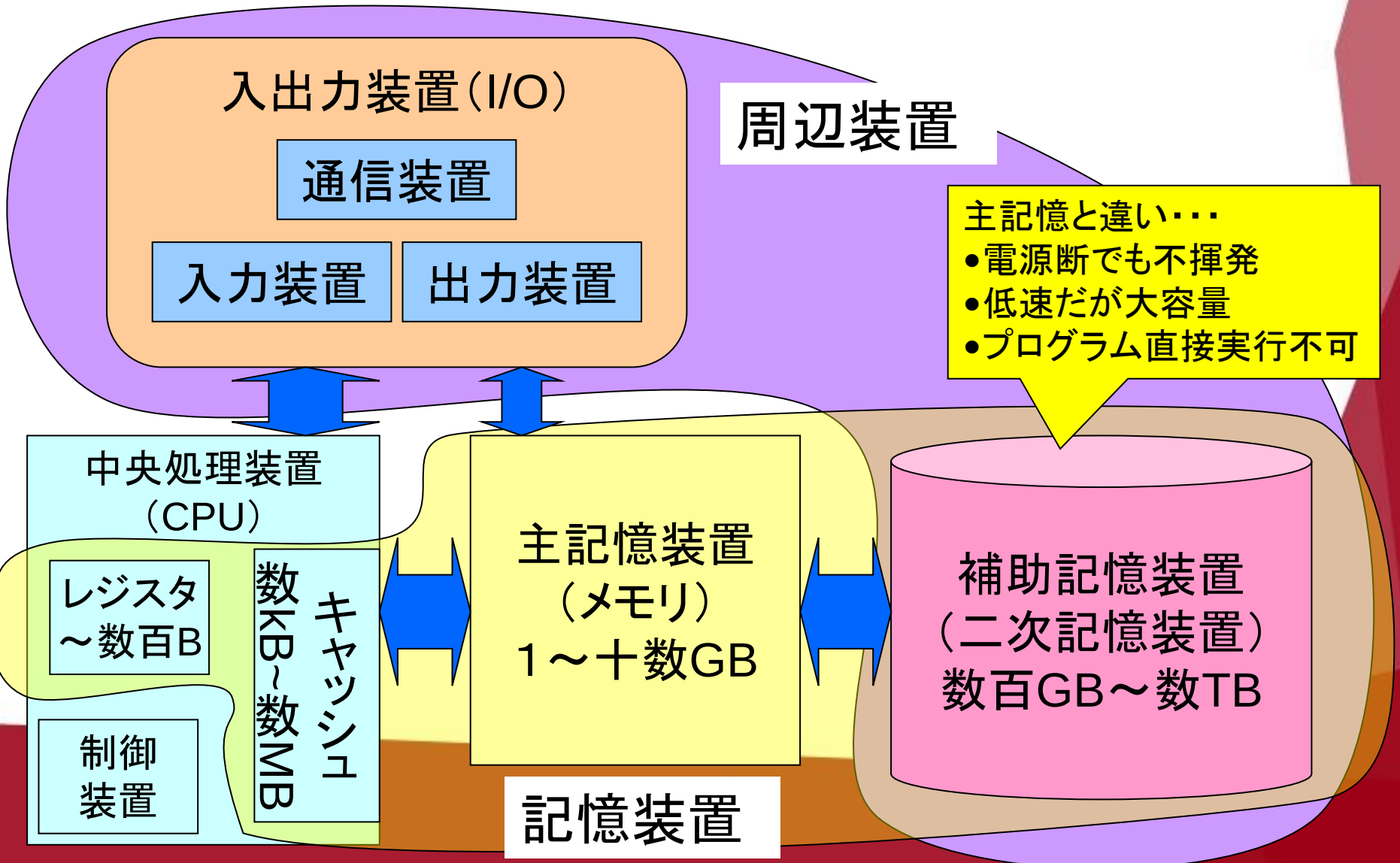
タイムラインを綺麗に整理しておくことが後の調査に役立つ

そのためファイルシステムの理解が必要

ファイルシステムと デジタルフォレンジック

- デジタル証拠を集める立場からは・・・
  - 今見えている「ファイル」は本当に全てか？
    隠蔽された/消去されたファイルはないか？
    - 消去ファイル内の内容は復元可能か？
      復元したものはどれだけ信頼できるか？
  - ファイルのメタ情報からなにがわかるか
- PCリサイクルやメディア破棄の立場からは・・・
  - 消去は完全か
    そこから情報漏えいの危険はないのか

二次記憶（補助記憶）とは？


入出力装置(I/O)
通信装置
入力装置
出力装置
周辺装置
主記憶と違い・・・
•電源断でも不揮発
•低速だが大容量
•プログラム直接実行不可
中央処理装置
(CPU)
レジスタ
～数百B
キャッシュ
数KB～数MB
制御
装置
主記憶装置
(メモリ)
1～十数GB
補助記憶装置
(二次記憶装置)
数百GB～数TB
記憶装置

二次記憶の分類

- ランダムアクセス可能か否か
  - ランダムアクセス可能なものは「ファイルシステム」を構築することができる＝ファイルが置ける
  - 後者はテープなど
- 書き換え可能か１回のみか
  - 特殊な場合として「ユーザ書き換え不能」(ROM)
- リムーバブルか否か

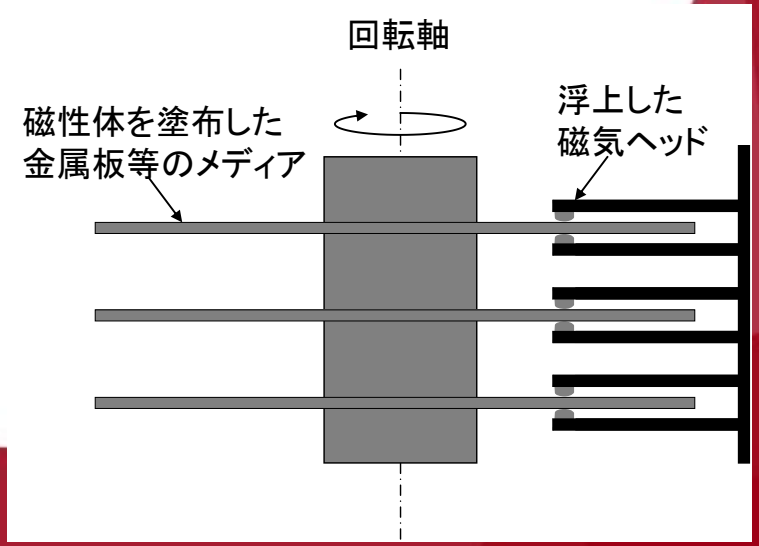
代表的二次記憶媒体

- 磁気記録
  - ハードディスク　（媒体：ガラス＋磁性体など）
  - 他にフロッピーディスク、テープなど
    - 磁気に弱い　熱にも弱い
- 光による記録
  - CD-R/RW/RAM, DVD-R/RW/RAM，BD
    - 最近の低価格化に伴い比較的消えやすい
  - 亜種として光磁気記録（MO）
    - 保存性はかなり高い
- 半導体記録
  - フラッシュメモリ（USB、ケータイ・デジカメ用カード、SSD等）
    - 電気的に壊れることがある
    - 最近保存性能劣化が激しい
      - 書き換え可能回数が数千回から数百回に
      - 年単位の保存は難しい（自然とデータが消える）

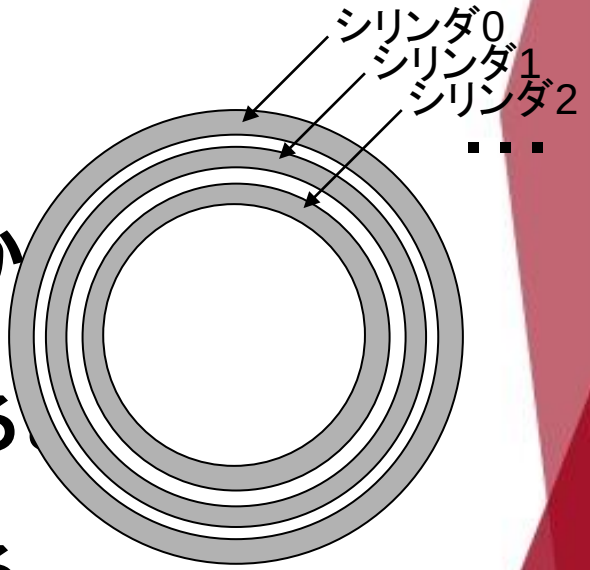
評価項目
- 読み書き速度
- 書き換え可能回数
- 磁気/熱/電気/光などに対する耐性
- 経年劣化への耐性

ハードディスク

- 高速大容量なので
  最も重要な二次記憶装置
- 金属かガラスの円盤(プラッタ）に磁性体を塗り
  磁気ヘッドで読み書き
  - 埃に弱いので通常密閉されている
  - 円盤の直径によって
    3.5inch,2.5inchなどと呼ばれる
  - ドライブ厚みは3.5inchは1cm
    2.5inchは12.5mm, 9.5mm
    7mmなどがある
  - 現在3.5inchで最大20TB
    2.5inchで最大4TB



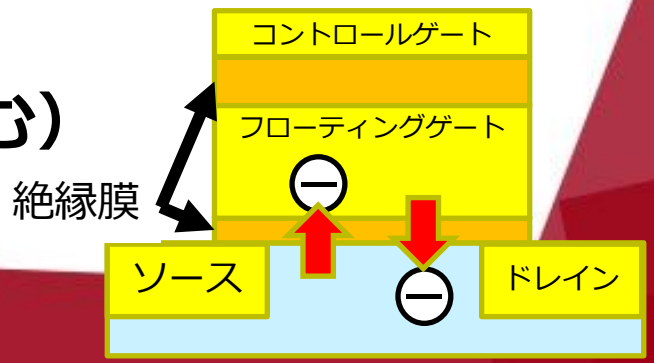
ハードディスクの論理的構造



- 同心円状に分けられた
  シリンダが、さらにセクタという
  固定長の単位（多くは512バイトか
  4096バイト）に分割された構造
- 中の構造はともかく、外部から見る
  単に『セクタ』を単位に自由に
  読み書き可能な磁気記録媒体である
  - 最近は4096バイト＝4kバイトの媒体が多い
  - セクタには通し番号がついており、
    番号を用いて読んだり書いたりする
  - 連続した番号のセクタのほうが読み書きが速いなどの
    特徴はあるが、基本的に読み書き制限はない

フラッシュメモリ

- 「電気的に書き換えが可能なROM」の一種
- 一応メモリではあるがRAMよりはるかに読み書きが遅い（特に書き込みは大変遅い）
  - そこで機械語を書き込んで直接メモリとして利用する時は内容を一度RAMにコピーしてから使う（シャドウ・メモリ）
- それでもハードディスクよりははるかに速いので二次記憶として使う例が増えている
  - SSD(ハードディスクの代わり）
    USBフラッシュメモリ,
    SDカード（microSDなども含む）

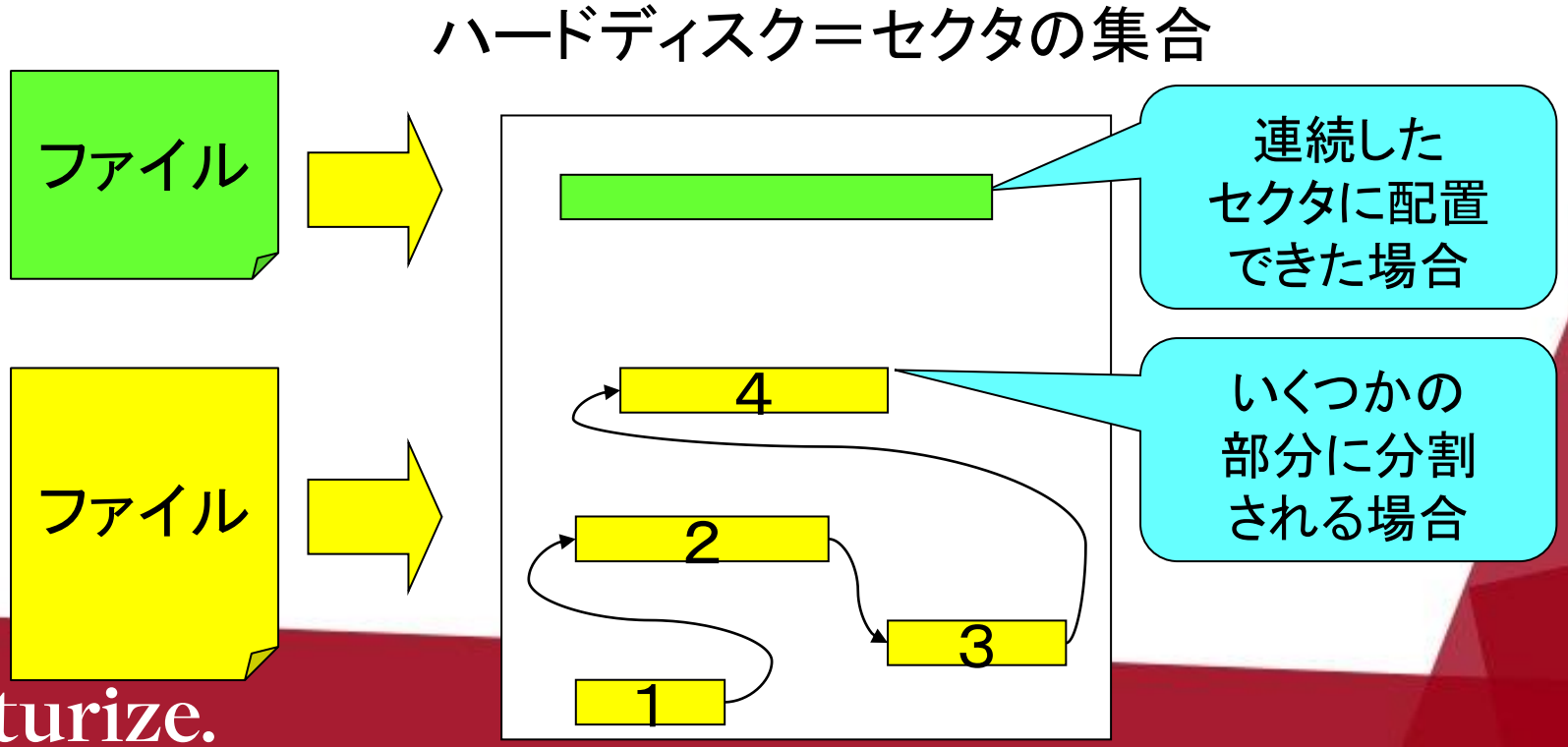


二次記憶としての フラッシュメモリの注意点

- 読み込みより書き込みが極めて遅い
  - 書き込み前に媒体を「消去」する必要がある
    - 読み書きは「ページ」単位
      消去はページをまとめた「ブロック」という単位で行われる
- 書き込み可能回数に制限がある
  - 書き込むたびに絶縁膜が劣化するので
    そのうちデータが保持できなくなる
    - 今は1セルあたり数十回になりつつある
    - セルあたりのビット数が増えると書き込み回数がさらに減る
      - SLCは1ビットだがMLCでは2～4ビット
    - そこで各セルの書き込み回数を平準化するために
      論理セクタと物理ページを分離し、物理ページ毎に書き込み回数が
      偏らないように平均化する手法が使われる（ウェアレベリングと呼ぶ）
- 時間と共にデータが消える
  - 年単位で見ると少しずつデータが消失する
    - その速度は絶縁膜の劣化具合によるので予測が困難

ファイルシステムとは

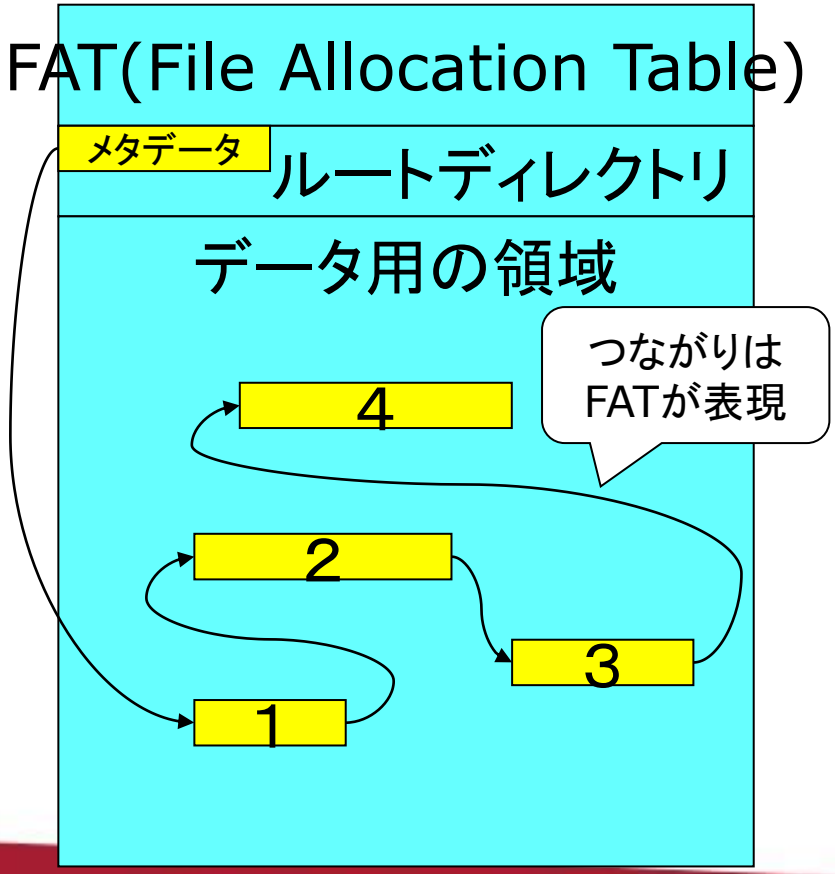
- ファイルという任意長のデータを、固定長のセクタの集合体であるメディアに分割して配置する仕組みを提供
- OSによっては複数提供されている
  - Windows→FAT, NTFSなど MacOS X→HFS, UFSなど



ファイルシステムが扱うもの

- ファイルを「メタデータ」と「データ本体」に分けて扱う
  - データ本体＝バイト列（テープのようなもの）
  - メタデータ＝ファイルについて以下の情報を扱う
    - ファイル名
    - ファイルの配置情報（メディア内のどのセクタか？）
    - ファイルの長さ（バイト数）
    - タイムスタンプ（作成日時・更新日時・最終アクセス日時など）
    - アクセス権に関する情報（所有者、読み取り権、書き込み権など）
    - その他アイコンやファイルの種類等
  - メタデータはOSにより必要要件が異なる
    同一OSでもファイルシステムによって多少異なる
    （FATよりNTFSのほうが詳細 etc）
- メディアの領域管理も行う必要
  （空き領域、メタデータ、ディレクトリetc・・・）

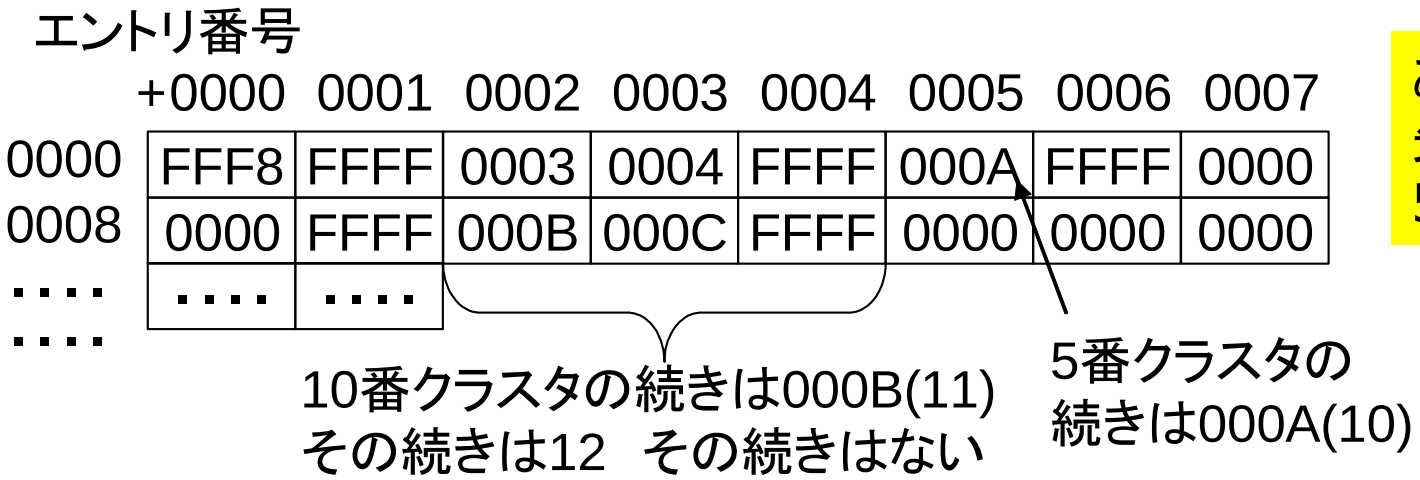
FATファイルシステムでのファイルの格納イメージ



- FATは以下のメタデータを保持
  - クラスタと呼ぶ単位で管理
  - 空きクラスタの情報
  - ファイルにおいて「どのクラスタとどのクラスタが繋がっているか」＝配置情報の一部（クラスタチェイン）
- ディレクトリは以下の情報を保持
  - ファイル名
  - タイムスタンプ
  - ファイルサイズ
  - 先頭のクラスタ番号
    - 先頭以降はFATを見ないとわからない

FATの構造

- セクタを集めたクラスタ単位で管理（1セクタ＝1クラスタの場合も多い）
- クラスタ番号１つごとに1つの「エントリ」がある（エントリ番号＝クラスタ番号）
- エントリの値が0ならそのクラスタは空き
- 空きでなければ
  「そのクラスタにあるファイルの続きはどのクラスタか」が書いてある



あるファイルの
先頭クラスタが
5番だった場合

※実際にはリトルエンディアンなので各エントリのバイトは逆順に並ぶ

FATファイルシステムにおけるディレクトリエントリの構造

- 原則として32バイト＝１エントリごとに1ファイルを表す
- 先頭の1バイトが重要な意味
  - 普段は（短い）ファイル名の先頭文字
  - ファイルが削除された場合はE5で置き換え

通常のファイルのエントリ（ファイル名の.は書き込まれない）

削除されたファイルのエントリ（先頭1文字が失われる）

長いファイル名のエントリ（1エントリで13文字ずつ逆順に格納される）

長いファイル名のエントリの実体（ファイル名は短縮して格納されている）

削除されたファイルのエントリ（LFNではファイル名は削除後も残る）

	ファイル名	属性	タイムスタンプ・ファイルサイズ・先頭クラスタ等
	TESTFILETXT	属性	タイムスタンプ・ファイルサイズ・先頭クラスタ等
E5	ELETED EXE	属性	タイムスタンプ・ファイルサイズ・先頭クラスタ等
43	ントリ.htm	0F	1
02	を持つ場合	0F	のディレクトリエ
01	とてもとて	0F	も長いファイル名
	とても~1HTM	属性	タイムスタンプ・ファイルサイズ・先頭クラスタ等
E5	ファイルの	0F	削除.DOC
E5	少し長めの	0F	ファイル名を持つ
E5	ュし長~2DOC	属性	タイムスタンプ・ファイルサイズ・先頭クラスタ等

FAT　Filesystemで格納されるメタデータ

- ファイル名
  - LFNの場合Unicodeで255文字まで
- ファイル属性
  - ファイルの種類（ファイル/ボリュームラベル/ディレクトリ）
  - 属性（読み取り専用/隠しファイル/システムファイル/アーカイブ）
- タイムスタンプ（現地時間）
  - 作成日時（0.01秒単位）：VFAT利用時のみ
  - 更新日時（2秒単位）
  - 最終アクセス日（1日単位）：VFAT利用時のみ

FATファイルシステムにおける ファイルの削除

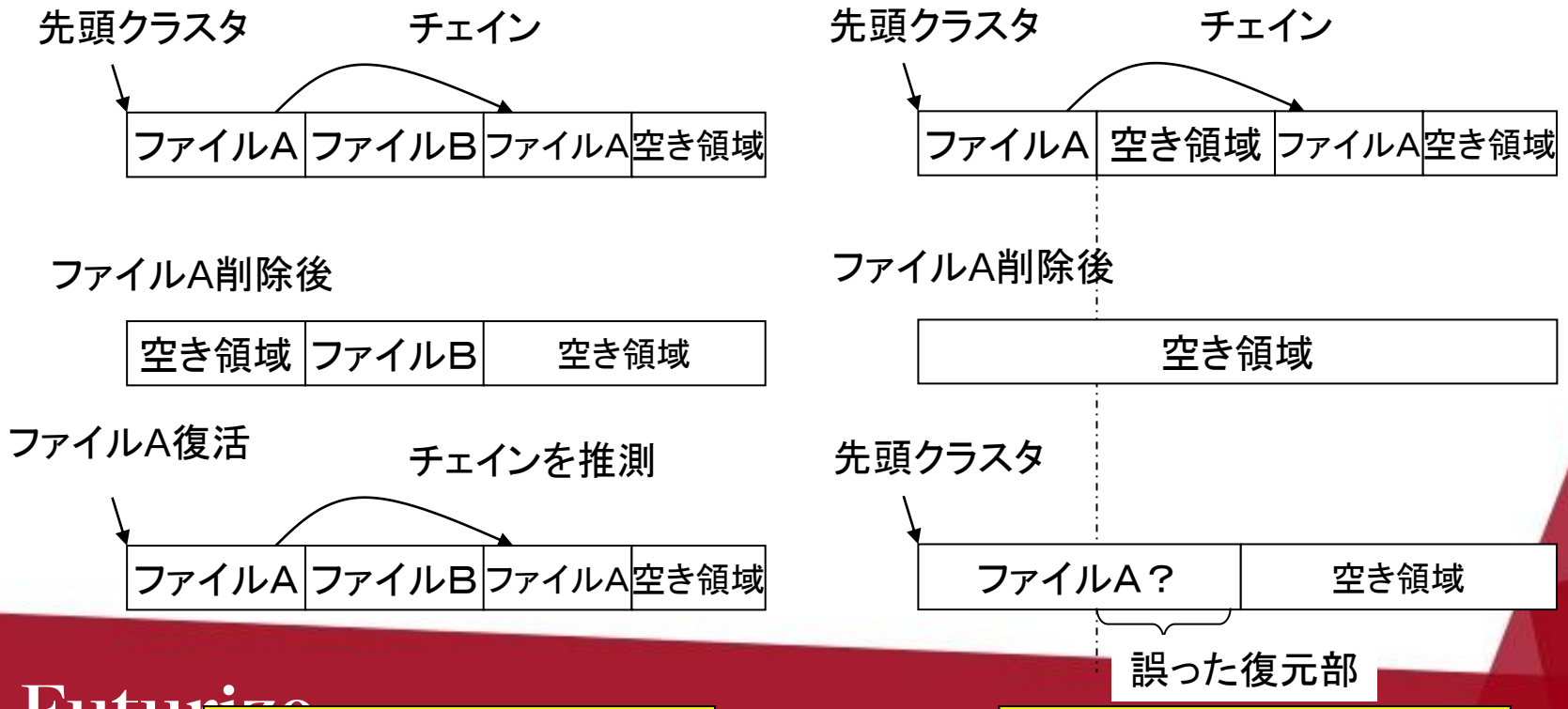
- ディレクトリエントリにおいて先頭の1バイトをE5にする（他は残す）
- FATの該当エントリを0（＝空き）にする
  - よってクラスタチェインの様子はわからなくなる ＝「ファイルの配置情報」は予測するしかない
- 以下の情報は維持される
  - ファイル名の先頭文字以外の多くのメタデータ
    - LFNの場合はファイル名も全部残っている
  - ファイルの配置情報
  - ファイルのデータ本体

FATファイルシステムにおける削除ファイルの復元

- 削除直後、まだデータ本体やディレクトリエントリが上書きされていなければ高い確率で戻ってくる
  - ファイル名の先頭文字をユーザに与えてもらう
    - 「長いファイル名」のときは与えてもらう必要すらない
  - ファイルの先頭クラスタはディレクトリエントリ内にあるのでわかる　ファイルサイズもわかる
  - よってファイルサイズ分の「空き領域」を適当に（例えばクラスタ番号の若い順に）つないで、FAT内のクラスタチェインを復元

クラスタチェインの復元がうまくいかない場合

- 上書きされていなくても、クラスタチェインの推測がうまくいかない場合がある・・・



うまく復元できる場合

誤った復元をする場合

「高度な」削除ファイル復元

- 一部のファイル復元ソフトはメタデータの残骸が信頼できない場合に備えて、データ本体のみを見てファイル復活する機能を備えている
  - File Carvingという
    - 技術的に面白いので学術研究も盛ん
- Wordのdocファイルなど多くのファイルは、ファイル内のデータ構造に特徴があるので、クラスタをつなぎ合わせてその構造が矛盾しないような組み合わせを探し、本来のデータ本体を検索していると思われる
  - 全組み合わせはとても試せないのでモデルを導入

FAT　Filesystemの種類

- FAT12，FAT16：FATの原型
  - FATが各クラスタを表現するのに12bit/16bit使っている
- FAT32
  - FATが各クラスタを表現するのに32bit使っている（実際は4bit予約されているので28bitまで）
  - Win95OSR2から利用可能
- VFAT
  - FATではなくディレクトリエントリを拡張している
  - 長いファイル名（LFN）や拡張タイムスタンプをサポート
- exFAT
  - Windows CE 6.0 / Vista SP1から導入された新しいFAT
  - 実はこれまでのFATとあまり互換性がない
- TFAT
  - Transaction-Safe FAT
    - WinCE 6.0で導入されたがその後ほぼ使われてない

FAT　Filesystemの利点・欠点

- 利点
  - 構造が簡単で実装が楽：デジカメ始めWindows / DOS以外でも広く使われている
- 欠点
  - ファイルサイズの制限（約4GB）
  - ボリュームサイズの制限（FAT32で通常2TB）
  - ファイルの所有者や権限といった情報が保存できない
  - メタデータ書き換え中に電源断が起きると復元が困難になる可能性あり
  - 速度が遅い（FATの構造に起因）

exFATファイルシステム

- ファイルの位置情報はほぼFAT32と同じ構造で記録
  - よって削除ファイルの復元の難易度はFATと変わらない
- ディレクトリエントリは全面改定
  - 32バイト単位は同じだが先頭の1バイトによって
    内部の構造が完全に変わる（よってFATと互換性なし）
    VFAT同様複数のエントリで1つのファイルを表す
  - ファイルサイズは64bit化(16EiBまで)
  - タイムスタンプはUTC基準　最終アクセスが2秒単位に
- 読み書き高速化のための工夫あり
  - ファイル検索の高速化のためハッシュ値を保持
    - この際大文字小文字正規化のためのテーブルを
      ファイルシステム上に特殊ファイルとして保持
  - 空きクラスタの情報を示すビットマップを保持
    - これも特殊ファイルとして取り扱い
- Transaction-Safeな実装あり(TexFAT)→あまり使われず
- SDメモリカードの新規格(SDXC)から利用可能

NTFSファイルシステム

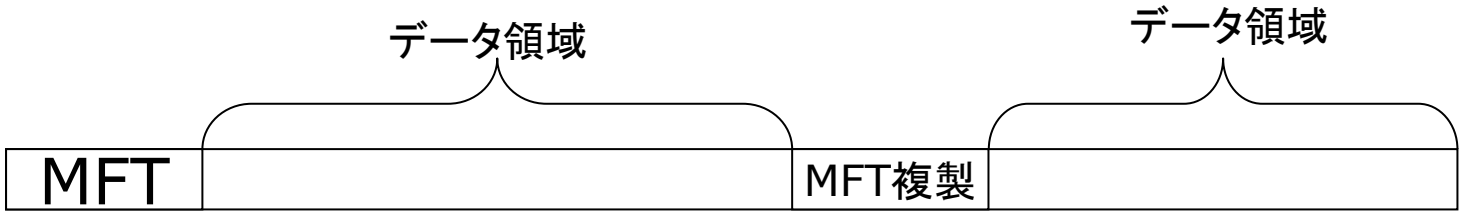
- FAT代替として開発されたファイルシステム
- 特徴：
  - メタデータを含め「全てがファイル」というデザイン
    ：単純で拡張性が高い
  - ジャーナルファイルシステムである
    - メタデータの更新前に更新を「ジャーナル」としてディスク上に
      書き込むので，不意の電源断が起きてもメタデータが壊れない
  - ファイルサイズ16EiB，ファイル数2^{32}
  - ファイルの所有者・グループなどごとに権限を付与可能
    ファイルの圧縮・暗号化・代替ファイルストリーム機能
    など
    多機能

NTFSのデータ構造

- MFT (Master File Table)と「それ以外」しかない
- MFT: １ｋＢの「ファイルレコード」が連なるテーブル
  基本的に1ファイルレコードが1ファイルを表す
  - まれに複数レコードで1つのファイルになる
- MFTはさらにAttributeによりファイル名や
  ファイル属性，ファイル位置情報が加わる
  - ファイルが十分小さいときはMFTの空き領域がデータ部に
- 特殊ないくつかのファイルで構成される
  - ファイルエントリ0の$MFTがMFT自身を表すとともに
    MFT内の空き領域を$BITMAPで表す
  - ファイルエントリ6の$Bitmapがデータ領域の空き領域を表す

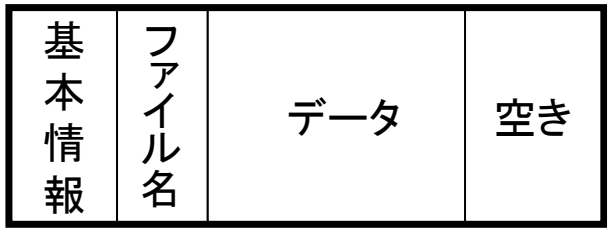
NTFSのデータ構造

NTFSパーティション構造

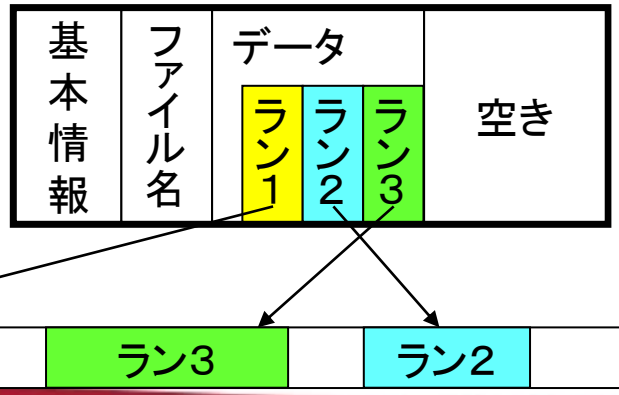


各MFTエントリの構造

小さなファイルの場合



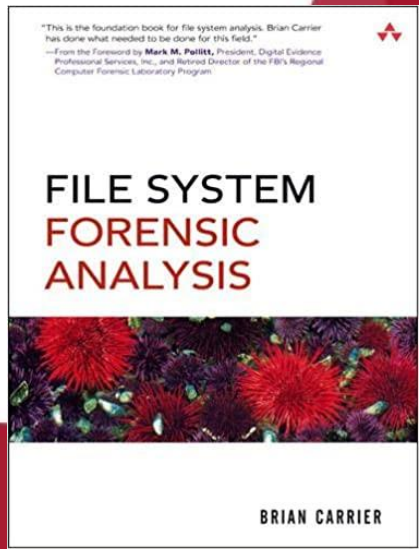
大きなファイルの場合



NTFSでのメタデータと削除

- NTFSでは、メタデータの格納域とデータ本体の格納域が分かれていて、その空き領域管理はそれぞれ独立で行われる
  - MFT(Master File Talbe)がメタデータ格納域データ本体の配置情報も含め全てをあらわす
  - ファイル削除時は双方の空き領域管理の仕組み(Bitmap)だけが操作される
    ＝メタデータもデータ本体も削除直後全て残る
  - よってファイルの復活はFATよりも楽

2005年の本だが今でも役立つ



他の「削除」は？

- 「ゴミ箱に入れる」→削除といえない
  - 単にファイルシステム内で名前/場所が変わるだけなのですぐに復元可能
    - 「ゴミ箱を空にする」でやっと普通の削除
  - 一部のソフトウェアで、自動バックアップ機能として「削除」操作をしてもゴミ箱内に残すものもある
- 「フォーマットする」→場合による
  - 多くの場合メタデータは初期化されるのでファイル復元は難しくなりそうだが、データ本体を上書きしない場合があるので「高度な復元」が可能
- 「パーティション削除」→削除といえない
  - ファイル復元と同様に簡単に復元可能

ファイルシステムの削除ではないが

- メールの削除・データベースの削除など
  - データベースファイルの内部は
    ファイルシステムと同様に構造がある
  - データベースファイルの構造によって
    削除の難度が異なる
  - しかし有名ソフトウェアに対してはツールがある
- ファイルの「上書き」
  - 一般にはこれが一番復活しづらい
    - データ本体が損傷を受ける場合が多いため
  - ただ、アプリケーションによる上書きは
    古いデータ本体が残っている可能な場合もある
  - ファイルの「上書き」のやり方によっては復元可能
    - 別のファイルを同一ファイル名に変更して上書きなど

復元できない確実な削除のためには

- データ本体を上書きする必要がある
- ハードディスクドライブ全体で行う場合
  - ファイルシステムを完全に上書きするツールを使用
  - 暗号化されている場合は暗号鍵を飛ばせば良い
- ファイル単体で行う場合
  - 削除するファイルを、削除前に無意味なデータで上書きする
    - 例：「安全な削除」MacOS、 Linuxのsrmなど
  - ただしこれで本当に確実に上書きされるかは議論あり
    - ファイルシステムが上書きを避けている可能性あり

「上書き」は本当に安全か？

- 1996 P.Gutmann 「上書き後も磁気面に残った情報から以前のデータが読み取れるよって何度も上書きして完全消去の必要あり」と指摘
  - しかし当時のハードディスクはMFMやRLLが主流現在はPRML系なので以前より読み取りは困難それに読み取りの設備もまだまだ高価
    →通常の企業や家庭では単なる上書きで十分
    より機密度が高くても数回上書きでよいと予想できる（ただし検証要）
  - より安全な消去のためにATAにSecure Eraseコマンドが定義されており現在のドライブは対応

  →ヘタに上書きするよりSecureEraseを実行する方が良い

現在役に立つ資料

PUBLICATIONS

SP 800-88 Rev. 1

Guidelines for Media Sanitization

Date Published: December 2014

Supersedes: SP 800-88 (09/01/2006)

Author(s)

Richard Kissel (NIST), Andrew Regenscheid (NIST), Matthew Scholl (NIST), Kevin Stine (NIST)

Abstract

Media sanitization refers to a process that renders access to target data on the media infeasible for a given level of effort. This guide will assist organizations and system owners in making practical sanitization decisions based on the categorization of confidentiality of their information.

Keywords

media sanitization; ensuring confidentiality; sanitization tools and methods; media types; mobile devices with storage; crypto erase; secure erase

Control Families

Maintenance; Media Protection; Risk Assessment

Futurize.

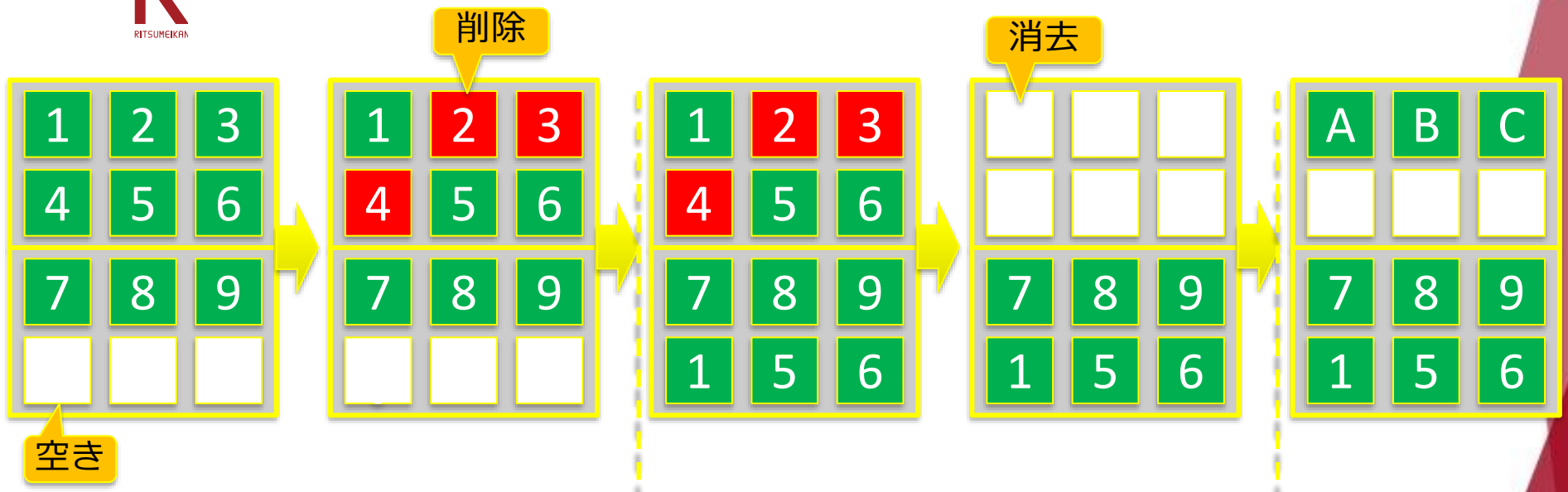
物理的破壊からの復元

- ハードディスクはプラッタが無事なら復元の可能性は高い（部品交換など）
  - クリーンルーム作業が必要
  - 多くの場合は物理的衝撃
    でもプラッタは無事　壊れるのはヘッド
- 逆に確実な消去のためには
  プラッタの物理的/電磁気的破壊が手っ取り早い

さらに最近の話題

- 暗号化しておいて「鍵を捨てる」＝**暗号化消去**
  - ハードディスク自体に暗号化機能をつけておく
  - 鍵はコマンド一発で消去できる→安全な消去
  - フォレンジック的には非常な困難
- SSDのTrim機能
  - NAND型フラッシュメモリを用いたSSDは HDDをエミュレートしている
  - 媒体の書き込みは「ページ」単位（16～32kB）かつ 書き込み前に「ブロック消去」が必要（ページより大きい）
  - そこで、ファイル消去時、不要になったセクタを OSがコントローラに伝え、上書きに備えて あらかじめ消去させる仕組みが生まれた（Trim）
  - よってファイル消去後の復活が困難に

SSD Trimの仕組み



Trimなしでは
ファイルを削除した
段階ではここまで

Trimありでは
ファイルを削除した
段階でここまで！

自己暗号化ドライブ (SED) Self Encrypting Drive

- 暗号化消去の最大の課題は鍵管理
  - 暗号鍵が消えると読めなくなるためよく「鍵のバックアップ」が推奨される
    - 例えばWindowsのBitLocker技術
  - しかしバックアップを取ると逆に消去の際にはバックアップ鍵も消去する必要
- SEDはHDDコントローラ内で鍵管理
  - Trusted Computing Group(TCG)のOpal SSC
  - MicrosoftのeDrive
  - 各社独自規格(Opalへの置き換え進む)
- 暗号アルゴリズムはIEEE 1667で定義
  - 暗号モードとしてXTSを用いる

ライブフォレンジック

- 最近は二次記憶のデータだけでは不十分
  - 「ファイルレスマルウェア」
  - 二次記憶の暗号化（一度シャットダウンすると二度と回復できない）
- そこで動作中の主記憶の内容をできるだけ保全しておく必要がある
  - ライブフォレンジックと呼ばれる

プロセスのダンプ

- ライブフォレンジックの必要から
  特定の実行中のプロセスの仮想メモリ空間を
  全てファイルに取り出したい場合がある
  ⇒プロセスのダンプ
- Windowsではprocexpやprocdumpなどが使える
  - 生成されたダンプファイルはWinDbgなどで見られる
- UNIX系ではプロセスダンプはcoreと呼ばれる
  - プロセスにQUITシグナルを飛ばすと強制終了するとともに
    coreダンプが生成される(kill –QUIT pid)
  - ダンプの最大サイズはulimitなどで設定
    ダンプ生成場所はkernel parameterで設定
  - 生成したcoreはdbgやgdbなどのデバッガで見られる

クラッシュダンプ（カーネルダンプ）

- OSが異常終了したときは物理メモリの内容がファイルに自動的にダンプされる
  - Windowsではいわゆる「青画面」の時に生成
  - 生成後はカーネルデバッガで内容を見られる
- Windowsにおいて強制的にカーネルダンプを得るには、レジストリに適切に設定の上Ctrl+Scrollキーを押す
  - USBキーボードとPS/2キーボードで異なるので注意
- Linuxにおいてはkdumpサービスを利用

アプリケーション一時ファイル

- アプリケーションが生成する一時ファイルも
  フォレンジック上は大きな手がかりになる
- 一時ファイルが生成される場所は決まっている
  - Windowsでは環境変数TEMPやTMPの場所
    - コマンドプロンプトからsetで確認
    - 設定したければコントロールパネルから編集できる
  - UNIXでは多くの場合/tmp, /var/tmp,
    /usr/tmp
    まれに環境変数TEMPの場所
- 一時ファイルの内容はアプリケーション依存
  - Office系ソフトウェアの場合は解析が進んでいる
    （自動保存や強制終了時の編集中文書の内容）

ログについて

- ログ(log)：元々は「丸太」の意味
  OSやアプリが管理上の必要から残す「記録」
  - フォレンジック上は非常に重要
- OSが集約するものとアプリ独自のものがある
- 基本的にはアプリケーションやOSが自発的に
  残すものなので最初からその機能を備える必要
  - 設定で「ログレベル」を変更できる場合はある
- Windowsでログが残る場所
  - イベントログ
  - レジストリ
  - 独自ファイル
- UNIXでログが残る場所
  - Syslogシステム
  - 独自ファイル

Windowsのイベントログ

- Windowsシステムサービスの「イベントサービス」が各プロセスやOSからイベントを収集して記録
  - ファイルの実体は*.evt *.evtx 場所はバージョン依存
  - 複数のファイルに別れており、設定した量を超えると古いものは捨てられる
  - ログサーバに集約する機能も有する
- イベントビューアーで見られる
  - アプリケーションログ
  - セキュリティログ
  - システムログ
  - その他
- レベルは「情報」「警告」「エラー」の3段階
- システムの起動日時、アプリケーションの起動時など雑多な情報が得られる

Windowsレジストリ

- 元はWindowsのOSやアプリケーションが設定情報を格納するためのデータベース
  - システム全体は¥windows¥system32¥config¥にsam, sam.sav, sam.logなどいくつかのファイルに別れて格納されている
  - ユーザごとの情報は¥Windows¥Profiles¥ユーザ名にNtuser.dat, Ntuser.dat.logなどに別れて格納されている
  - 内容はファイルシステム的だが格納できるデータ形式に「型」がついており制限がある
- ¥HKEY_CLASSES_ROOT
  ¥HKEY_CURRENT_USER　（よく¥HKCUなどと略す）
  ¥HKEY_LOCAL_MACHINE
  ¥HKEY_USERS
  ¥HKEY_CURRENT_CONFIG
- この中にいくつかログ的なものがある
  - 例えば¥HKCU¥Software¥Microsoft¥Windows¥Currentversion¥Explorer¥に最近使ったファイル等の情報が残存

UNIX syslog

- syslogは信頼性も秘匿性もないため
  各種新提案があるが標準化されていない
  （旧syslogはRFC3164⇒RFC5424で標準化）
- UDPでログサーバに集約する機能を有する
- FACILITY値が0-23で決まっており0がカーネル、
  1がユーザ、2がmail等、発信元の意味を規定
  - MAIL, DAEMONなどマクロ定義名で呼ばれることも
- 深刻さは0-7で決まっており小さいほど深刻
  - LOG_ERROR, LOG_INFOなどマクロでも呼ばれる
- ログはsyslogdと呼ばれるプログラムに集約され
  プレーンテキストで記録される
- 新提案としてsyslog-ng / rsyslogなど

R syslogが集めたメールのログ例

Oct 1 03:00:00 oniboshi postfix/pickup[79342]: 78C9D1B66E9D: uid=0 from=<root>
Oct 1 03:00:00 oniboshi postfix/cleanup[79505]: 78C9D1B66E9D: message-id=<20150930180000.78C9D1B66E9D@oniboshi.cysec.cs.ritsumei.ac.jp>
Oct 1 03:00:00 oniboshi postfix/qmgr[677]: 78C9D1B66E9D: from=<root@oniboshi.cysec.cs.ritsumei.ac.jp>, size=579, nrcpt=1 (queue active)
Oct 1 03:00:01 oniboshi postfix/local[79507]: 78C9D1B66E9D: to=<tetsu@oniboshi.cysec.cs.ritsumei.ac.jp>, orig_to=<root>, relay=local, delay=0.66, delays=0.05/0.02/0/0.59, dsn=2.0.0, status=sent (delivered to maildir)
Oct 1 03:00:01 oniboshi postfix/qmgr[677]: 78C9D1B66E9D: removed
Oct 1 03:01:13 oniboshi postfix/pickup[79342]: 6DA6D1B66E9F: uid=0 from=<root>
Oct 1 03:01:13 oniboshi postfix/cleanup[79505]: 6DA6D1B66E9F: message-id=<20150930180113.6DA6D1B66E9F@oniboshi.cysec.cs.ritsumei.ac.jp>
Oct 1 03:01:13 oniboshi postfix/qmgr[677]: 6DA6D1B66E9F: from=<root@oniboshi.cysec.cs.ritsumei.ac.jp>, size=33254, nrcpt=1 (queue active)
Oct 1 03:01:13 oniboshi postfix/local[79507]: 6DA6D1B66E9F: to=<tetsu@oniboshi.cysec.cs.ritsumei.ac.jp>, orig_to=<root>, relay=local, delay=0.04, delays=0.01/0/0/0.03, dsn=2.0.0, status=sent (delivered to maildir)
Oct 1 03:01:13 oniboshi postfix/qmgr[677]: 6DA6D1B66E9F: removed
Oct 1 03:01:33 oniboshi postfix/pickup[79342]: DDB0A1B66E9D: uid=0 from=<root>
Oct 1 03:01:33 oniboshi postfix/cleanup[79505]: DDB0A1B66E9D: message-id=<20150930180133.DDB0A1B66E9D@oniboshi.cysec.cs.ritsumei.ac.jp>
Oct 1 03:01:33 oniboshi postfix/qmgr[677]: DDB0A1B66E9D: from=<root@oniboshi.cysec.cs.ritsumei.ac.jp>, size=1816, nrcpt=1 (queue active)
Oct 1 03:01:34 oniboshi postfix/local[79507]: DDB0A1B66E9D: to=<tetsu@oniboshi.

通常のアプリケーションログ

- 各アプリケーションがsyslogを介さずに独自にログファイルを生成する例がある　形式も様々
- UNIXでは/var/log, /usr/local/var/log等のディレクトリに書き込む例が多い
- 例1: 代表的proxyのsquidのaccess.log
  1370768968.019　　75 127.0.0.1 TCP_MISS/200 551 GET http://www.google-analytics.com/__utm.gif? - DIRECT/74.125.235.233 image/gif
  - 最初の数値は時刻だが独自形式で理解できないetc.
- 例2: thttpdのアクセスログ
  186.225.xx.xx - - [28/Oct/2013:21:07:30 +0900] “HEAD /133.19.3.1/ HTTP/1.0” 404 0 “” “”
  121.116.yy.yy - - [28/Oct/2013:21:10:42 +0900] "GET /cysec.cs.ritsumei.ac.jp/mail HTTP/1.1" 404 0 "" "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.101 Safari/537.36“
  - httpサーバのログは似た形式に統一されている

その他Windowsで有用な情報

- アプリケーションによっては雑多な痕跡を¥Users¥ユーザ名¥AppData以下に残していく
- 中には利用状況に関し多くの情報が得られる場合がある（アプリケーション依存）
  - 例1: iTunesで同期を取ったデバイスのバックアップ C:¥Users¥ユーザ名¥AppData¥Roaming¥Apple Computer¥MobileSync¥Backup
  - 例2: Adobe Flashが残す永続的オブジェクトは C:¥Users¥ユーザ名¥AppData¥Roaming¥Macromedia¥Flash Player¥#SharedObjects

改めてフォレンジックとは何か？

- いわゆる実務的なフォレンジックとは
  「削除ファイルの復活を行って
  　それも含め中のファイルの証拠を集める」
- 専用のツールが沢山
  - 商用
    - EnCase (Guidance Software)
    - FTK (AccessData)
    - X-Ways Forensics (X-Ways　Software)
    - Oxygen Foreniscs (スマホ・ケータイ用)
  - フリーのもの
    - Autopsyなど
    - Volatirity Framework（ライブフォレンジック）

おそらく今後話題になるのは メディアフォレンジック

- Deep Fake