TCG技術を使った研究
クラウド環境におけるWindows
ゲストの完全性検証技術

情報通信研究機構
情報通信セキュリティ研究センター
安藤類央

はじめに

〇クラウドコンピューティングの普及
Amazon EC2, hybrid cloud

〇遠隔環境の安全性（integrity)の検証
　遠隔地にあるサーバは安全に起動・稼動しているのか。
　　遠隔地にあるWebサーバの検証：大事なパスワードを
　　預けていいのものか。
　　遠隔地にあるストレージサーバの検証：大事なファイル・
　　データを預けていいものか。

〇サーバ側の安全性確認：Remote attestationの重要性
　パスワードによる認証以前に、遠隔地にあるサーバが安全に
　起動されているか確認する。

〇クライアントOSの安全性確認：手元にあるあるいは預けたWindows OSの安全性
　（integrity）の検証。手元あるいは遠隔にあるWindows OSが感染（不正に改竄）して
いないか確認する。

関連研究プロジェクト

〇完全性検証
最近2004～2009年
　vTPM (IBM) Vitrual Trusted Platform Module
　TPM　チップを仮想化し、XENで利用可能にした。

　TVD (IBM) Trusted Virtual Domain
　アクセス制御やアテステーションを組み合わせて信頼できるゲストOSのドメインを構成する。

　IMA (IBM) Integrated Measurement Architecture
　ブートから、アプリケーションの起動までの安全性（完全性）を検証する。（トラストチェーンの構築）

最新2009年の研究
　仮想化技術の完全性検証への適用
　HIMA: North Carolina 大学
　　Annual Computer Security Applications　Conference 2009

〇観測・通知
Patagonix: Tronto University
　仮想マシン上でどのOSがどの実行ファイルを実行したか仮想マシンモニタ側で検証可能
XenAcess
　仮想マシンモニタXenで仮想している仮想OSの情報を得るためのAPI

関連研究プロジェクト
仮想化技術と完全性検証

○vTPM virtual Trusted Platform Module (IBM)

TPMチップを仮想化し、仮想化環境で完全性の検証ができるようにする。TPMを仮想化してゲストOSからアクセス出来るようにすることで、仮想化されたゲストOSにおいても、Trusted Computing技術を利用可能にする。現在、XenがvTPMサポートしている。

○VTD Virtual Trusted Domain (IBM TRL)

仮想化技術において、VMMによるアクセス制御機構と、ゲストOSへのアテステーションを元に、ゲストOSを信頼できるドメインの一員として管理する。

関連研究プロジェクト
仮想化技術と完全検証

〇HIMA (North Carolina State Univ and IBM)
Hypervisor（仮想マシンモニタ）を用いたゲストOSの完全性検証。IBM Watson研のLinux IMAへのhypervisorへの拡張。仮想マシンモニタ技術を用いて、ゲストOSのisolationと、consistency(TOCTTOU)の双方を確保する。

〇VMKnoppix (産総研、AIST）
Knoppix: On memory(ハードディスクなしで起動する）LINUX。Knoppix上でVMが起動し、VTPM（仮想化されたTPM）を用いて起動などを安全に行うことが可能。

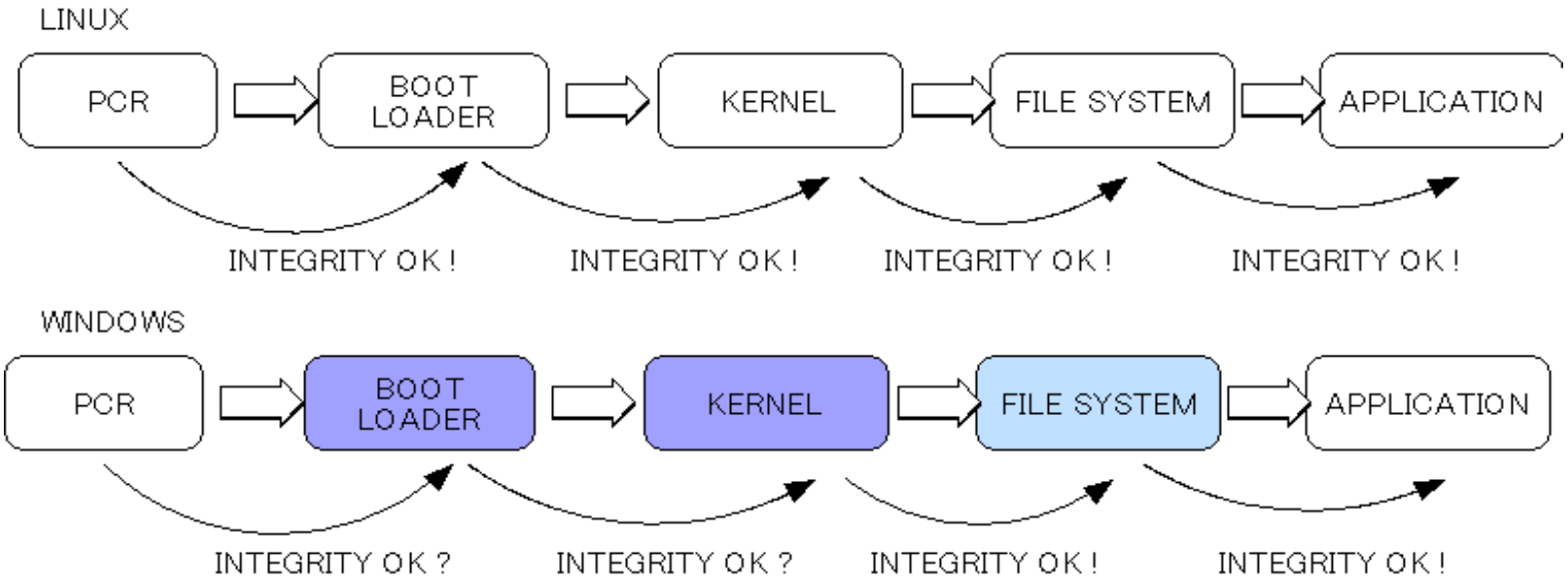
現状のTCG技術の問題点
Windows上のintegrity検証

- Trust Chainの問題

Trust Chainとは：起動の初めからOSやアプリケーションが安全に起動、稼動しているか検証するための仕組み。

改竄されないハードウェアの値（PCR）→ブートローダが安全に起動したか→カーネルがセキュアにロードされたか→ファイルシステムが安全にロードされたか→アプリケーションが安全に起動されたか。

Linuxと異なり、クローズドソースのWindowsは、セキュアブートからのTrust Chainが途切れてしまう。計測ソフトウェアの安全性（integrity)の保障ができない。



Windows上のintegrity検証
なぜWindowsではtrust chainが観測計測できないのか

〇Linux OS上での完全性計測
- 多くの機関で研究されている。LinuxではIBM発のIntegrity Measurement Architectureで
- Trust Chainが構築できる。

〇Windows OS上での完全性計測
多くの人が使用している割に、あまり研究されていない。クローズドソース
のため、情報が入手しずらく、処理内容が分からない。

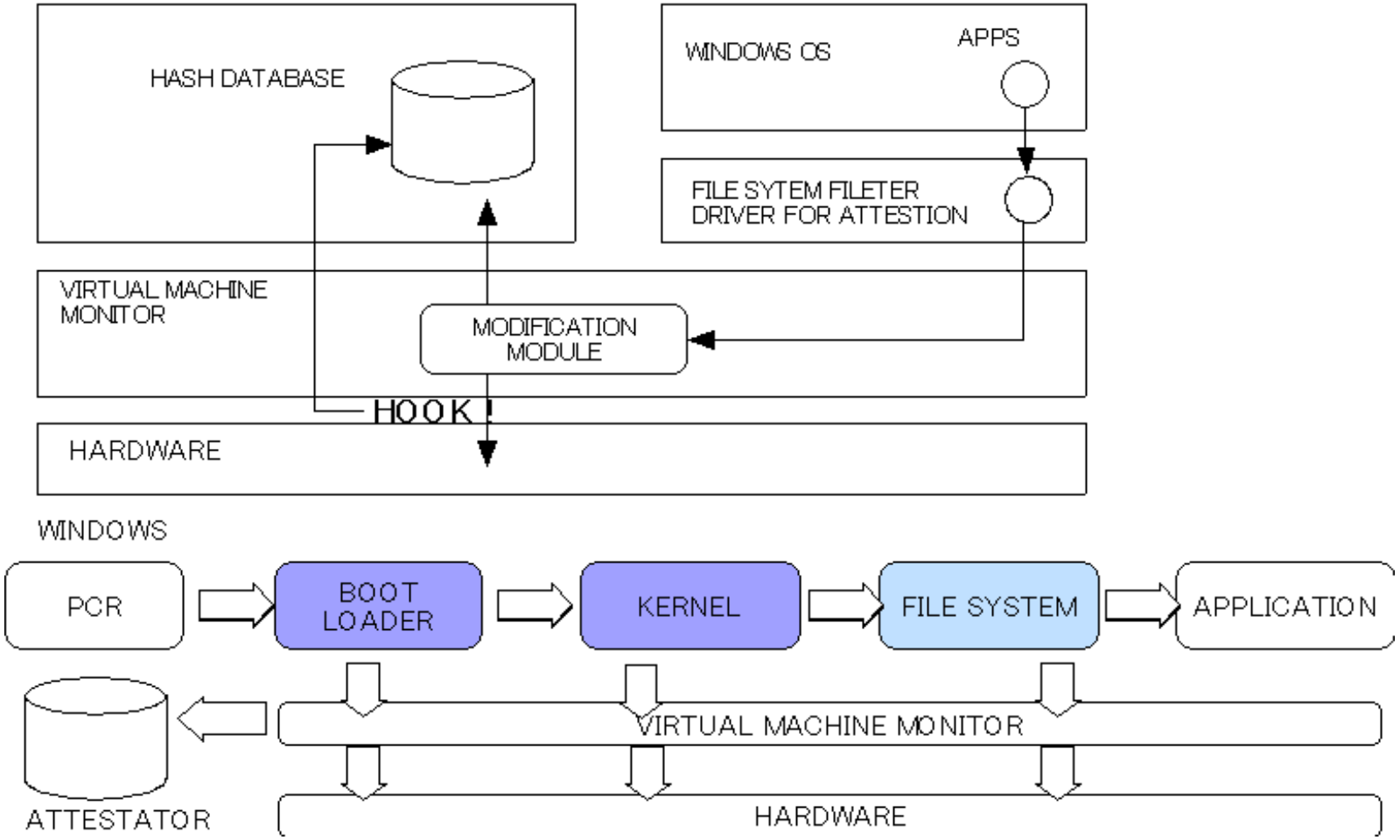
〇対策
①Windows ファイルシステム（のフィルタリング）ドライバを用意することで、
ファイルの実行時などにハッシュ値をとり、ある程度の完全性を計測する
ことができる。

②仮想マシンモニタ（外側）からWindows OSのブート時の挙動を観測し、
Trust chainを構築する。

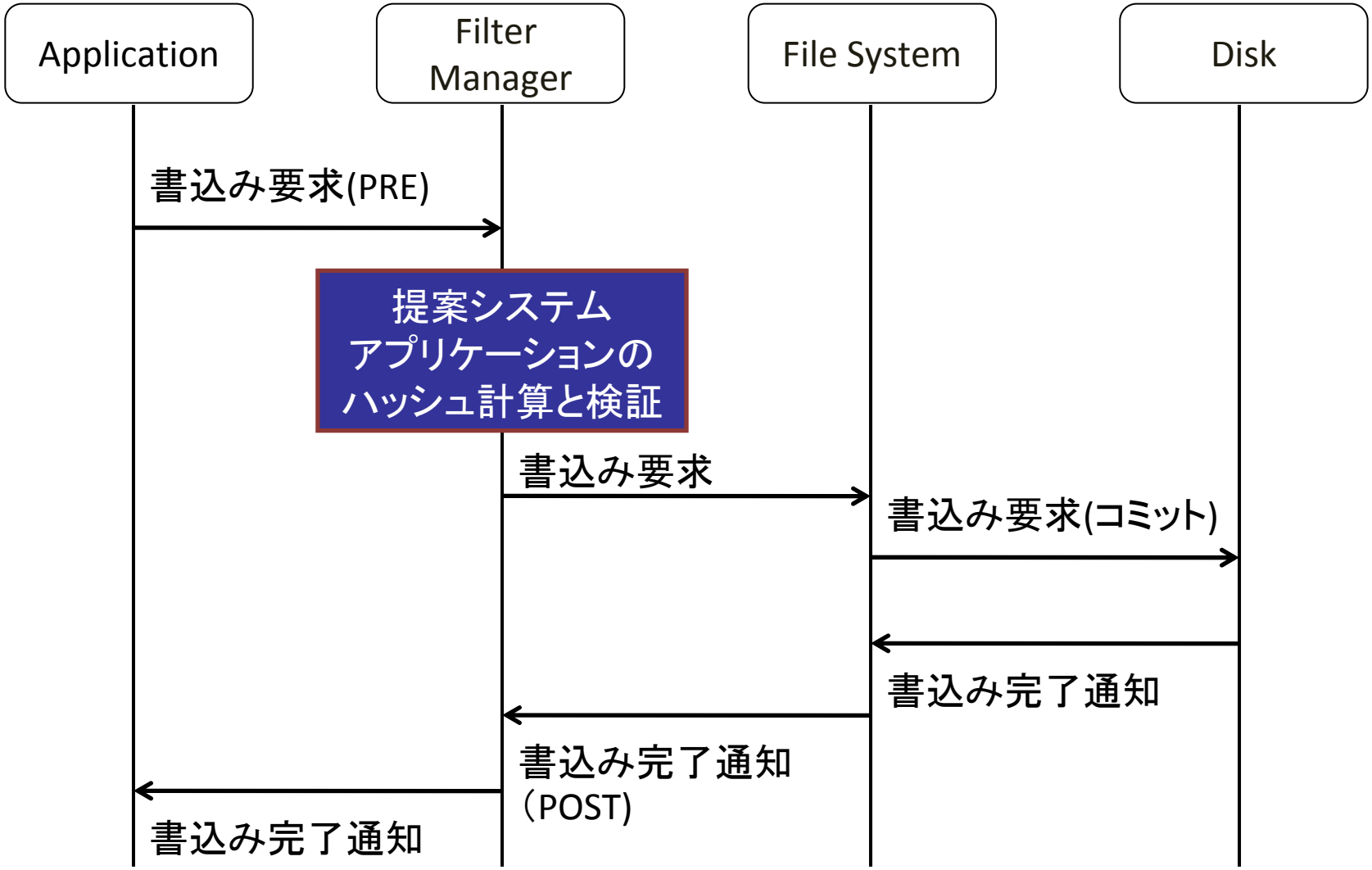
①と②を組み合わせて、Trust Chainを構築する。

今後の課題課題：Windows OS上Trust Chainの構築

- 仮想化技術を用いたwindows上のTrust Chainの構築（Windows OSを修整し、仮想マシンモニタからWindows OSの起動、稼動を観測する。）



提案システム


Application
Filter Manager
File System
Disk
書込み要求(PRE)
提案システム
アプリケーションの
ハッシュ計算と検証
書込み要求
書込み要求(コミット)
書込み完了通知
書込み完了通知
(POST)
書込み完了通知

適用技術

- Windows OS上でアクセス制御を行う手段
  - ユーザーモード
    - DLLインジェクション
    - APIフック
  - カーネルモード
    - フィルタドライバ

・ ファイルアクセスを
すべて監視できる
・ OSと一番近い層で動く
・ アクセス制御に必要な情報が
取得できる

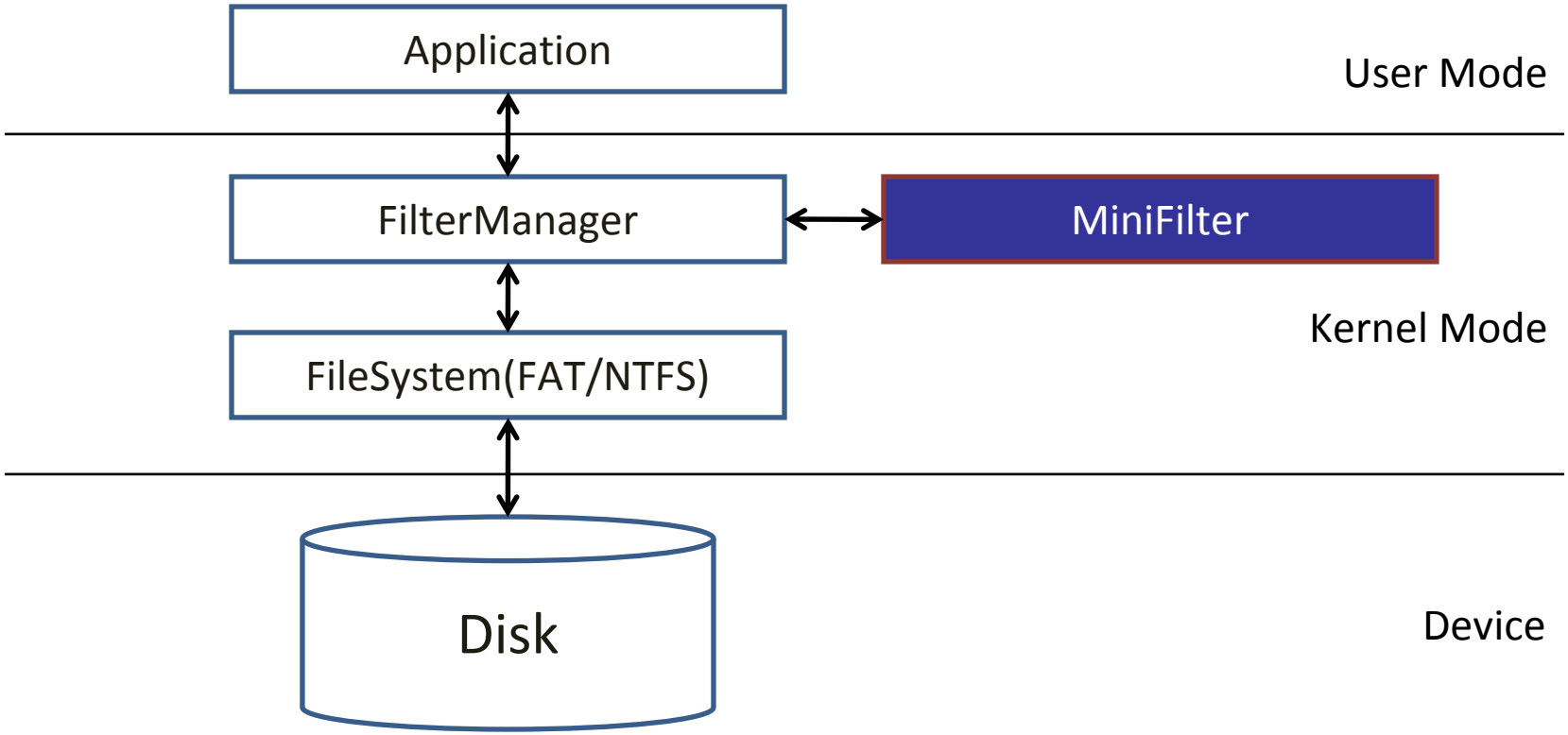
適用技術

FilterManager

- Microsoftが提供するファイルシステムフィルタドライバ
- Windows XP SP2以降
- サードパーティの開発を簡素化
  - フィルタするIRPのみ編集
  - 開発時間の短縮
  - 挿入する階層を指定可能

適用技術

FilterManagerの構成



適用技術

FilterDriver

- FilterManagerとMiniFilter
- フィルタするIRPをFilterManagerに登録
- フィルタするIRPの処理を実装

IRP：ファイルIO要求の受け渡し

フィルタマネージャの詳細

- MiniFilterの登録

```
typedef struct _FLT_REGISTRATION {
    USHORT Size;
    USHORT Version;
    FLT_REGISTRATION_FLAGS Flags;
    CONST FLT_CONTEXT_REGISTRATION *ContextRegistration;
    CONST FLT_OPERATION_REGISTRATION *OperationRegistration;
    PFLT_FILTER_UNLOAD_CALLBACK FilterUnloadCallback;
    PFLT_INSTANCE_SETUP_CALLBACK InstanceSetupCallback;
    PFLT_INSTANCE_QUERY_TEARDOWN_CALLBACK
InstanceQueryTeardownCallback;
    PFLT_INSTANCE_TEARDOWN_CALLBACK InstanceTeardownStartCallback;
    PFLT_INSTANCE_TEARDOWN_CALLBACK InstanceTeardownCompleteCallback;
    PFLT_GENERATE_FILE_NAME GenerateFileNameCallback;
    PFLT_NORMALIZE_NAME_COMPONENT NormalizeNameComponentCallback;
    PFLT_NORMALIZE_CONTEXT_CLEANUP NormalizeContextCleanupCallback;
} FLT_REGISTRATION, *PFLT_REGISTRATION;
```

フィルタマネージャの詳細

フィルタするIRPの設定(1)

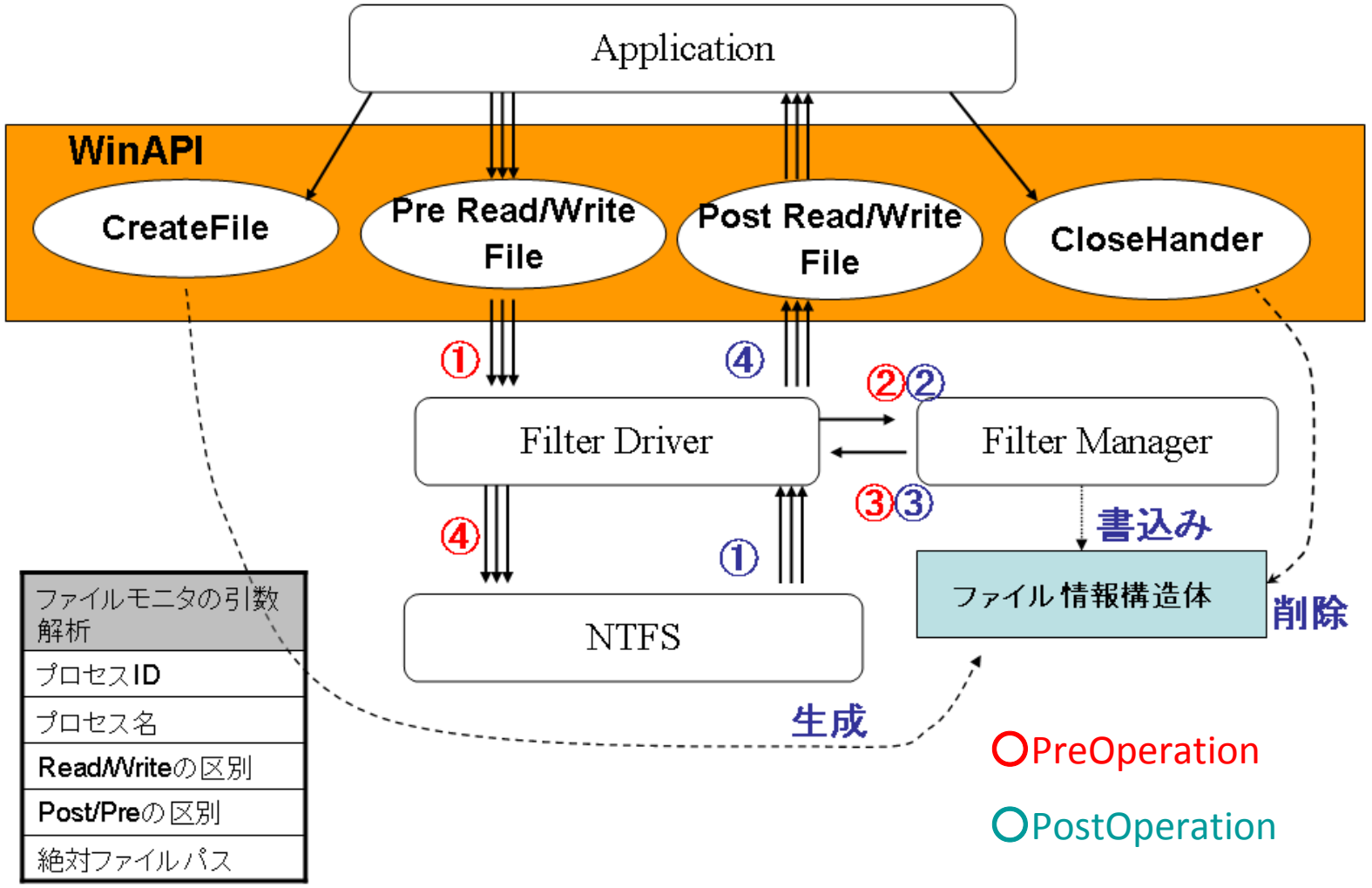
```
typedef struct _FLT_OPERATION_REGISTRATION {
    UCHAR MajorFunction;
    FLT_OPERATION_REGISTRATION_FLAGS Flags;
    PFLT_PRE_OPERATION_CALLBACK PreOperation;
    PFLT_POST_OPERATION_CALLBACK PostOperation;
    PVOID Reserved1;
} FLT_OPERATION_REGISTRATION,
*PFLT_OPERATION_REGISTRATION;
```

ファイルシステム呼び出し前のコールバック関数

ファイルシステム呼び出し後のコールバック関数

フィルタマネージャの詳細

アクセス制御の仕組み



フィルタマネージャの詳細

ファイル情報構造体

```
typedef struct _FILE_ACCESS_DATA{
    BOOLEAN                 IsDirectory;
    UNICODE_STRING  FullPath;
    ULONG           AccessRecord;
    ULONG           AccessAttributes;
    ULONG           ProcessID;
    PWCHAR                  ProcessName;
    ULONG           ProcessAttributes;
} FILE_ACCESS_DATA, *PFILE_ACCESS_DATA;
```

ファイルアクセスを行うパス名

ファイルアクセスの要求プロセス名

プロセスのアクセス属性

フィルタマネージャの詳細

フィルタするIRPの設定(2)

No	IRP要求	PRE	POST
1	IRP_MJ_CREATE	○	○
2	IRP_MJ_CLOSE	○	×
3	IRP_MJ_CLEANUP	○	×
4	IRP_MJ_READ	○	×
5	IRP_MJ_WRITE	○	×
6	IRP_MJ_SET_INFORMATION	○	×

フィルタマネージャの詳細

関数の型		FLT_PREOP_CALLBACK_STATUS
機能		・ファイル情報構造体の取得 ・アクセス可否の判断 ・アクセス可否に応じた戻り値の設定
引数	1	PFLT_CALLBACK_DATA Data
	2	PCFLT_RELATED_OBJECTS FltObjects
	3	PVOID *CompletionContext
戻り値	1	FLT_PREOP_SUCCESS_WITH_CALLBACK (成功)
	2	FLT_PREOP_SUCCESS_NO_CALLBACK (成功)
	3	FLT_PREOP_COMPLETE (成功/失敗)
失敗時の設定		戻り値をFLT_PREOP_COMPLETEに設定 引数DataのIoStatus.Statusにエラー値を設定

今後のソリューション
仮想化によるTrust Chainの構築

①フィルタドライバによる観測の問題点

フィルタドライバのロード以前、ブート時の完全性の検証ができない。

②仮想マシンモニタによる観測の問題点

ロード時の観測、完全性を検証できる可能性があるが、ブート後のアプリーションの完全性の検証が難しい。（semantic gap: 仮想マシンモニタからはゲストOSでなにか起こっているか知るのは難しい。

ソリューションの適用例：TPMを用いたWEBサーバ IIS（または Windows apache)の 安全性検証

遠隔地にあるWEBサーバが安全に起動、稼動されているか。

①前述のスライド①②によるフィルタドライバによるWindows OS修正と、仮想マシンモニタによる外部観測

②Web server (IIS)を稼動するWindows server OSの完全性の検証

③フィルタドライバを用いたIISのexe, sys, ファイルアクセスの完全性検証

ご清聴ありがとうございました