BitVisorのためのマルウェアの検出機能

2012年12月4日

電気通信大学

河﨑 雄大　大山 恵弘

背景

- 近年多くのマルウェアが発見されている

- マルウェア検出にアンチウイルスソフトが使われる

- 既存のアンチウイルスソフトではいくつかの問題点が存在する

既存のアンチウイルスソフトの問題点

1. OSの管理者権限を持ったユーザによる無効化
   - 低スペックなPCでアンチウイルスソフトが負担になっている場合
   - ユーザの意図しない処理で消してしまう場合
   - 金銭的な理由の場合
2. マルウェアによる無効化
   - OSがマルウェアに乗っ取られた場合

アンチウイルスを強制させたい場合にシステム管理者の意に反している

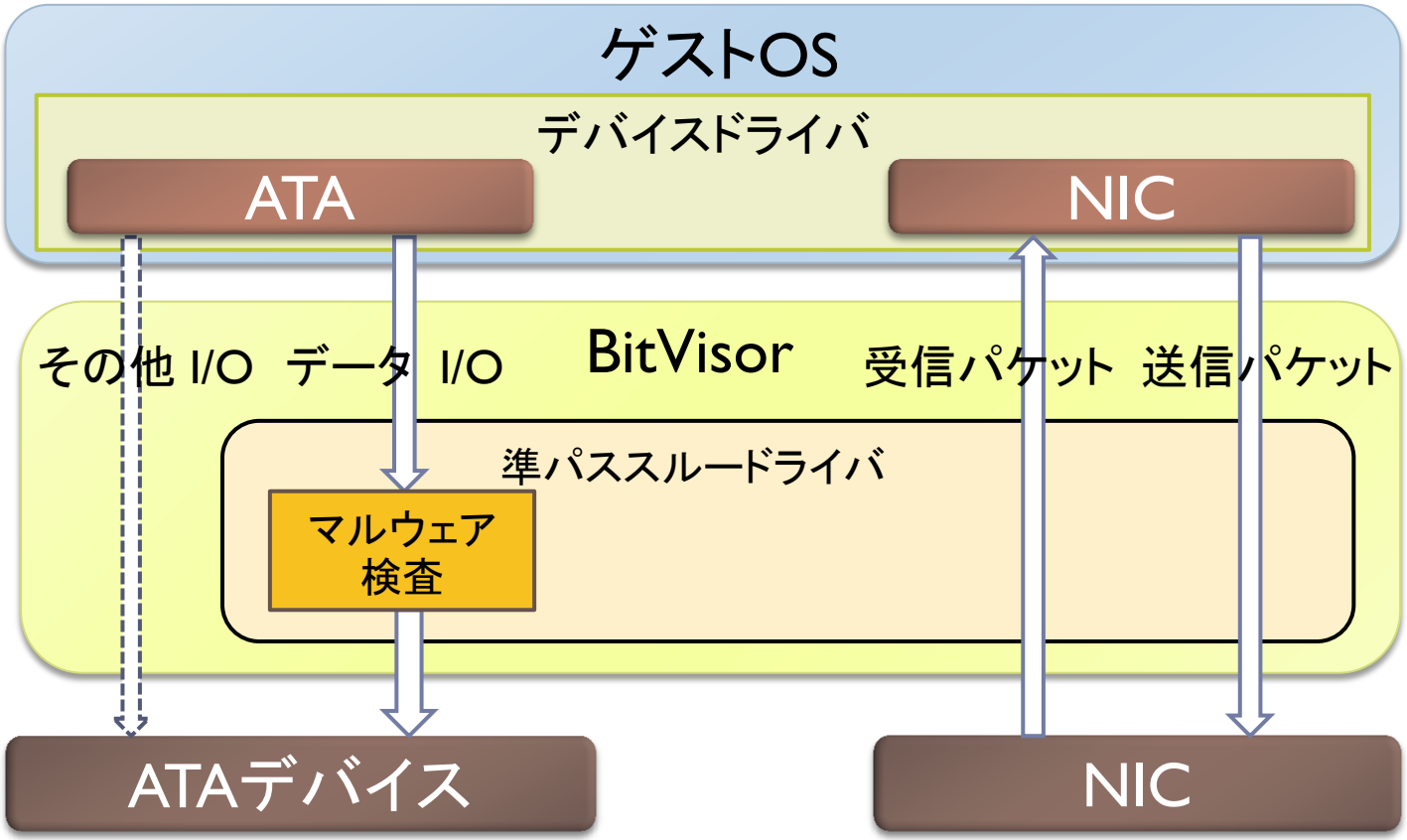
OSの外でセキュリティの処理を行いたい！

目的と方針

- 目的
  - BitVisorに対してマルウェアの検出機能の提案
- 方針
  - OSがデバイスに書き込むデータに対して検査を行う
    - 対象：ATA
  - 検出方法はシグネチャマッチング
    - シグネチャのパターンデータはBitVisorのバイナリに埋め込む

マルウェア検出機能の概要（1）

- 検査するデータ：OSとデバイスの間を流れるバイト列
  - PIOとDMAに対応



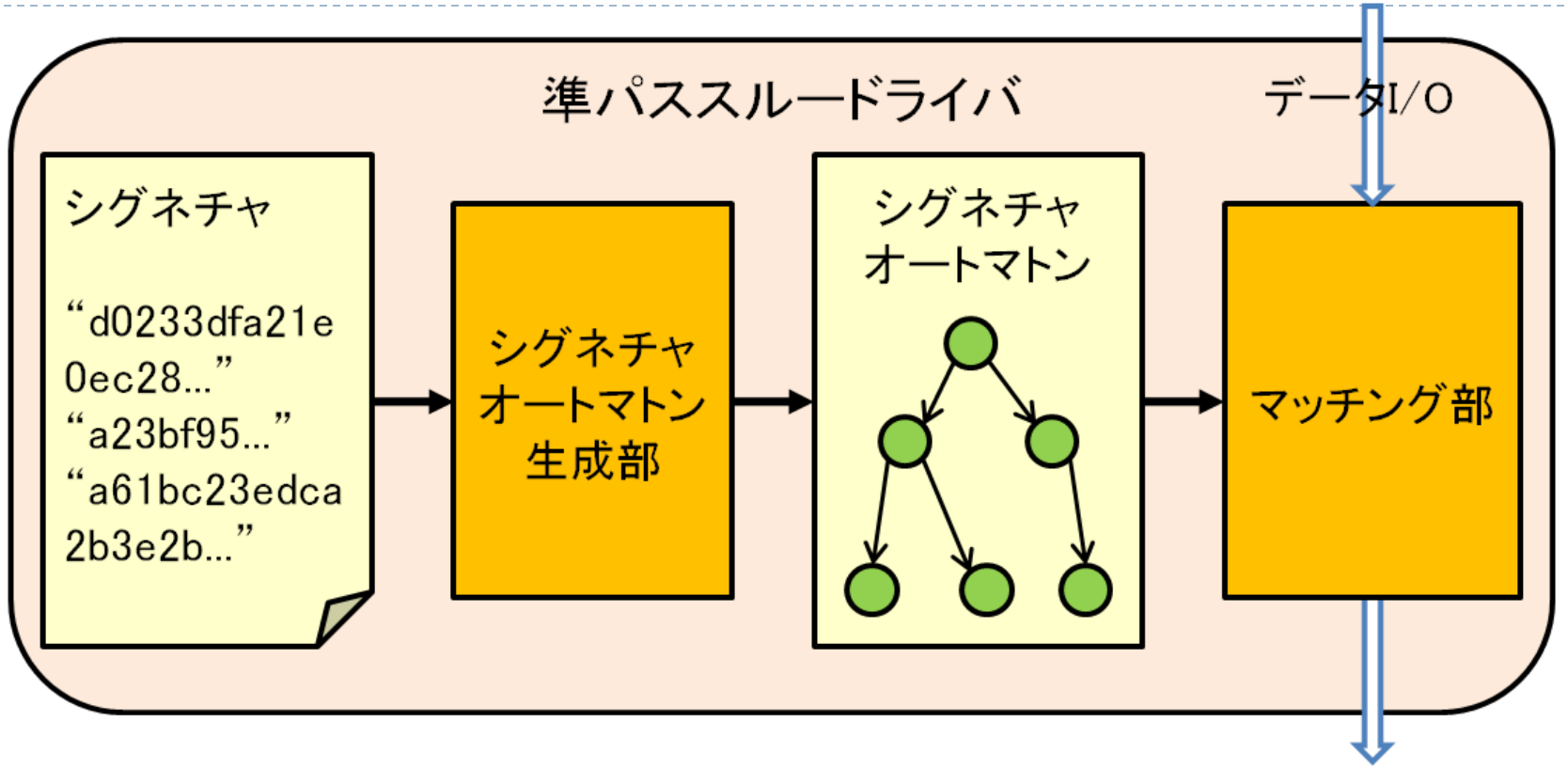
マルウェア検出機能の概要

- マッチングアルゴリズム：Aho-Corasick法
- 使用するシグネチャ：ClamAVのシグネチャ

マルウェアの
名前
＋
シグネチャ
（16進数表記）

```
Aircop=32e4cd16cd1233c0cd130e07b80002b9
Burghofer=8e06120033ff8bf30e1fb90d02f3a4
Burglar=03042e8ba4050433c033db2effac09045081c70304
b90a00b87677902e31054790e2f958c3
Carnage=e8050502002ea3b5022ec706b3029f00b440b99f02
ba1000cd21b8004233c999cd21b440b91a00
:
```

マルウェア検査方法

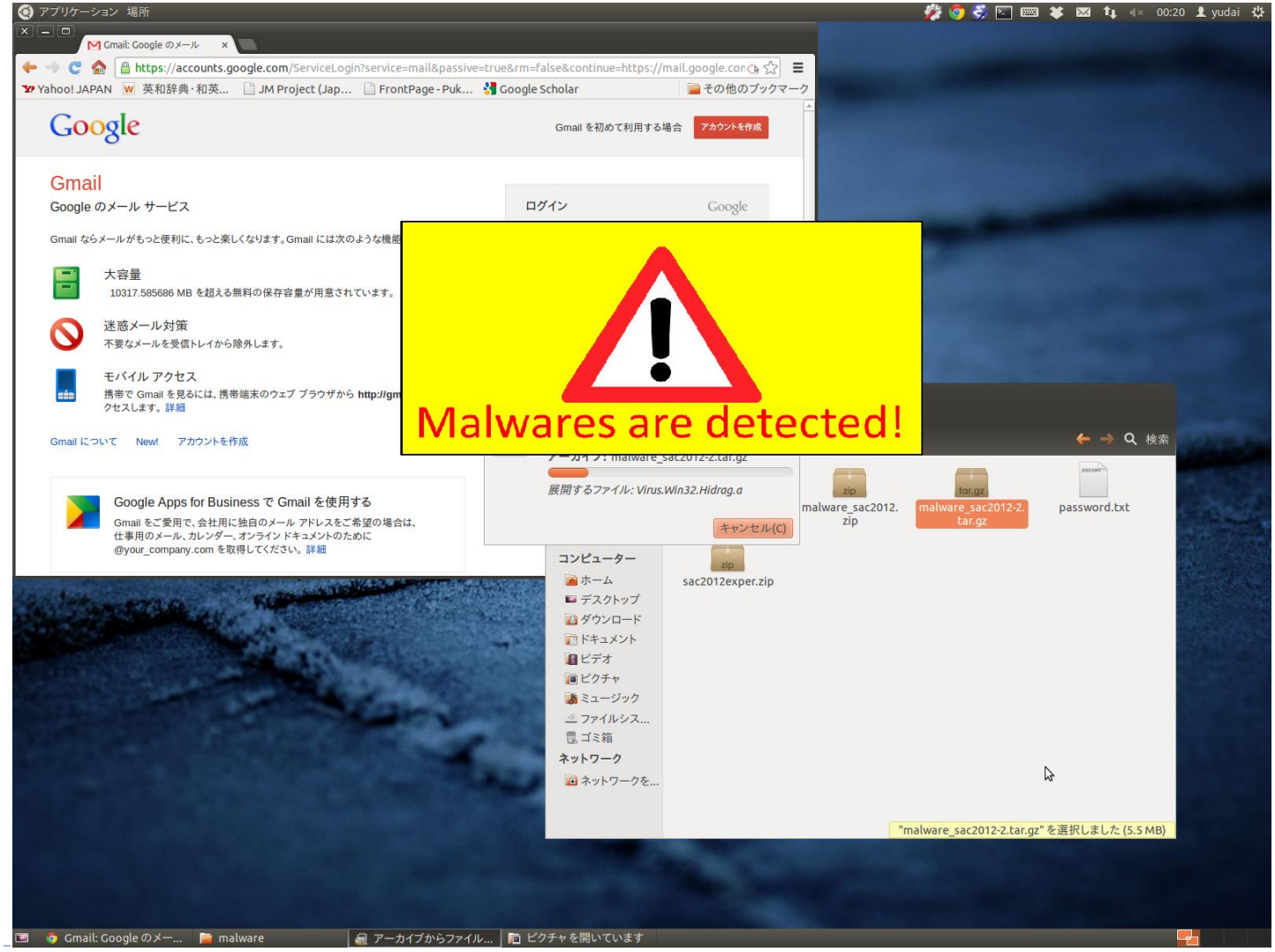


- 起動時にシグネチャオートマトンを作成
- シグネチャオートマトンを基にマルウェア検査を実行

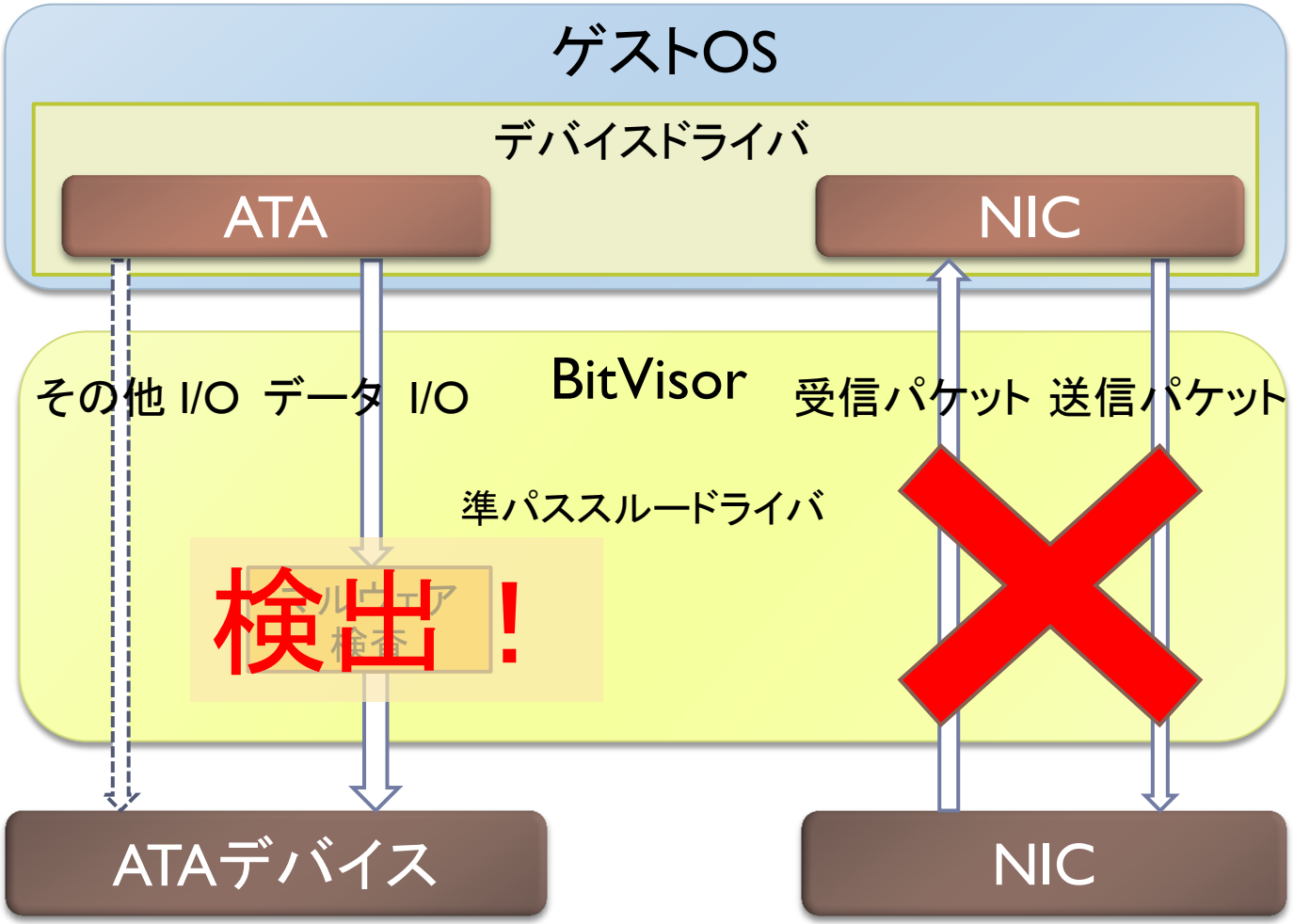
マルウェアが検出された場合

- 警告をデスクトップ上に表示する
- ネットワークを遮断する
- マッチした部分をゼロフィルする（未実装）

対処1: 警告画面



対処2:ネットワークの遮断



対処3: ゼロフィル（未実装）

OSとデバイスの間を流れるバイト列

6f34e1b876c4ac38b64138f64831b683c26a
486d231486c32c8d463198b64832a618463
1284682a3164a871236486c32864b3286e4
876213846000000000000000000006c34
86e3b864912e38a648d32648d6c32f986023
a15a09730b573b128736856b38b9640f83d2
e6a4812c63c486138a64128364183648314
08a6128036

本システムのシグネチャデータの問題

- シグネチャデータを置く場所がないので本システムではバイナリに埋め込んで使用している
- 日々新種のマルウェアが発見されている
- 新たなシグネチャデータが不可欠になってくる
- **動的なシグネチャ更新機能が必要！**

動的なシグネチャ更新機能

- ゲストOSを用いた動的更新機能
  - 補助的な更新方法
- ネットワーク越しの動的更新機能
  - メインの更新方法

ゲストOSを用いた動的更新機能

- ゲストOSに更新データのファイルを置き、更新プログラムで更新を行う

- Intel VTで用意されているVMCALL命令を用いる

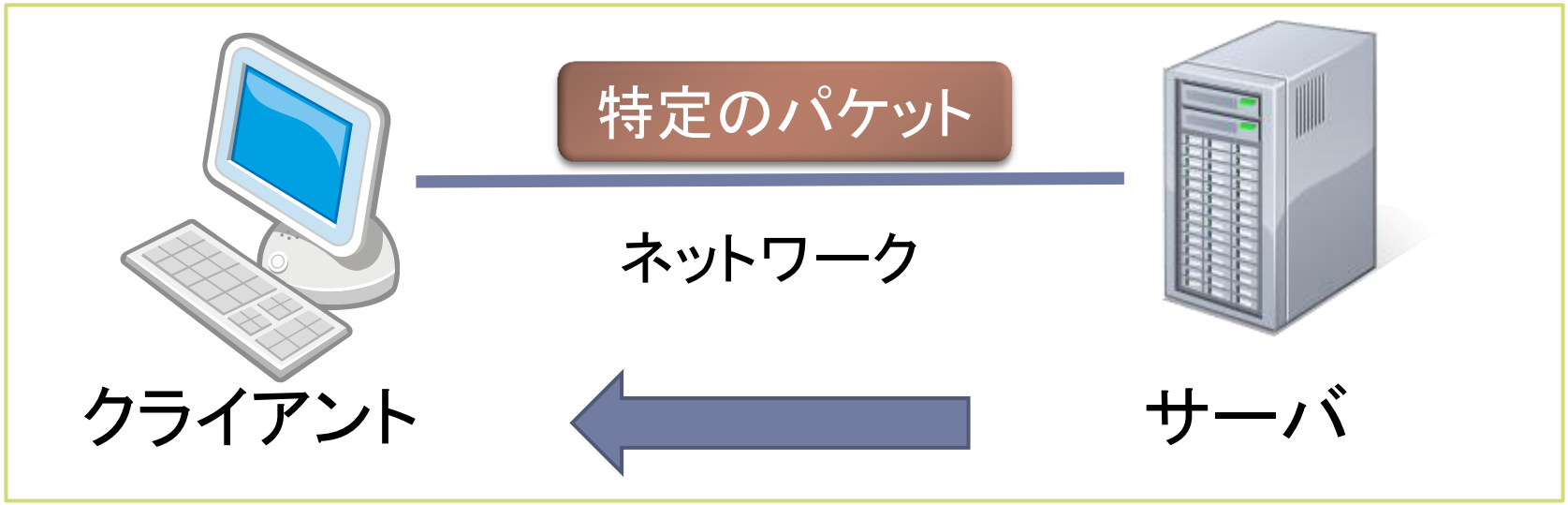
```
static int vmcall(unsigned long a, unsigned long b,
                  unsigned long c, unsigned long d) {
  int ret;
  __asm__ __volatile__
    (".byte 0xf,0x1,0xc1"
     : "=a" (ret) : "a" (a), "b" (b), "c" (c), "d" (d),
    );
    return ret;
}
```

更新プログラムの詳細

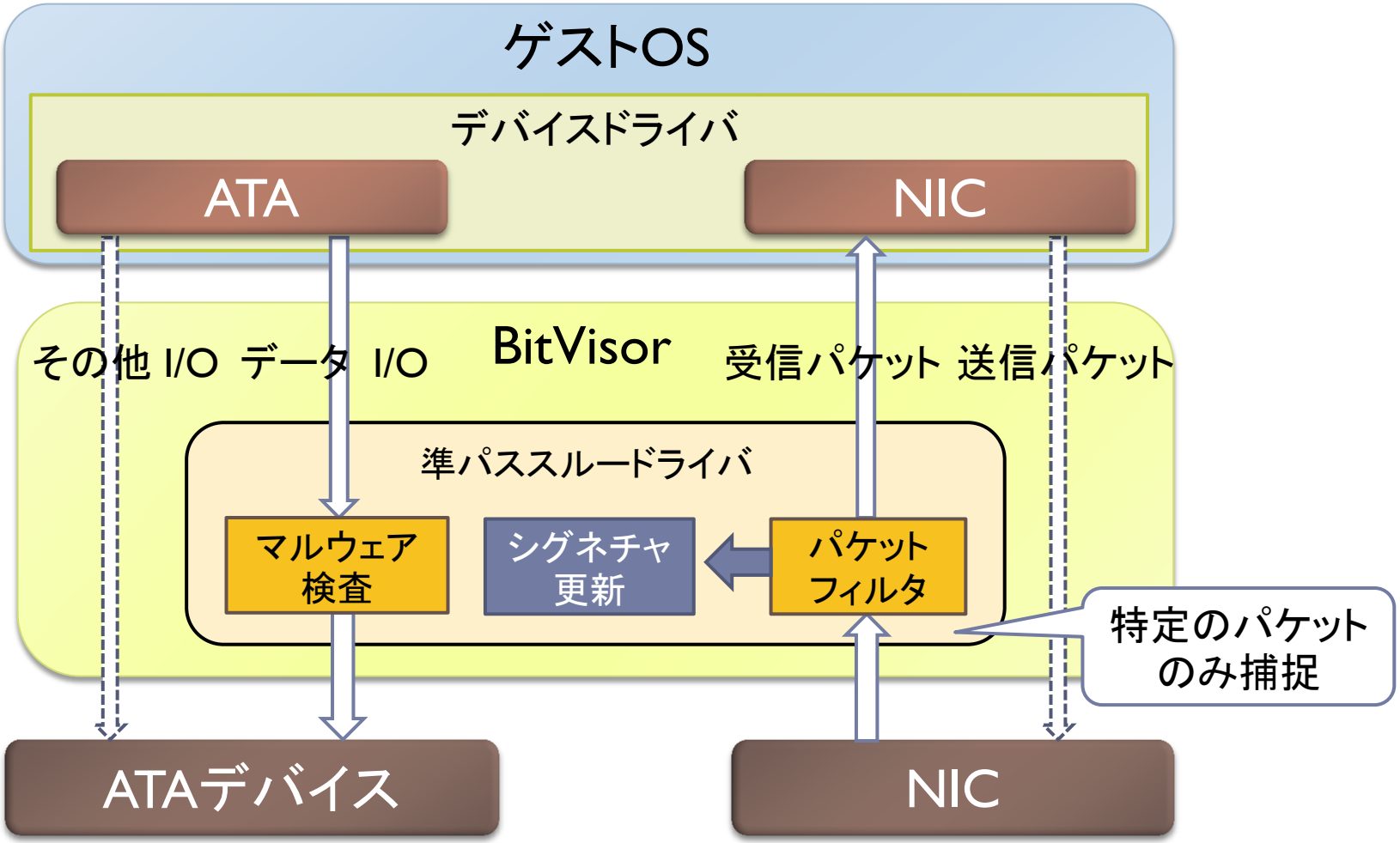
- ゲストOSのメモリに更新データを書き込む
- そのアドレスを汎用レジスタに入れVMCALL命令を呼び出す
  - RAX: 2 ←本機能の識別番号
  - RBX: 更新データが置かれたメモリの先頭アドレス
  - RCX: 更新データの長さ
  - RDX: 電子署名が置かれたメモリの先頭アドレス
- BitVisorで電子署名を基にシグネチャの整合性を確認した後、更新を行う

ネットワーク越しの動的更新機能

- ネットワーク越しに更新を行う
- 特定のパケットでシグネチャ更新を行う

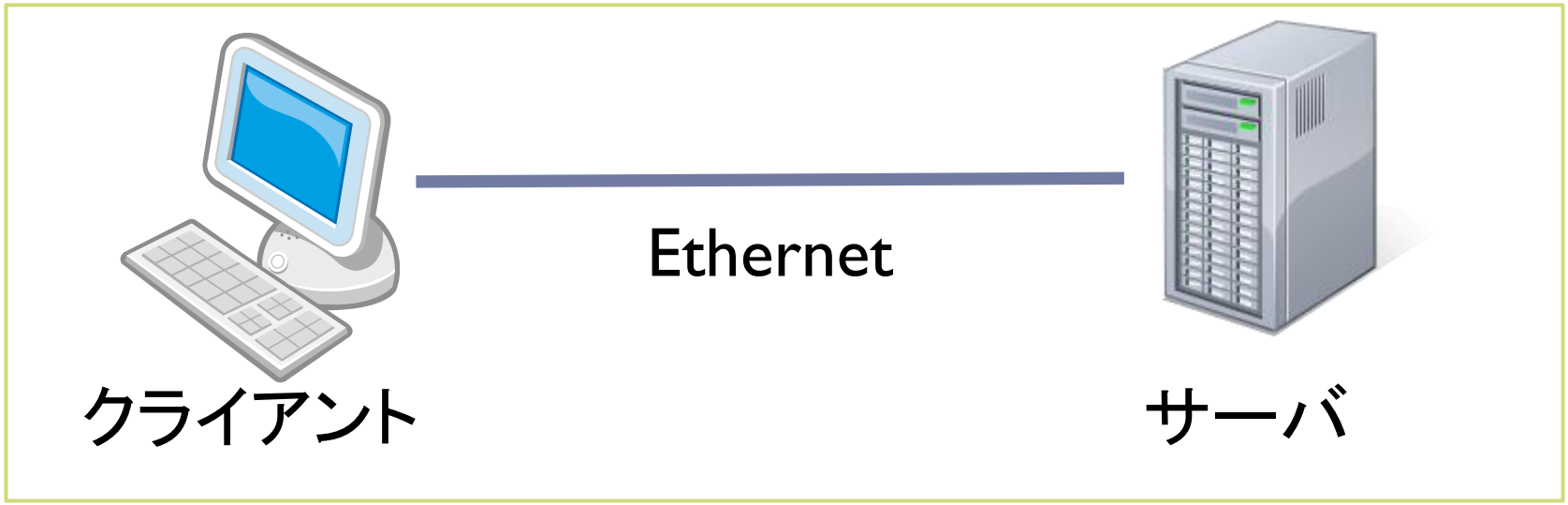


ネットワーク越しの動的更新機能



ネットワーク越しの動的更新機能の詳細

- 想定環境
  - Layer2: Ethernet
  - Layer3: IPv4
  - Layer4: UDP (ポート番号: 専用なもの)

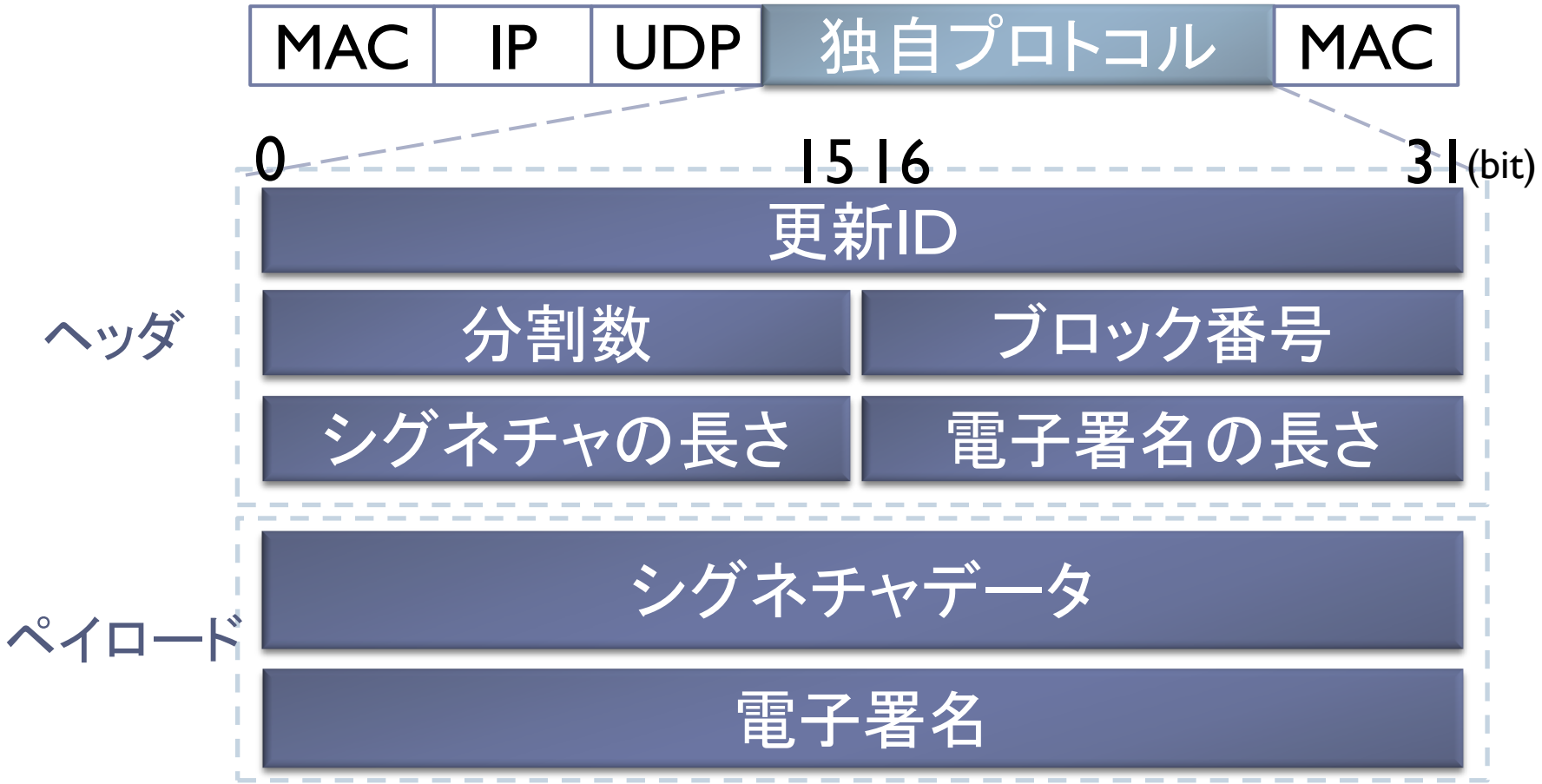


なぜUDPか？

- コネクションレスだから
  - TCPだと3ウェイハンドシェイクが必要
- 複数の相手に同時に送信可だから
- UDPだと実装が楽だから

更新に用いる特定パケット

- UDP/IPベースのパケット



独自プロトコルのペイロード部

- シグネチャデータ
  - マルウェアの名前+マルウェアのシグネチャ（16進数表記）
- 電子署名
  - シグネチャデータのハッシュを秘密鍵で暗号化したもの

シグネチャデータ

new_virus1=0f3be373e986a78cddbd0f150f4764a273d78a70dd09bcdd0296cfb3102635562ab5c978e0a4ad9d2811830f60844285ab4fcb8b9e9e9b07e
new_virus2=373e986a78cddbd0f150f4764a273d78a70dd09bcdd0296cfb3102635562ab5c

電子署名

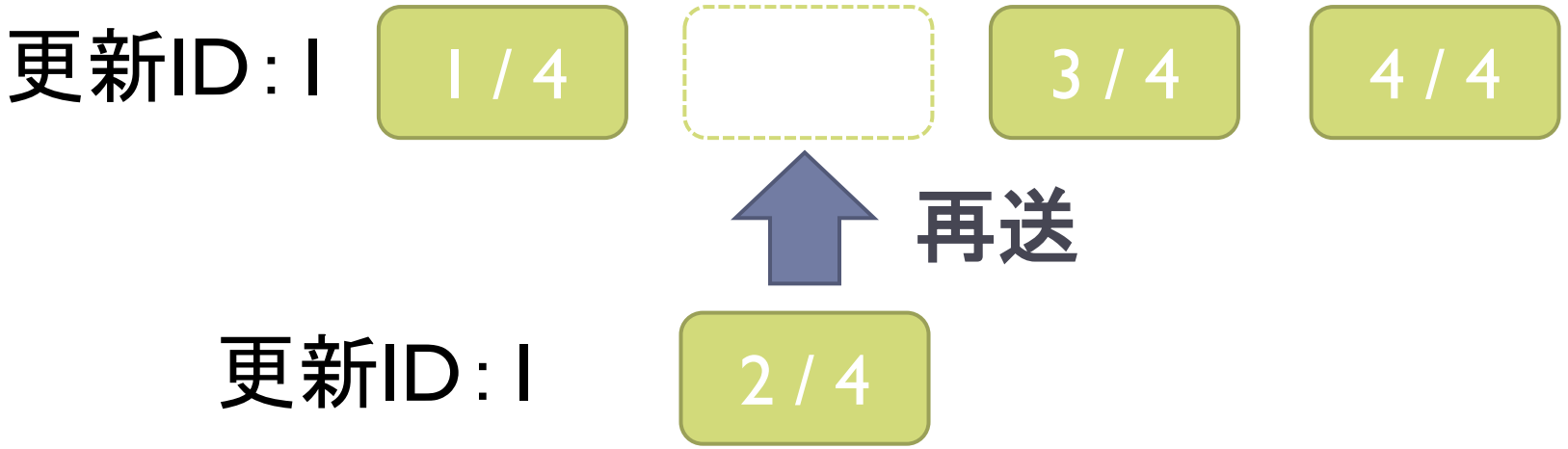
40d6ce3e7d50953bb11234c3a73618d45eac77747c146cf550e55ee62dc22bf8f899eef8790b92bd33fee00c330fdebdaf064f21668586d2b60a92c8b9d9773e

ネットワーク越しの更新機能の問題

- パケット落ち
- パケットの改ざん
- パケットの大量送り付け

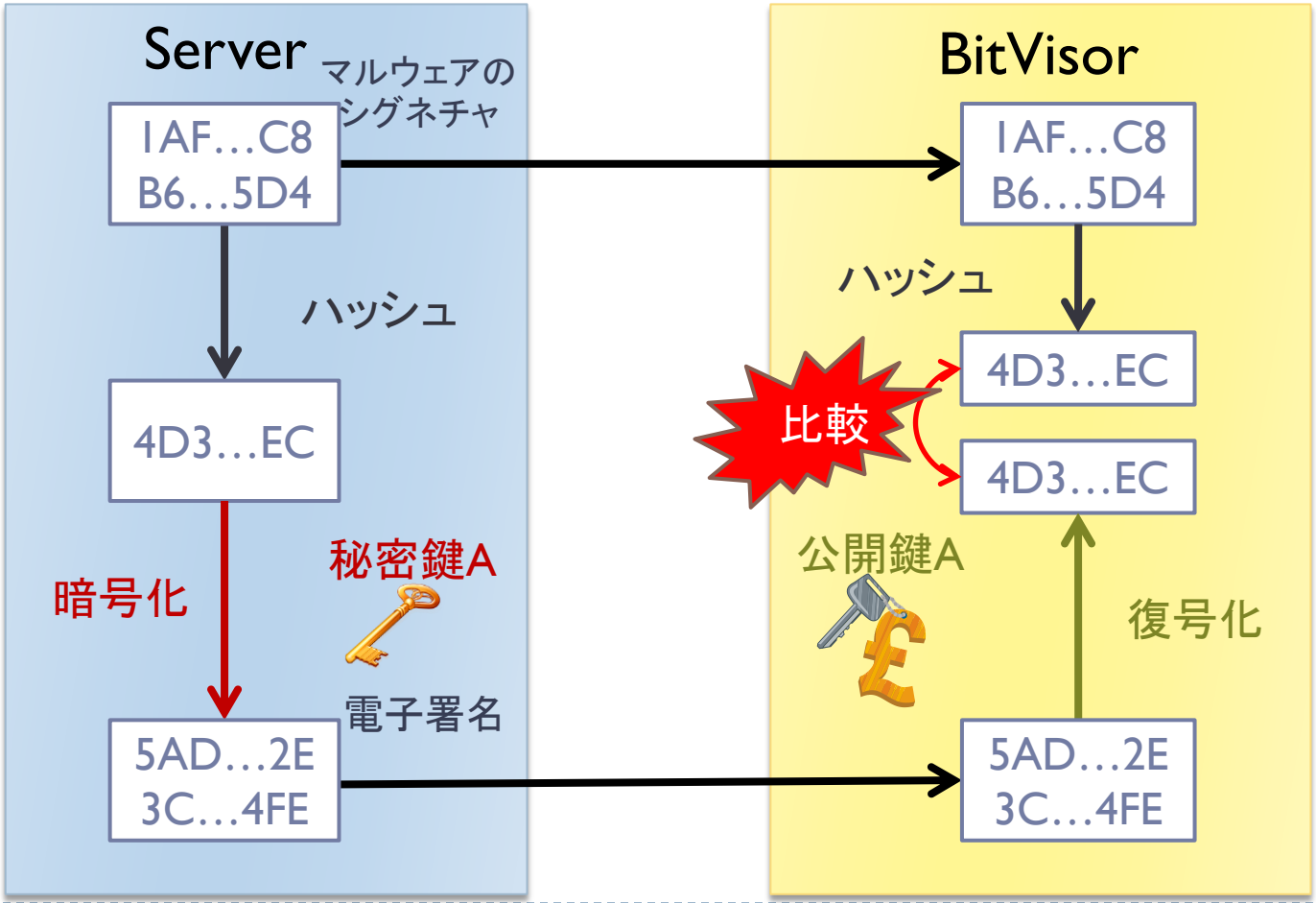
パケット落ちへの対処

- 更新IDと分割数、パケットの位置で対処



パケットの改ざんへの対処

- 電子署名でシグネチャデータの整合性を確認



パケットの大量送り付けへの対処

- 攻撃メカニズム
  - 外部から更新用のパケットに見せかけたパケットを大量に送り付けることによってシステムをダウンさせる攻撃
- 対処方法
  - 現在考案中
  - DoS攻撃の対処方法を用いる

評価

- 実験環境
  - CPU: Intel Pentium D 3.0 GHz
  - メモリ: 2 GB
  - VMM: BitVisor 1.3
  - ゲストOS: Ubuntu 12.04 32-bit
    Linux 3.2.0-34-generic-pae

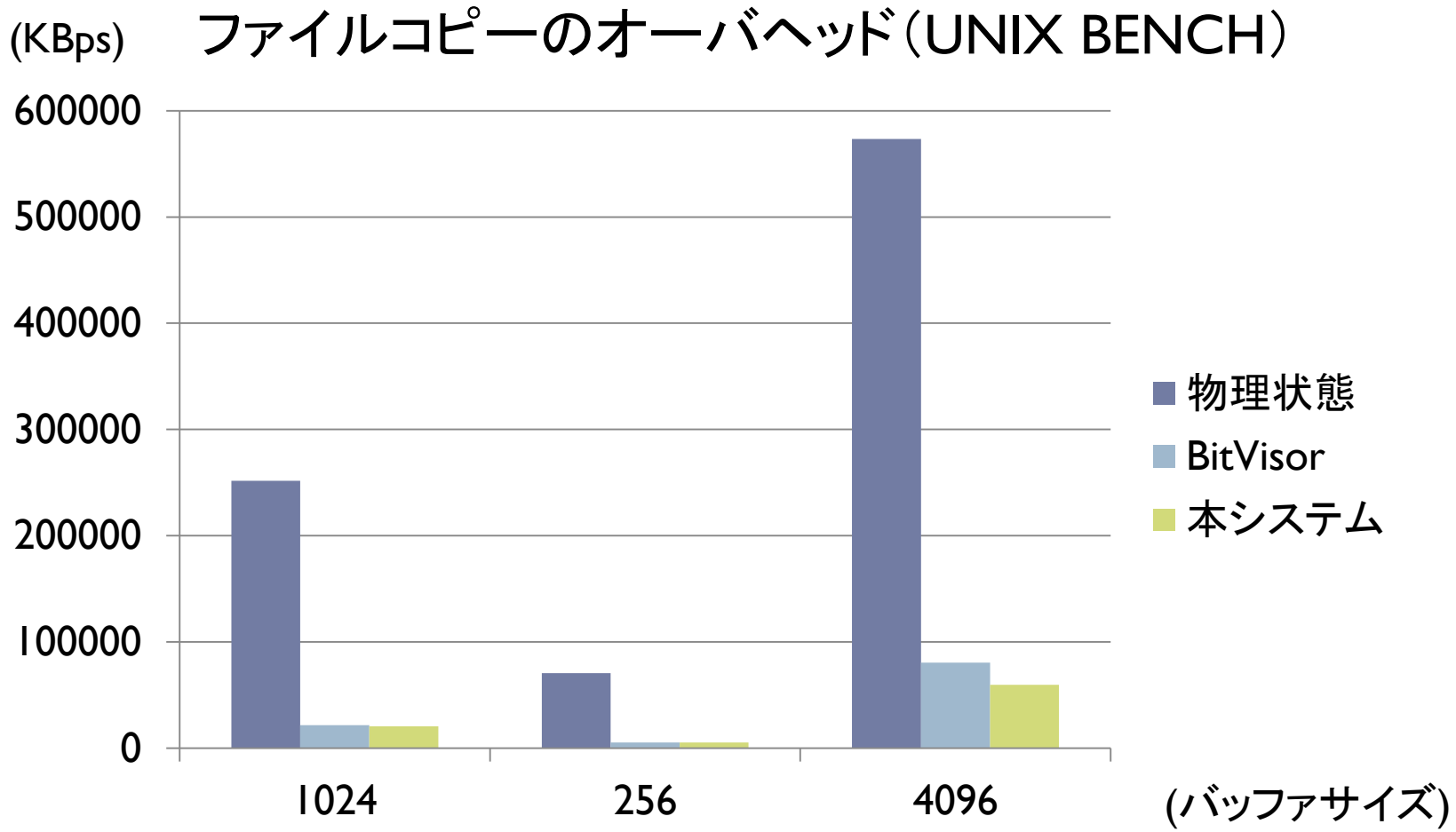
マルウェア検出実験の結果

- 実験対象:
  - 実際のマルウェア226個

	本システム	ClamAV
検出	206個	226個
未検出	20個	0個

実際のマルウェアに対しても有効である

本システムを導入した際のオーバヘッド



BitVisorに比べて20%のオーバヘッドが生じた

関連研究

- VMwatcher [Jiangら 2007]
  - VM上で動くゲストOSのハードディスクやメモリ上のデータに対してマルウェア検出を行うシステム
  - システムはホストOS上で動作
  - 本システムとは動作する場所が異なる

- Clam AV
  - 既存のアンチウイルスソフト
  - シグネチャデータを更新する際、公開鍵暗号方式による電子署名を使用
  - 本システムではVMMによるマルウェア検出システムへ導入

まとめと今後の課題

- まとめ
  - BitVisorに対してマルウェアの検出機能を導入した
  - シグネチャの動的更新機能も提案した

- 今後の課題
  - NIC，USBなどのデバイスへの対応
  - ゼロフィル機能などの未実装の機能の実装
  - パケットの大量送り付け攻撃などへの対処