Lec 16: Anti-Malware

CS492E: Introduction to Software Security

Sang Kil Cha

Anti-Malware or Anti-Virus (AV)

We will interchangeably use the terms.

Terminology

- Virus
- Worm
- Trojan
- Rootkit
- Spyware
- Bots
- Backdoor
- Adware
- Ransomware
- Etc.

AV (Anti-Virus)

Cohen’s Question

Given an arbitrary program, can we design a Turing machine that determines whether the program is malicious or not?

No, this is an undecidable problem!

Informal Proof

Define a function ***isVirus*** that takes a program as input, and outputs true if the program is a virus or false otherwise. Let's assume that this function exists:

```
def isVirus(prog):
    … # somehow test prog and returns true or false
```

Informal Proof (cont'd)

Define a function ***myVirus***:

```
def myVirus(): # consider myVirus as a program
    if isVirus(myVirus):
        return # do nothing
    else:
        infectOtherPrograms()
        destroyUserData()
        return
```

Self contradictory

Cohen's Conclusion

- Precise virus detection is not decidable.

- Virus removal (AV) is not always guaranteed because it is dependent on virus detection.

Simplest Malware Detection

- Compute hashes of malware samples
- Compute hashes for target files and find ones that match with one of the malware hashes (a.k.a. signatures)

What's wrong?

Easy to Bypass

- Add a dummy (dead) code
- Reorder instructions
- Replace instructions with semantically equivalent ones

Hash-based detection is still used in AV, why?

Pattern Matching (RegExp)

```
closeDoc{-35}setTimeOut{-30}addAnnot...
```

A ClamAV signature for CVE-2016-0931
(Adobe Acrobat PDF exploit)

Defeating Pattern-based Detection

Signature: `(\xb0\x0b)(.*)(\xcd\x80)`

```
mov al, 0xb
int 0x80
```

↓

Bypassing signature-based detection is so easy!

```
mov al, 0xa
inc al
int 0x80
```

Polymorphism

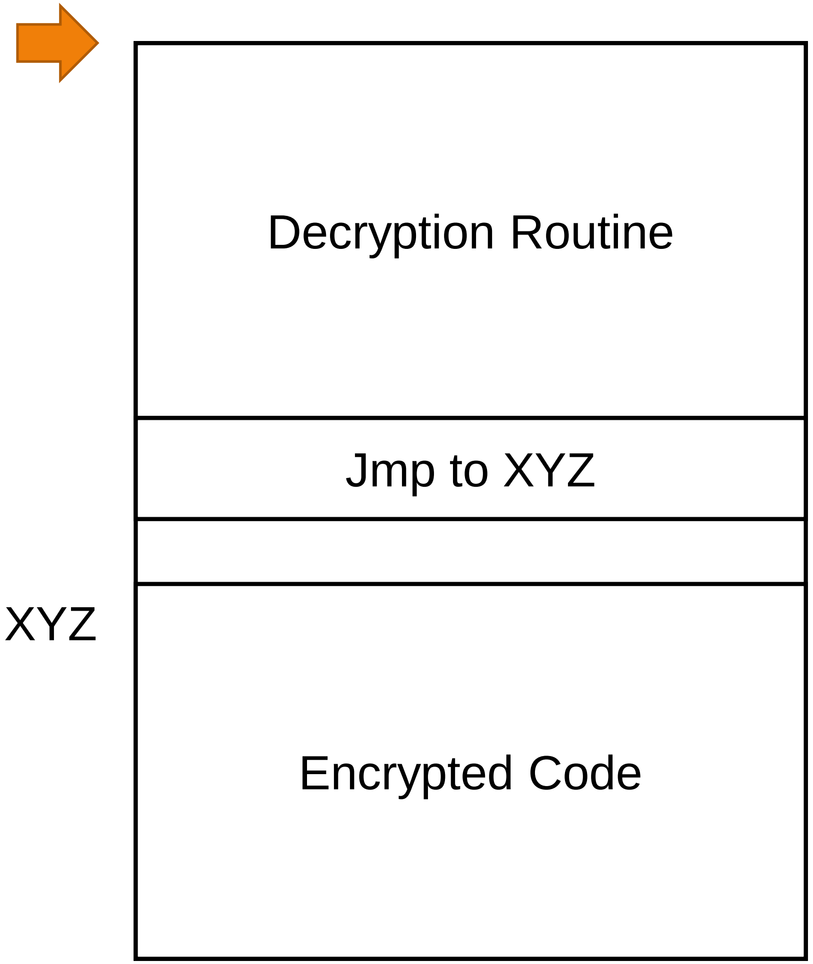
Change the form of malware when it propagates in order to ***bypass pattern matching***

Changing the Form?

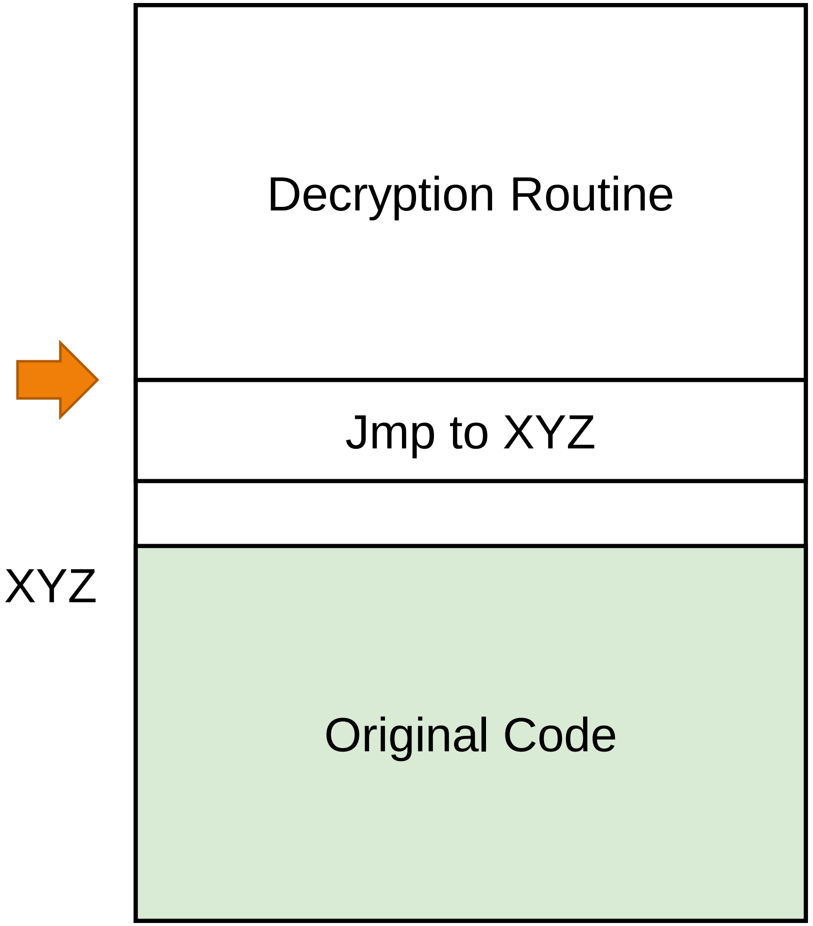
- Malicious use:
  Bypass malicious code detection (≈ Intrusion detection)

- Benign use:
  Software protection (make reverse engineering difficult)

Polymorphism Example



Polymorphism Example



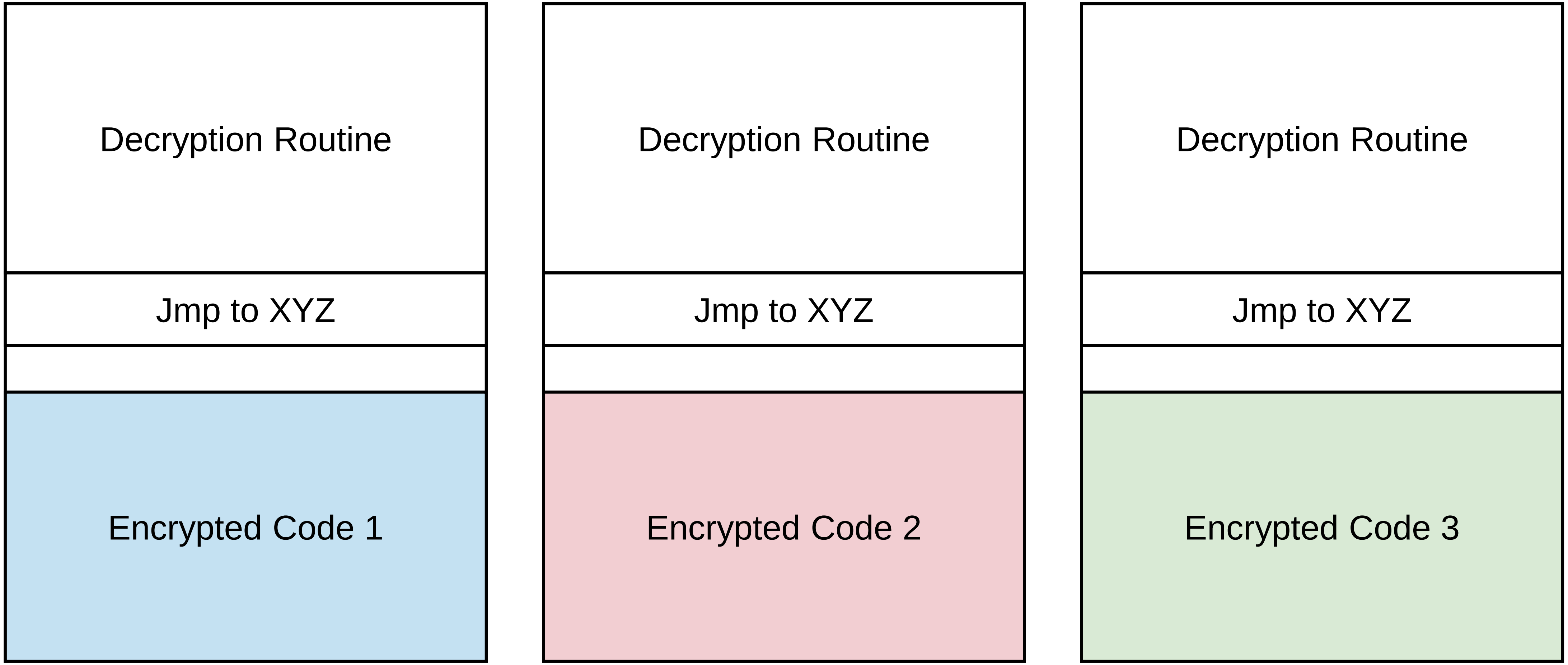
XYZ is often called ***OEP*** (Original Entry Point)

We can produce millions of distinct binaries (with the same semantics) by just changing the encryption key

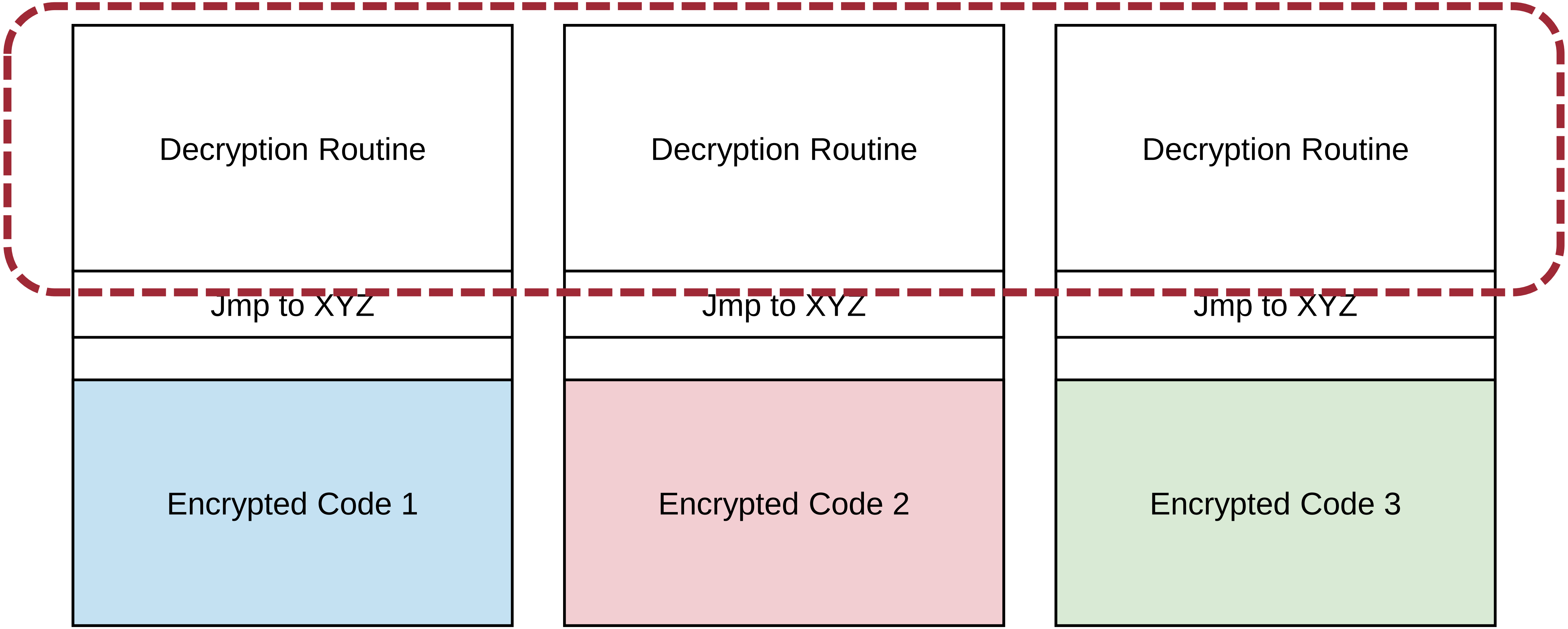
Self-Modifying Code

- Code that alters its own instructions while it is running
- W ^ X (Write xor eXecute) policy of modern OS?

Polymorphism Example



Checking Decryption Routine



Possible to Create Signatures

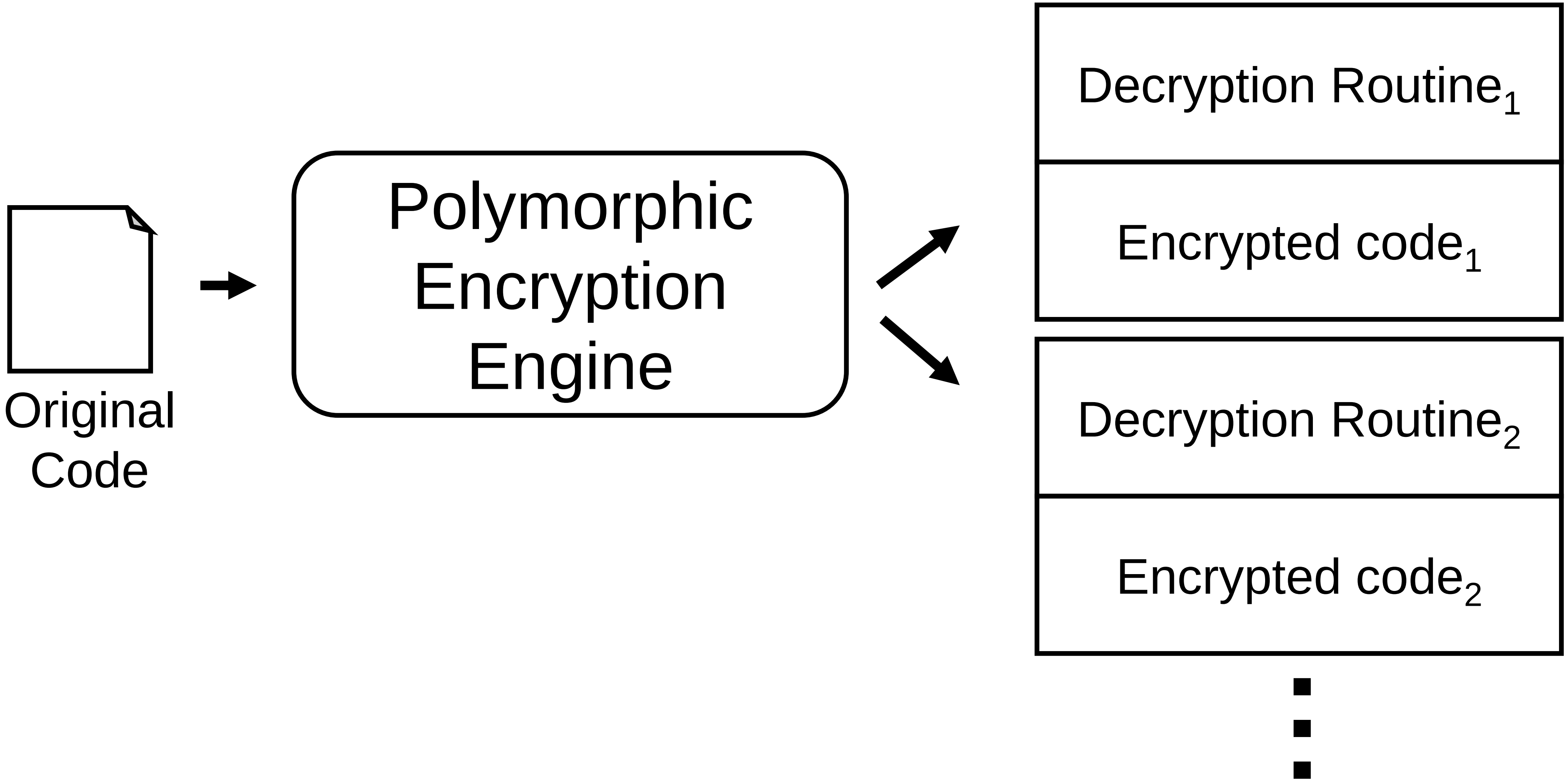
Decryption Routine	Decryption Routine	Decryption Routine

Next Question:
Can we also make the decryption routine polymorphic?

Polymorphic Encryption

Make the encryption/decryption routine unique!

Polymorphic Encryption (cont'd)



Polymorphic Encryption Example

```
for ( int i = 0; i < codeLen / 4; i++ ) {
  v = in[i]; // for every 4-byte value of the orig code
  key[i] = random_int(); // random 4-byte int
  op[i] = random_op(); // random operation
  switch ( op[i] ) {
  case ADD: v += key[i]; break;
  case SUB: v -= key[i]; break;
  case XOR: v ^= key[i]; break;
  ... // omitted
  }
  out[i] = v; // store the encrypted code
}
```

Polymorphic Decryption Example

```
for ( int i = 0; i < codeLen / 4; i++ ) {
  v = in[i]; // for every 4-byte of the encrypted code
  k = key[i];
  switch ( op[i] ) {
  case ADD: v -= k; break;
  case SUB: v += k; break;
  case XOR: v ^= k; break;
  ...
  }
  out[i] = v; // store the decrypted code
}
// The encrypted code can be located here (self-modifying)
```

Can We Still Write Signatures?

- Signature database will easily blow up
- Simple static pattern matching does not help anymore

Any issues in polymorphic encryption?

In-Memory Detection

- The same original code will be eventually unpacked to memory at some point
- Memory-based scanning still works! (no more static detection)
- Generic unpacking technique exists

Performance vs. Security

- Performance really matters
- **Signature-based detection** is still largely popular

Fun Fact

- Signature-based detection is fast
- But it gets slower as we add more signatures

More # of malicious apps
⇒ More # of signatures
⇒ More memory
⇒ Poor cache performance
⇒ Slow!

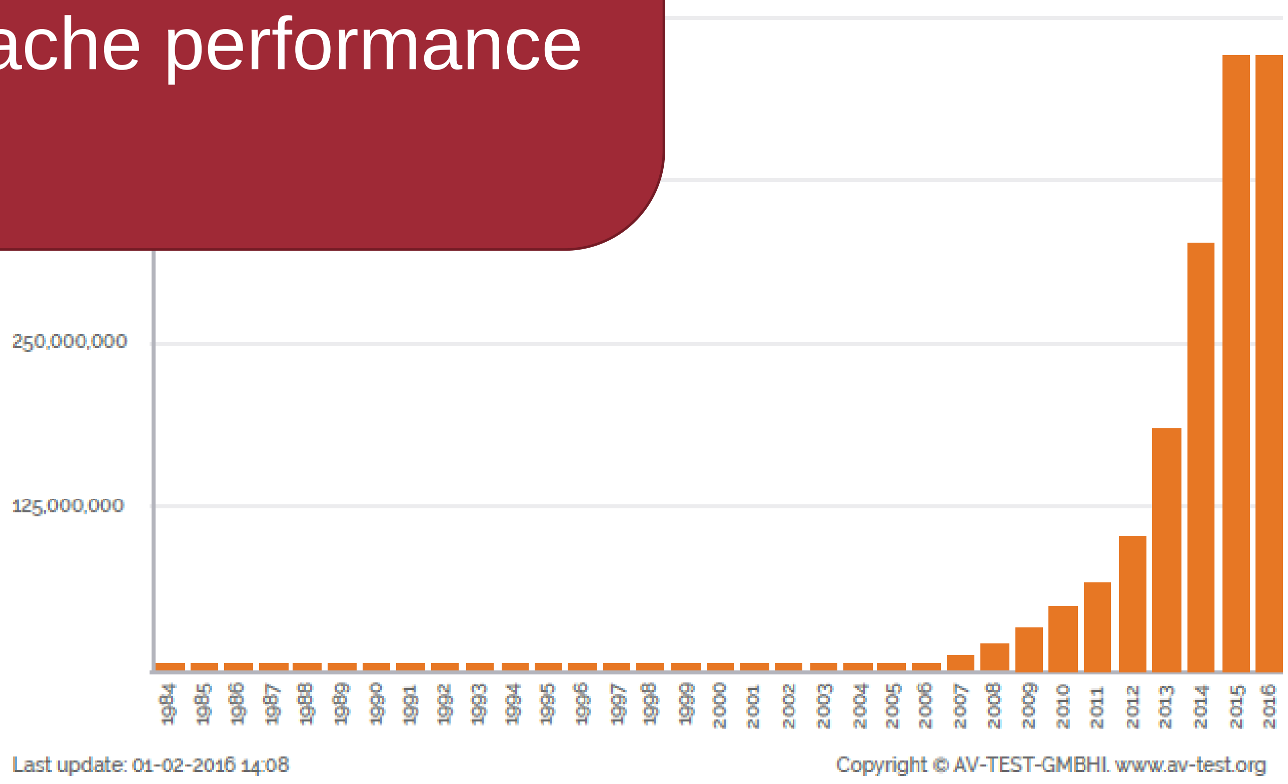


Figure 1: Total known malware

Motivation

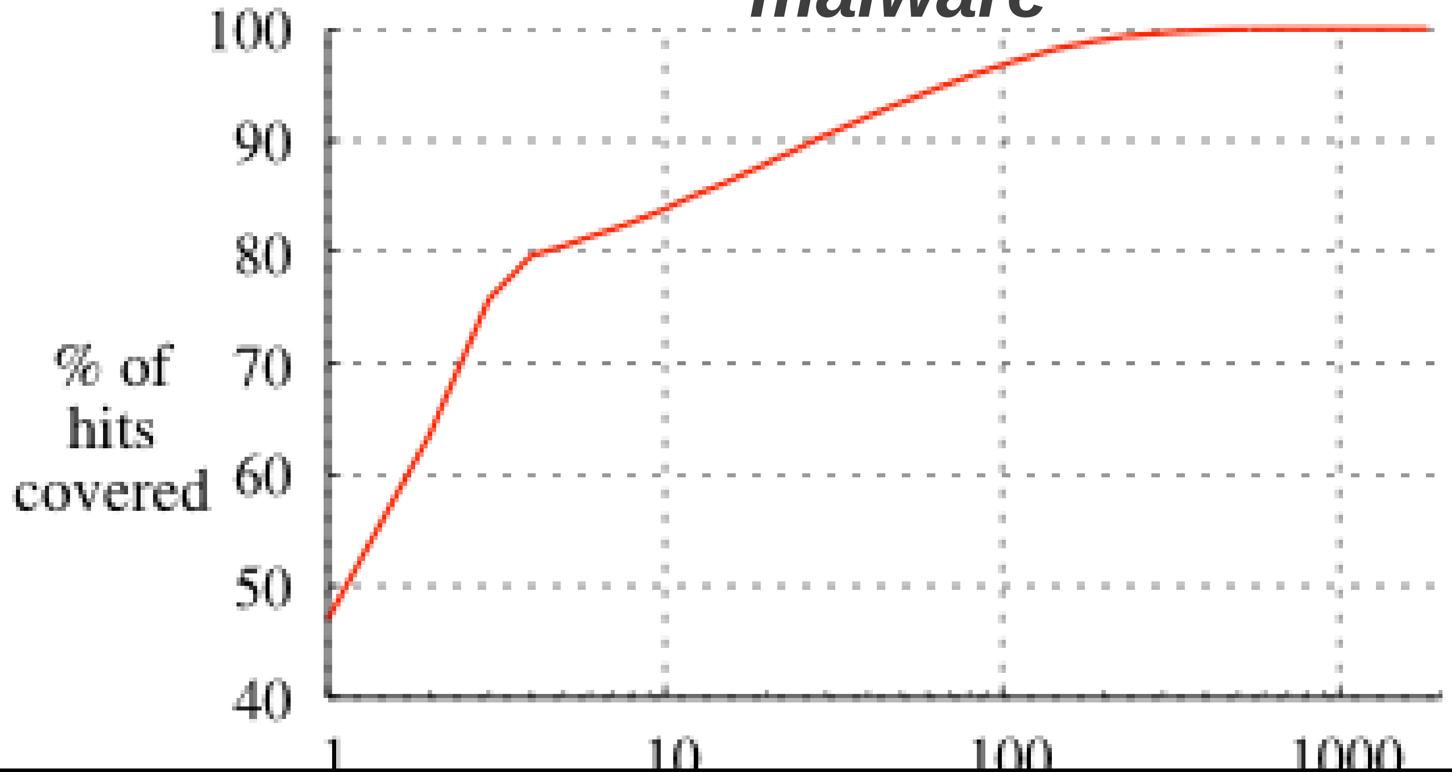
Can we make signature-based scanning fast and more scalable?

SplitScreen: Enabling Efficient, Distributed Malware Detection, ***NSDI 2010***

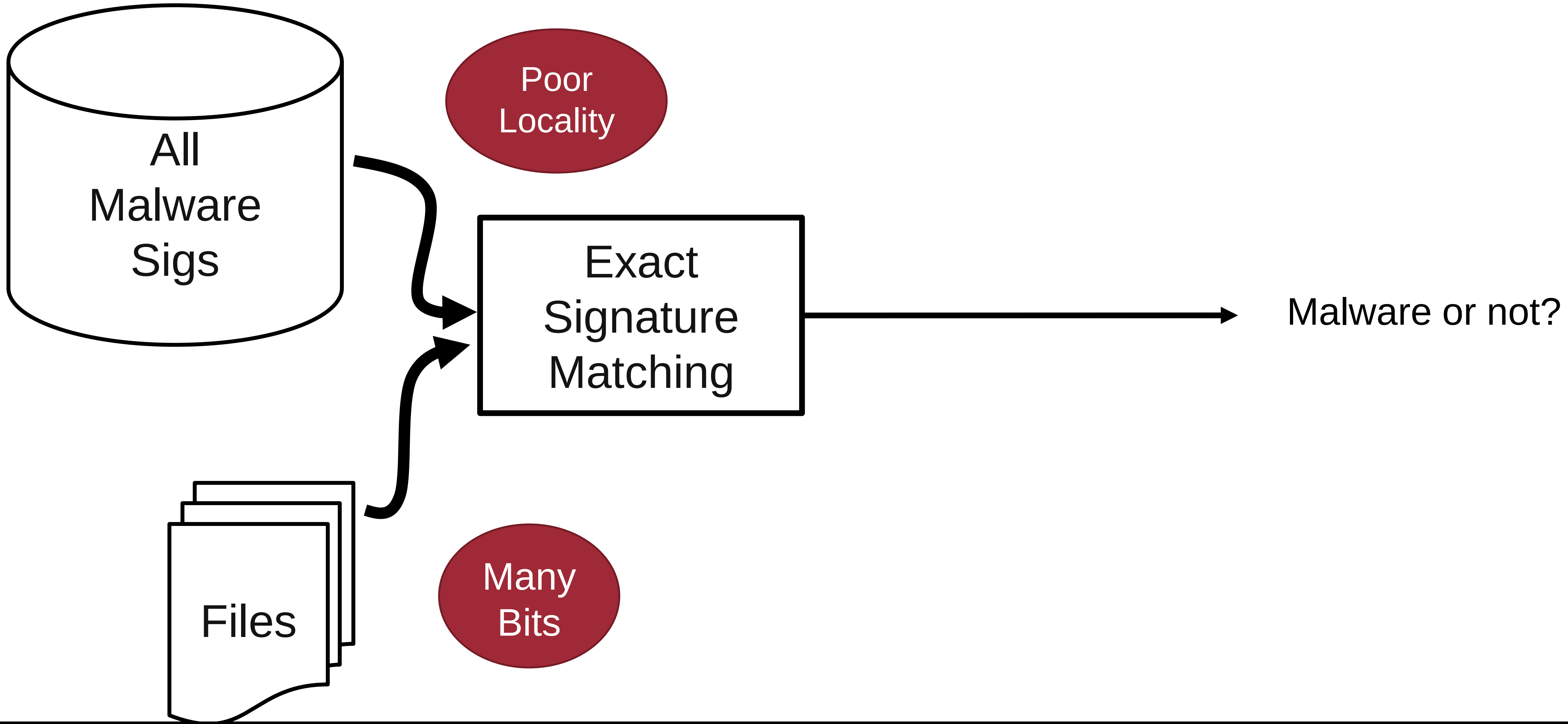
Opportunity: Fewer Signatures Matched

4 month study of CMU email malware

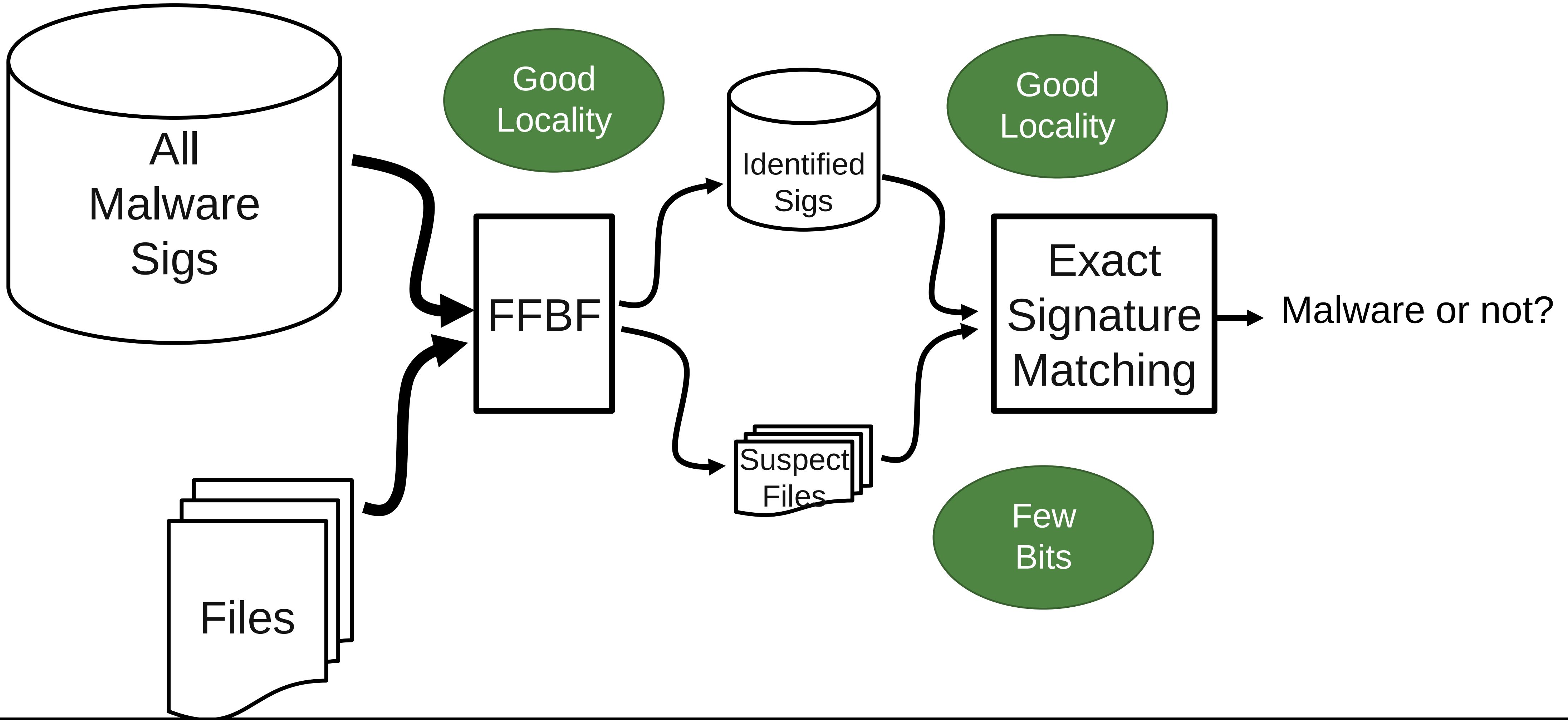
< 1% of signatures used by ClamAV for all malware



Traditional Signature-based AV



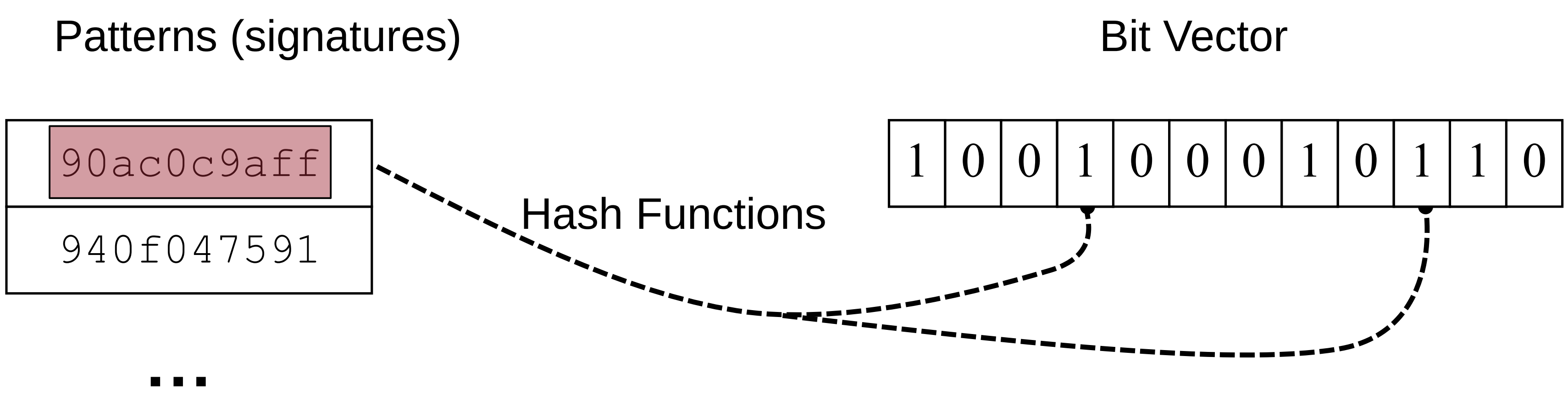
SplitScreen Architecture



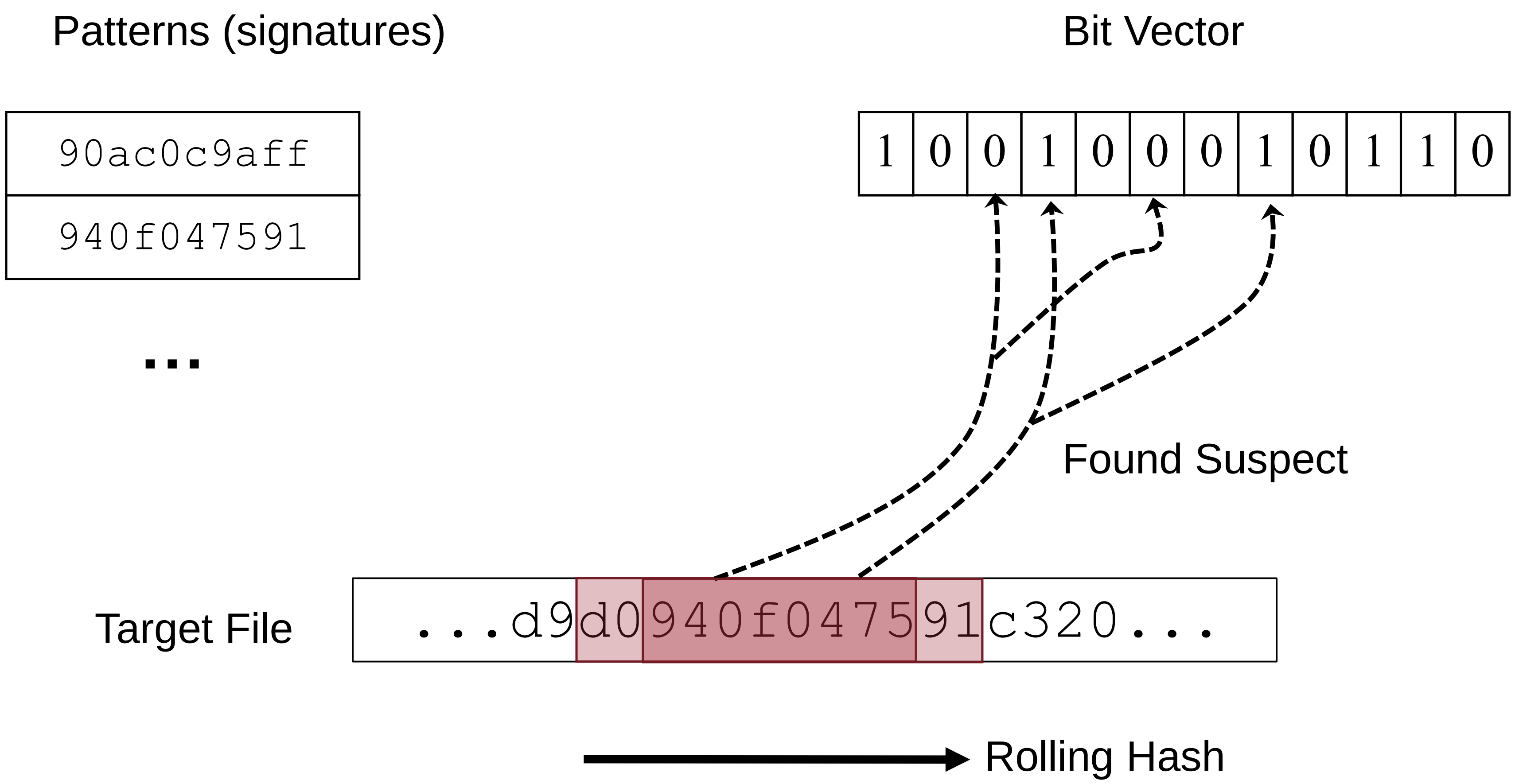
FFBF: Feed-Forward Bloom Filter

- A modified Bloom filter
- Quick matching with one-sided error
  - False positives possible
  - False negatives not possible

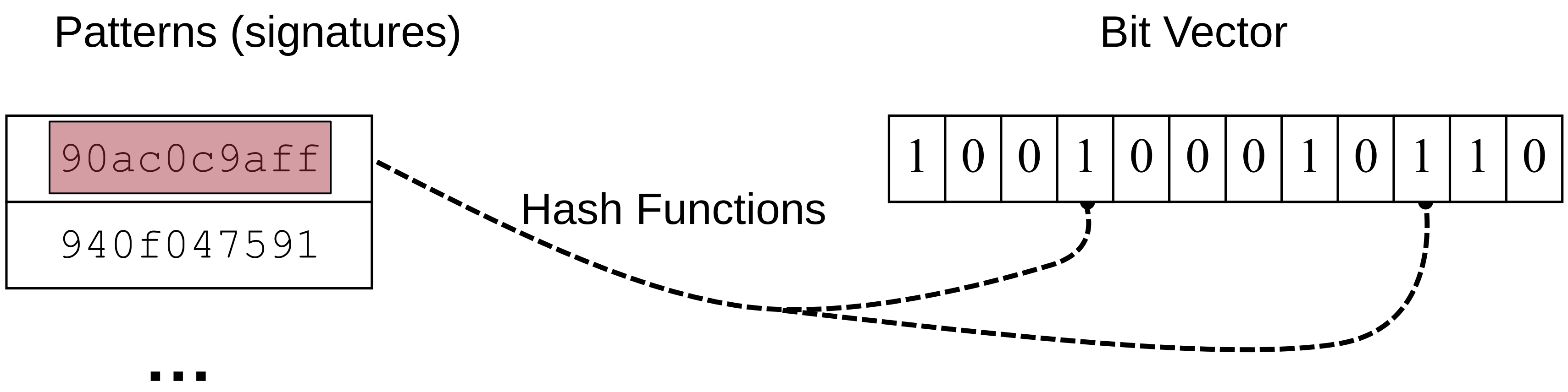
Traditional Bloom Filter



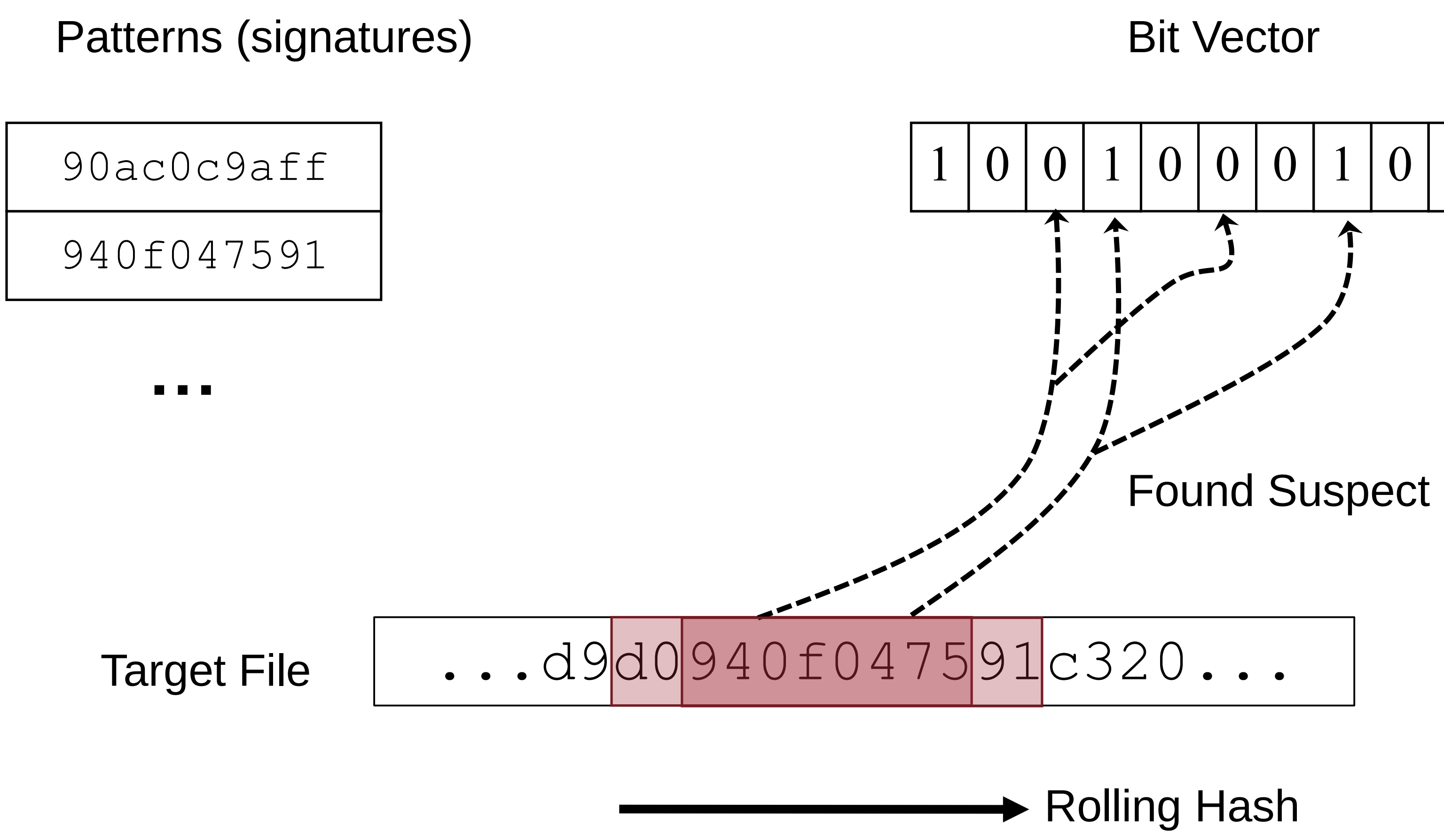
Traditional Bloom Filter



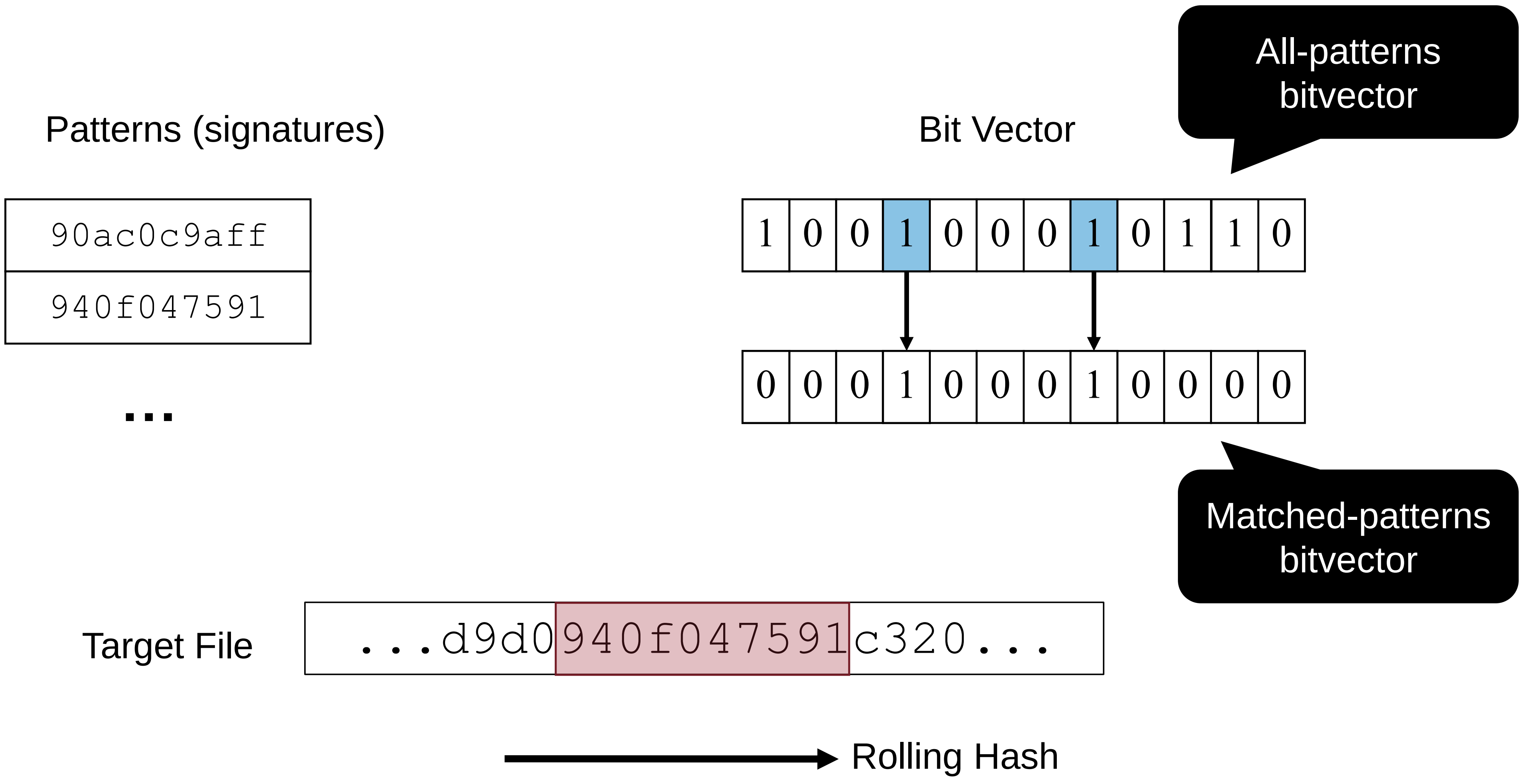
Feed-Forward Bloom Filter



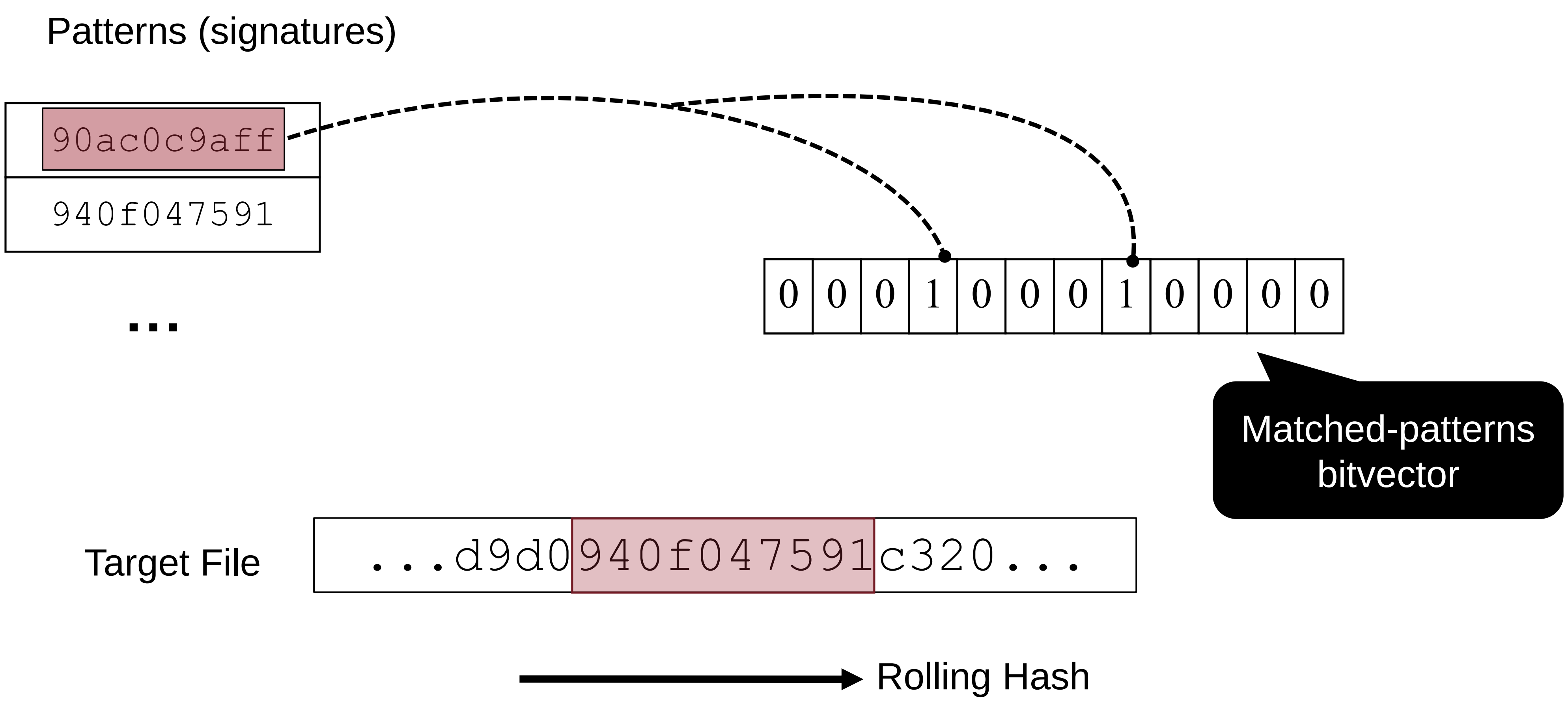
Feed-Forward Bloom Filter



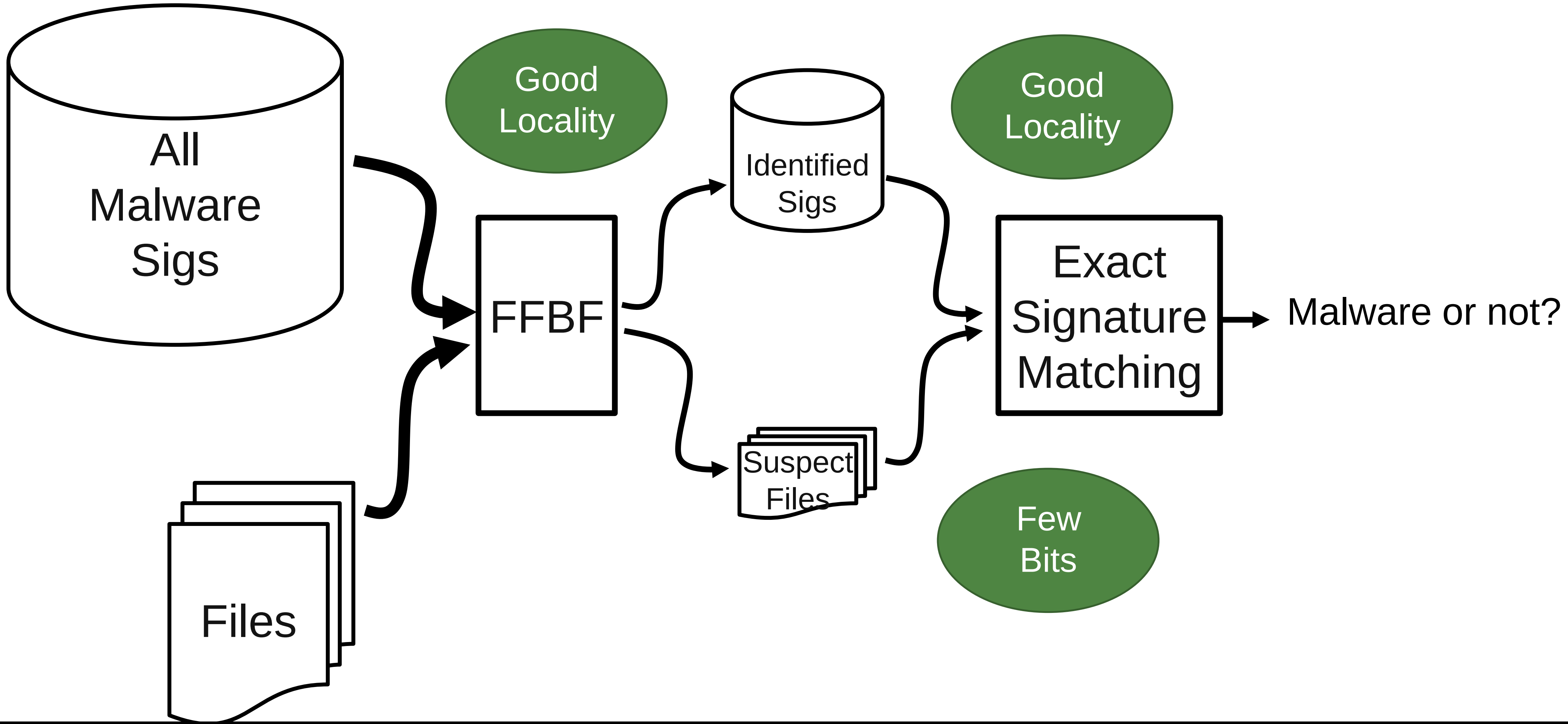
Feed-Forward Bloom Filter



Pattern Filtering



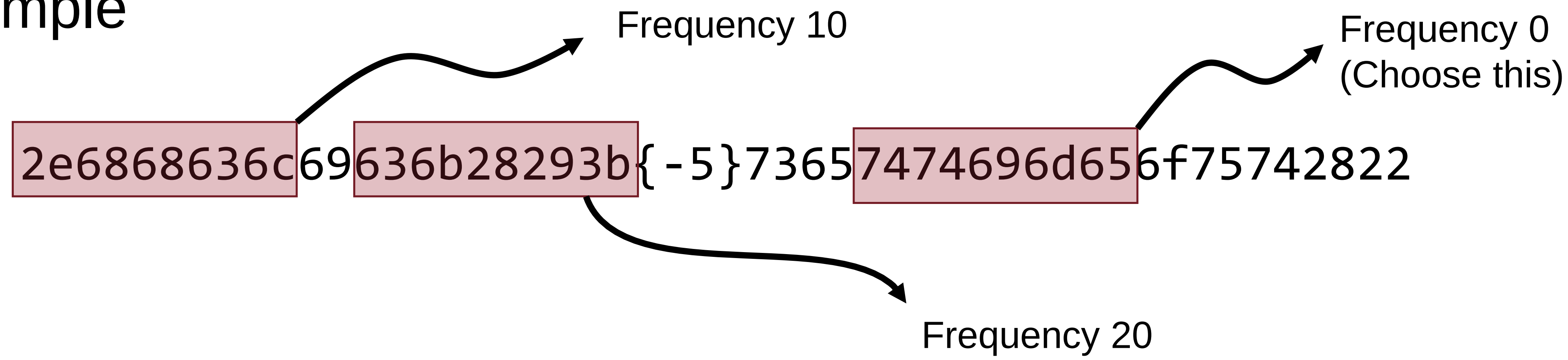
SplitScreen Recap



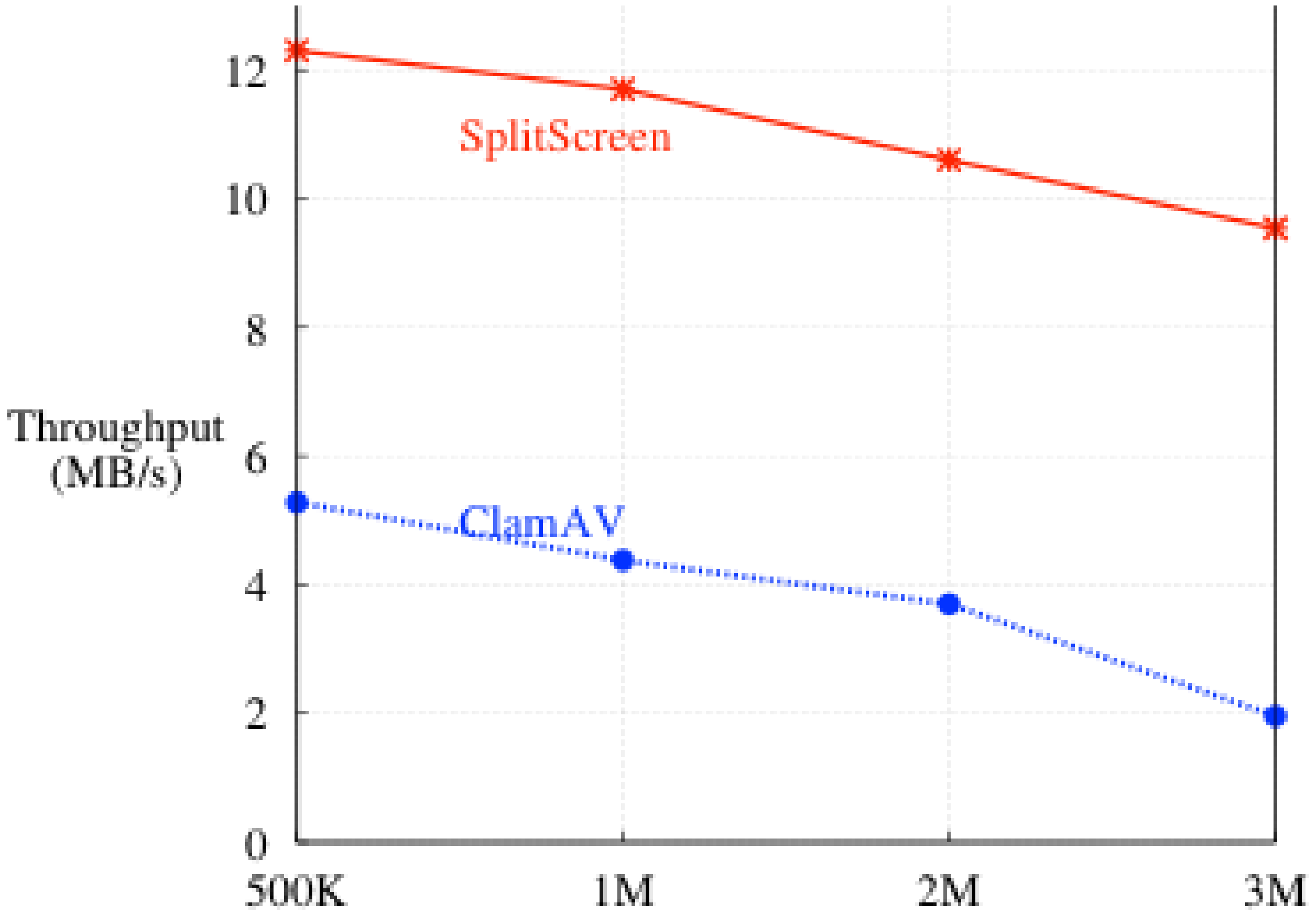
Frequency-based Signature Fragment Selection

- Choose signature fragment based-upon frequency when initialize FFBF. (Choose a fragment of the ***least*** frequency)

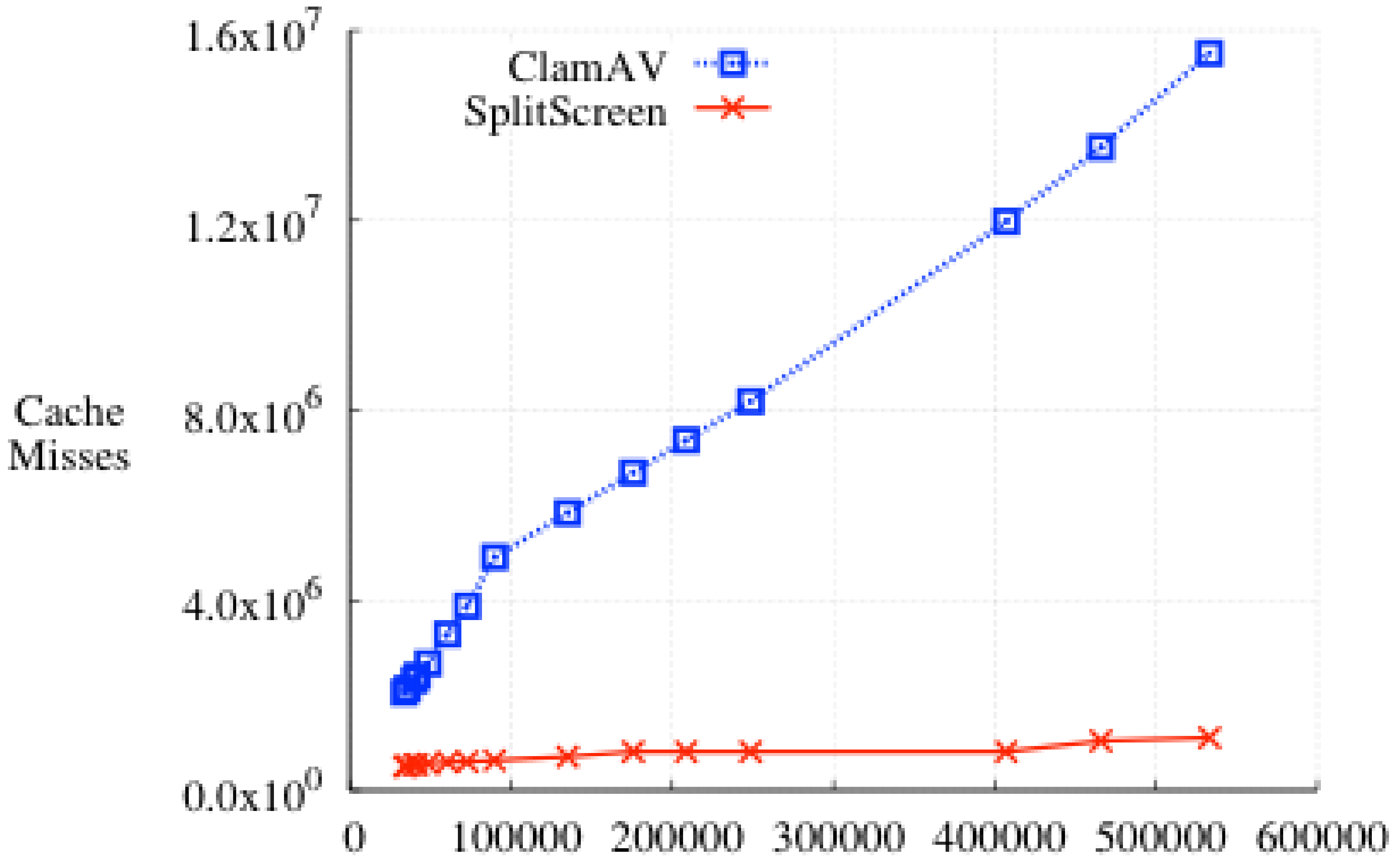
- Example



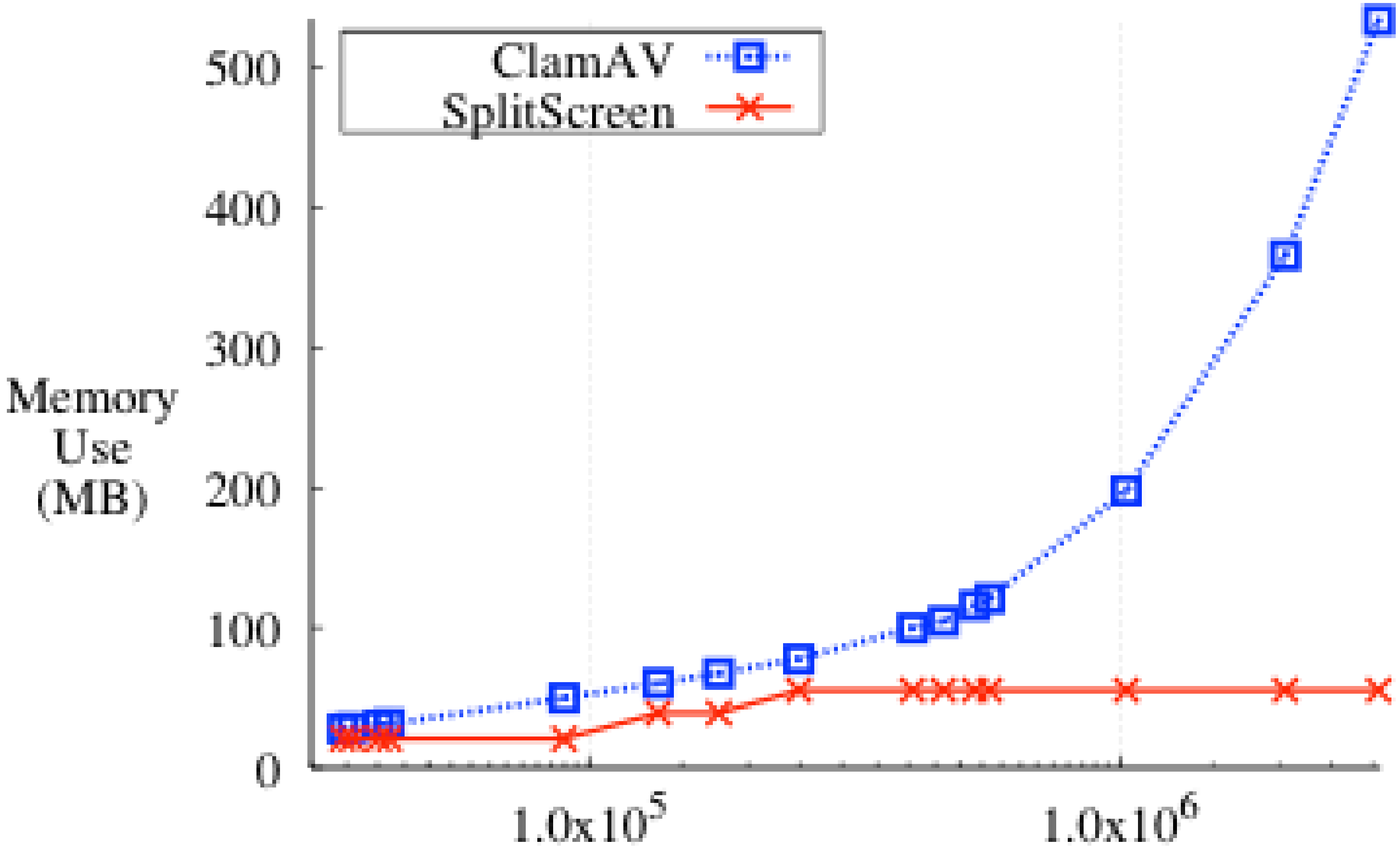
Thoughput (1.6 GB Clean Files)



Better Cache Performance



Less Memory



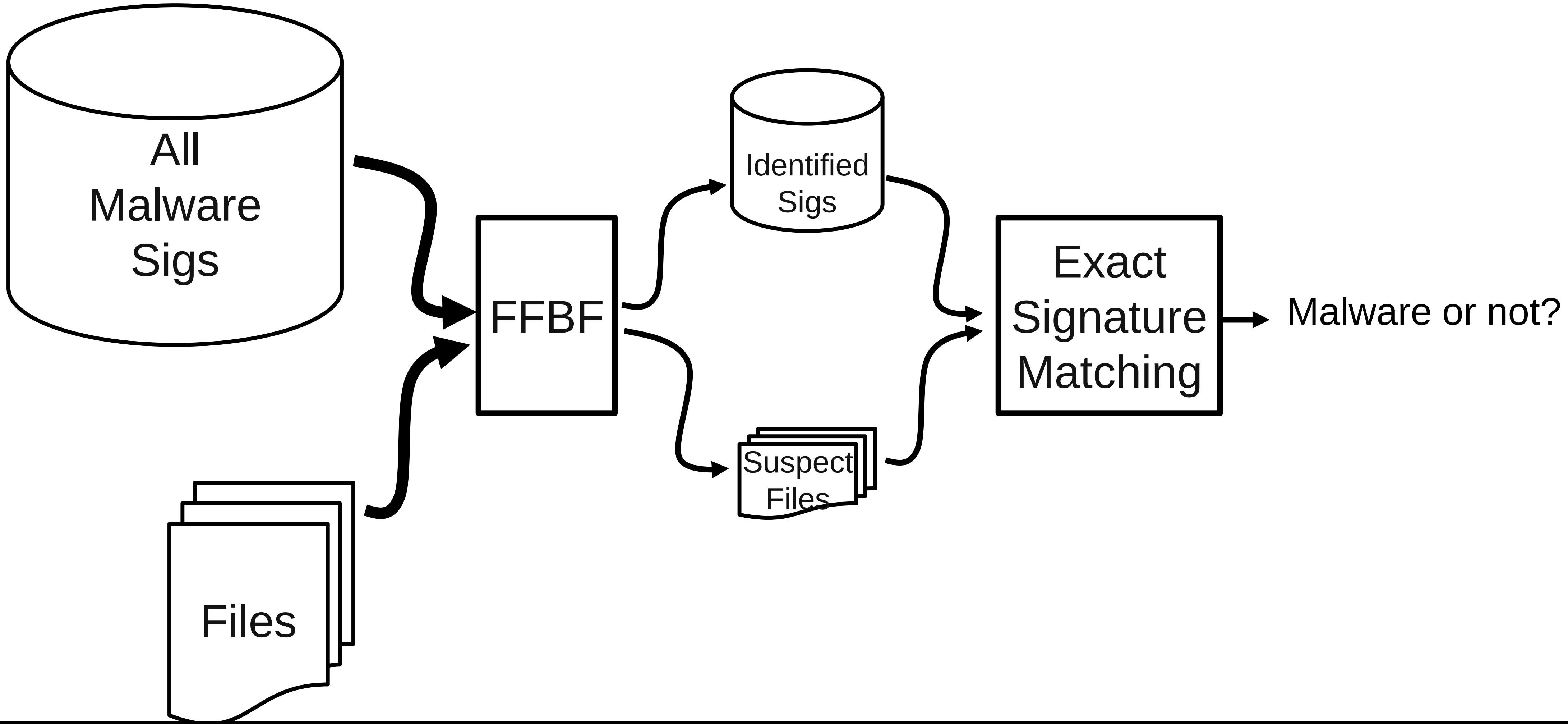
ClamAV
SplitScreen
Memory Use (MB)
500
400
300
200
100
0
1.0x10^5
1.0x10^6

Signature Distribution Cost?

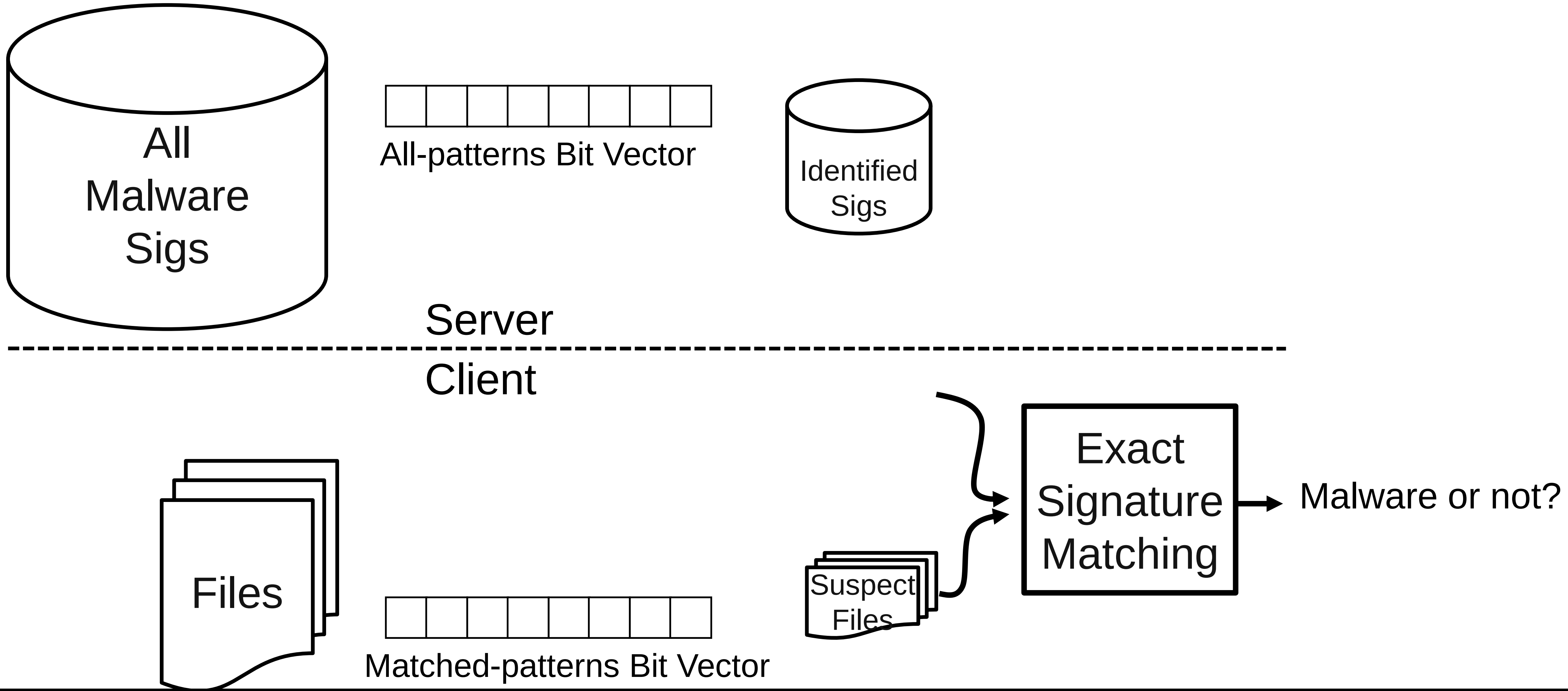
As the signature database gets larger, distributing it also becomes expensive!

SplitScreen allows
on-demand signature distribution

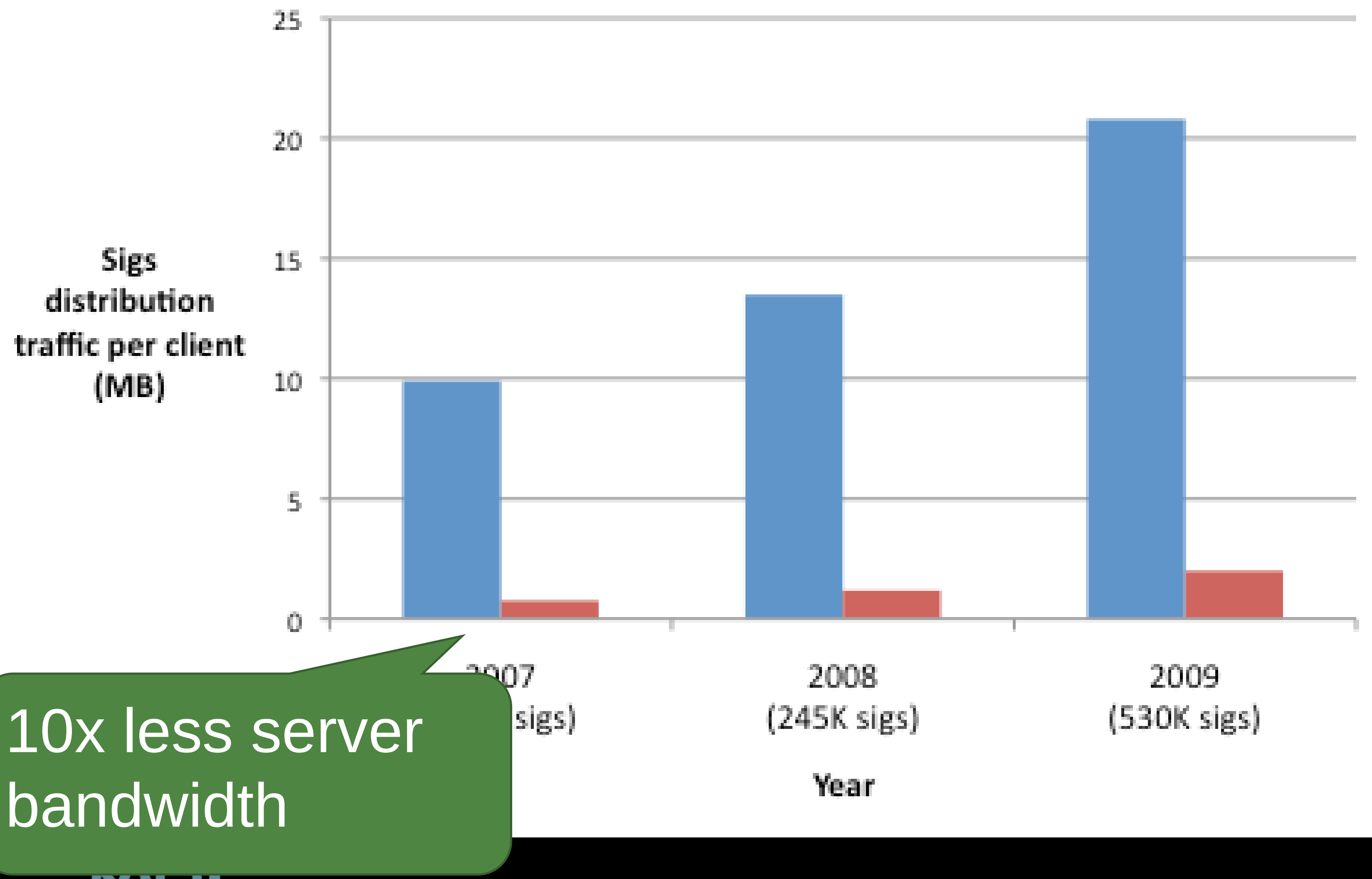
On-Demand Signature Distribution



On-Demand Signature Distribution



Lower Signature Distribution Cost



Conclusion

- Perfect AV is not feasible

- Infinite war between malware authors and defenders
  - Hash-based detection
  - Signature-based detection
  - Polymorphic malware
  - Polymorphic encryption

- Signature-based detection is still critical, and SplitScreen enables efficient and distributed malware detection

Questions?