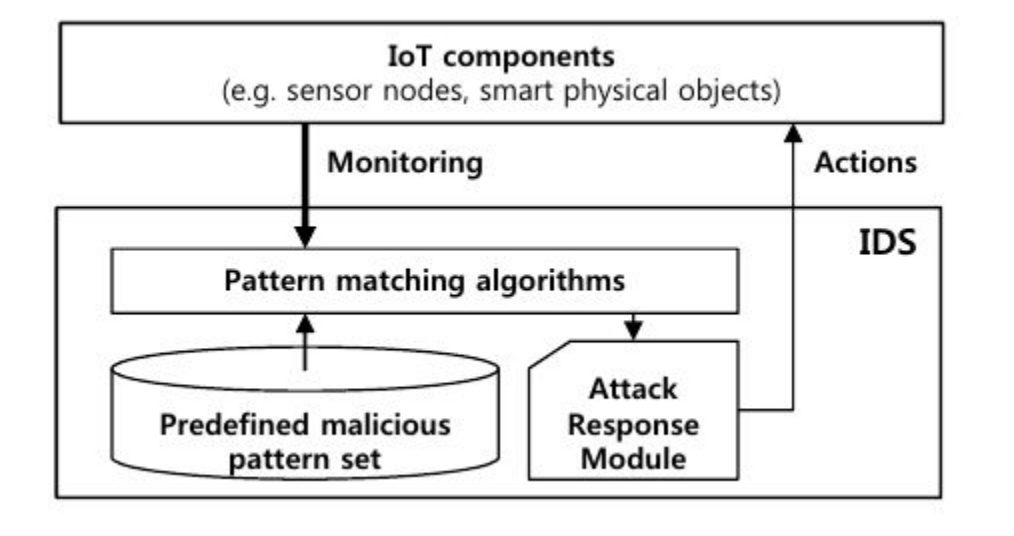
“A Malicious Pattern Detection Engine for Embedded Security Systems in the Internet of Things”

Doohwan Oh, Deokho Kim and Won Woo Ro

Daryl Hernández C.

Introducción

IOT + IDS


IoT components
(e.g. sensor nodes, smart physical objects)
Monitoring
Actions
IDS
Pattern matching algorithms
Predefined malicious pattern set
Attack Response Module

Un poco de contexto

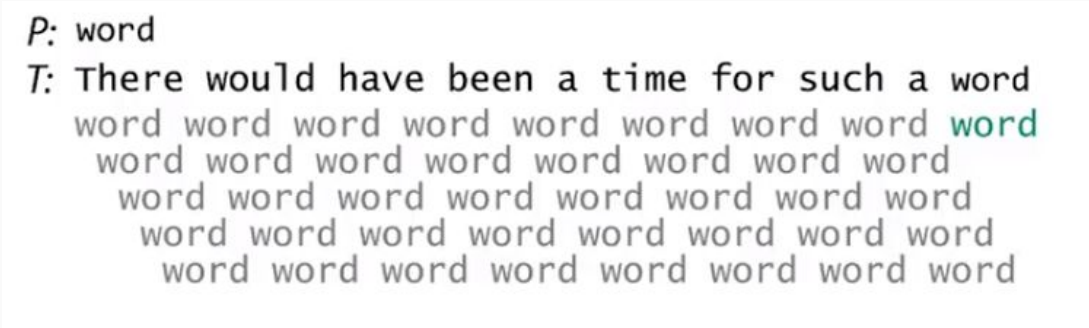
Búsqueda de patrones en cadenas de texto

- Algoritmo Naive
- Algoritmo de Boyer - Moore
- Algoritmo de Wu - Manber

Naïve

El algoritmo Naive es el más simple de todos.

Consiste en buscar carácter por carácter hasta que haya coincidencia.



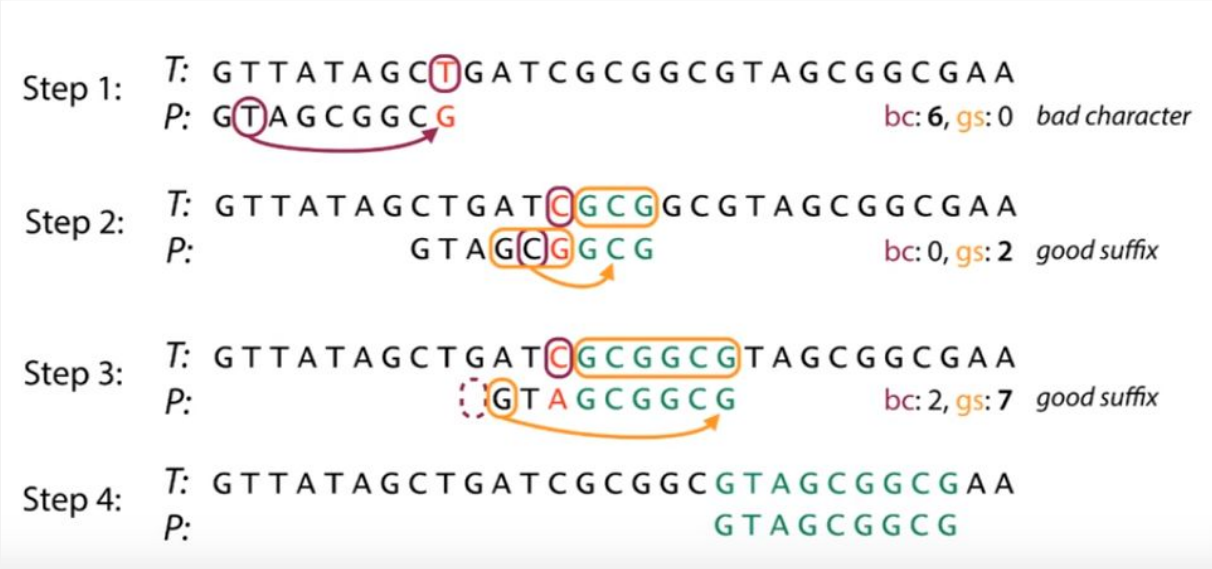
Boyer - Moore

Punto de referencia estándar cuando se trata de eficiencia.

Consiste en “saltos” (shifts).

2 reglas:

1. Carácter erróneo
2. Buen sufijo



Wu - Manber

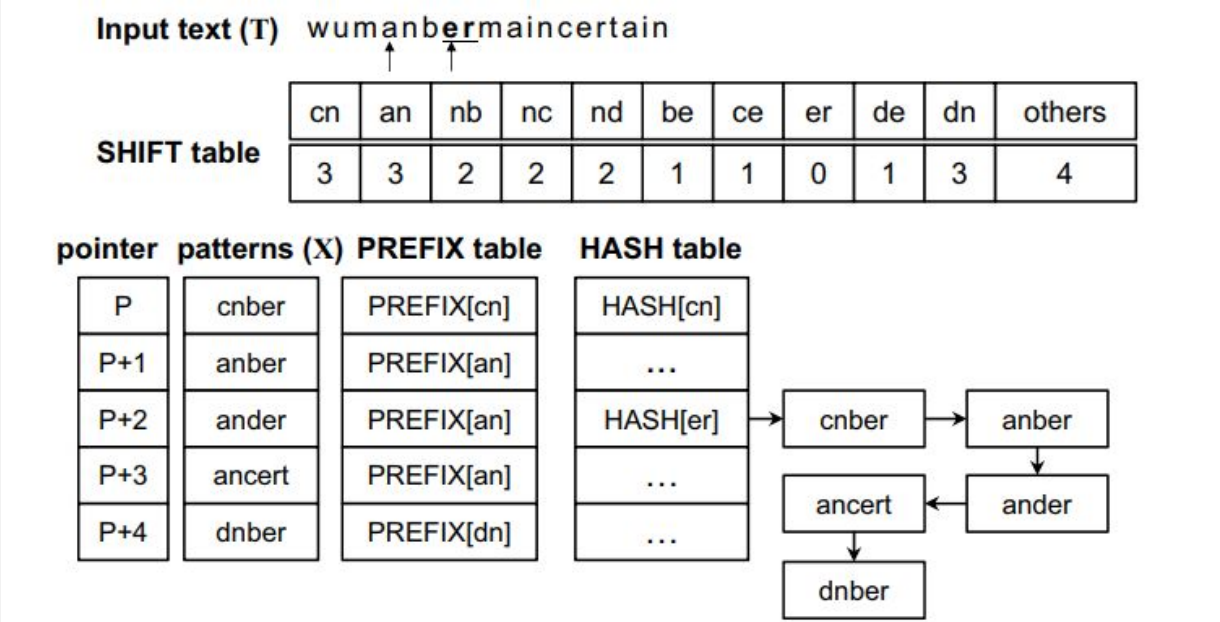
Algoritmo detrás de agrep.

Basado en Boyer - Moore.

Permite la búsqueda de múltiples patrones dentro de un texto.

Posee 2 etapas:

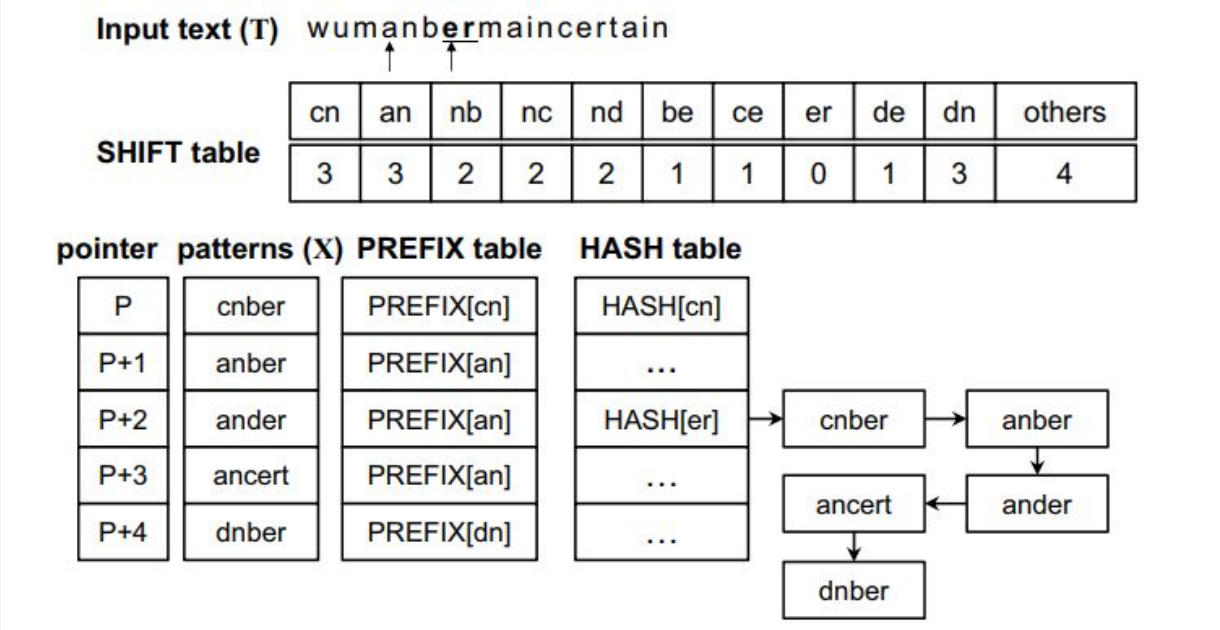
1. Preprocesamiento (3 tablas)
   a. SHITF
   b. HASH
   c. PREFIX
2. Escaneo



Wu - Manber

2. Escaneo

a. Calcular hash (h) del bloque, ver SHIFT(h), si es 0, avanzar a siguiente etapa.
b. Calcular hash (p) del prefijo, comparar con el patrón, si coinciden, avanzar a siguiente etapa.
c. Comparar resto de caracteres hasta que: no coincidan, coincidan todos (patrón encontrado). Regresar a b)



Modificaciones propuestas

- Adición de “Desplazamiento Auxiliar”
- Adición de “Decisión Temprana”
- Adición de “Búsqueda de Bordes”

¿Por qué?

- Es posible saltar comparaciones luego de encontrar una ocurrencia de patrón.
- Los patrones se ordenan por sufijo, lo que obliga a comparar todos los prefijos.

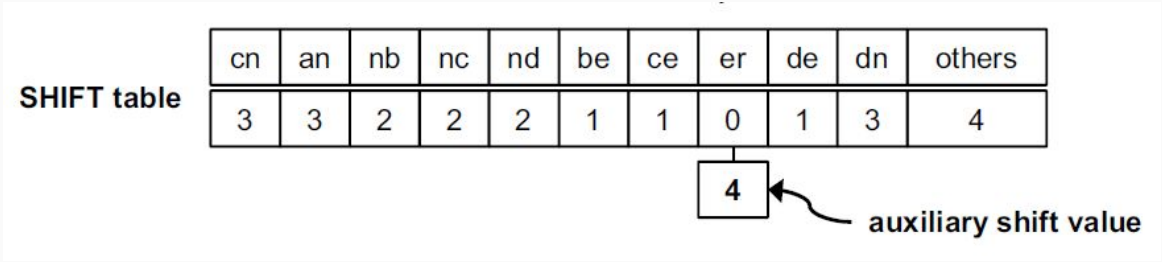
Desplazamiento Auxiliar

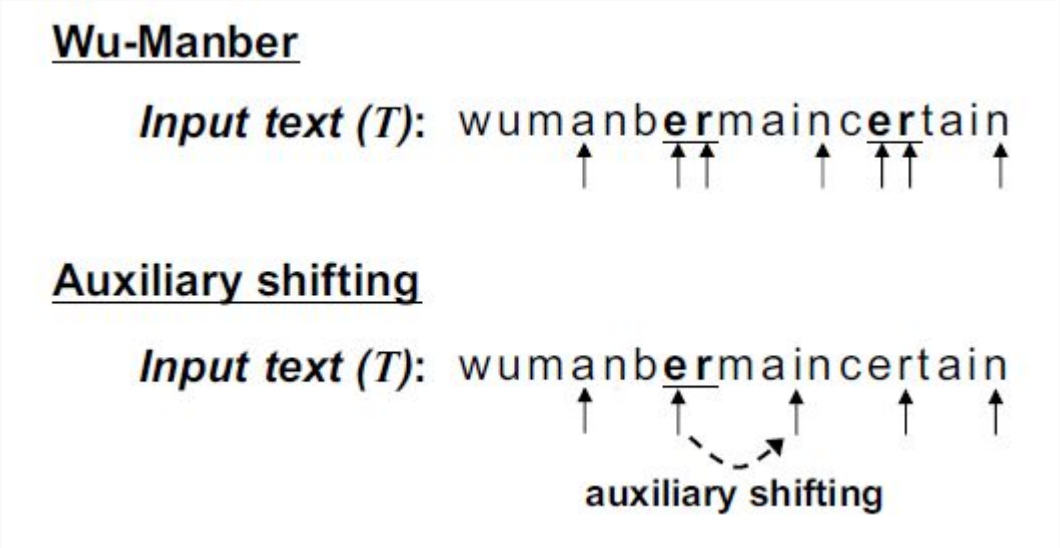
Luego de completar el paso b) - c) del algoritmo de Wu - Manber, ya se examinaron todas las posibilidades con el bloque, por lo que es seguro avanzar en más de un espacio.

Se define ASHIFT cuando SHIFT es nulo, en cuyo caso toma su valor anterior.

Se modifica el paso b):

Luego de comparar todos los patrones, avanzar tanto como ASHIFT.



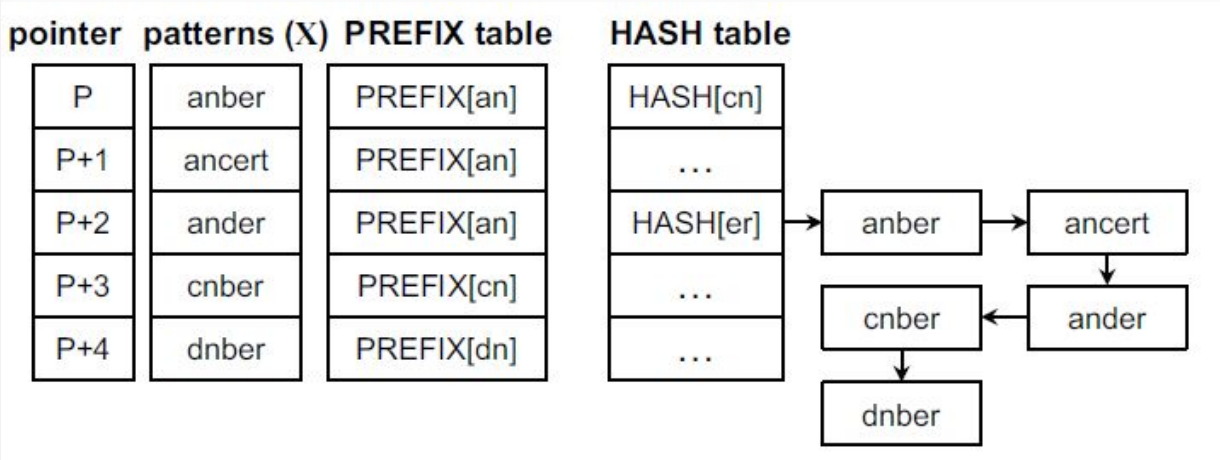


Decisión Temprana

Es posible omitir ciertas comparaciones si es que ordenamos los patrones contenidos en HASH.

Paso b): Si el prefijo no coincide y es mayor que p (o se compararon todos), desplazarse según ASHIFT y regresar a paso a). Si es menor, continuar.

Paso c): Cuando se encuentra una no coincidencia, y el valor del carácter del patrón es mayor que el del texto, desplazarse según ASHIFT y regresar a paso a)



Búsqueda de Bordes

Aún se puede optimizar la búsqueda de prefijos en patrones.

Paso b): Utilizando búsqueda binaria, escanear los prefijos en HASH[h] para encontrar el borde del mismo grupo de prefijos, si no existe, desplazarse según ASHIFT y regresar a a).

Paso c): Para patrones dentro del grupo, el resto de los caracteres se comparan, si todos coinciden, el patrón fue encontrado, regresar a a). Si ocurre una no coincidencia, o se prueban todos los patrones en el grupo, desplazarse según ASHIFT, volver a a).

Análisis teórico

- Adición de “Desplazamiento Auxiliar”
- Adición de “Decisión Temprana”
- Adición de “Búsqueda de Bordes”

- PM: Prefix Matching
- CM: Character Matching
- N: Número total de patrones
- Σ: Tamaño del alfabeto
- B: Tamaño de bloque
- B’: Tamaño de prefijo
- M: Largo promedio de patrones

*Para futuras imágenes

Análisis teórico

Wu - Manber:

$$E\{PM\} = \frac{N}{\Sigma^B}$$

$$E\{CM\} = \frac{N}{\Sigma^{B+B'}} \sum_{i=0}^{M-B'} \frac{1}{\Sigma^i}$$

$$O(N/\Sigma^B)$$

Decisión temprana:

$$E\{PM_{ED}\} = \frac{1}{\Sigma^{B'}} \sum_{i=1}^{\Sigma^{B'}} \frac{N}{\Sigma^{B+B'}} i$$

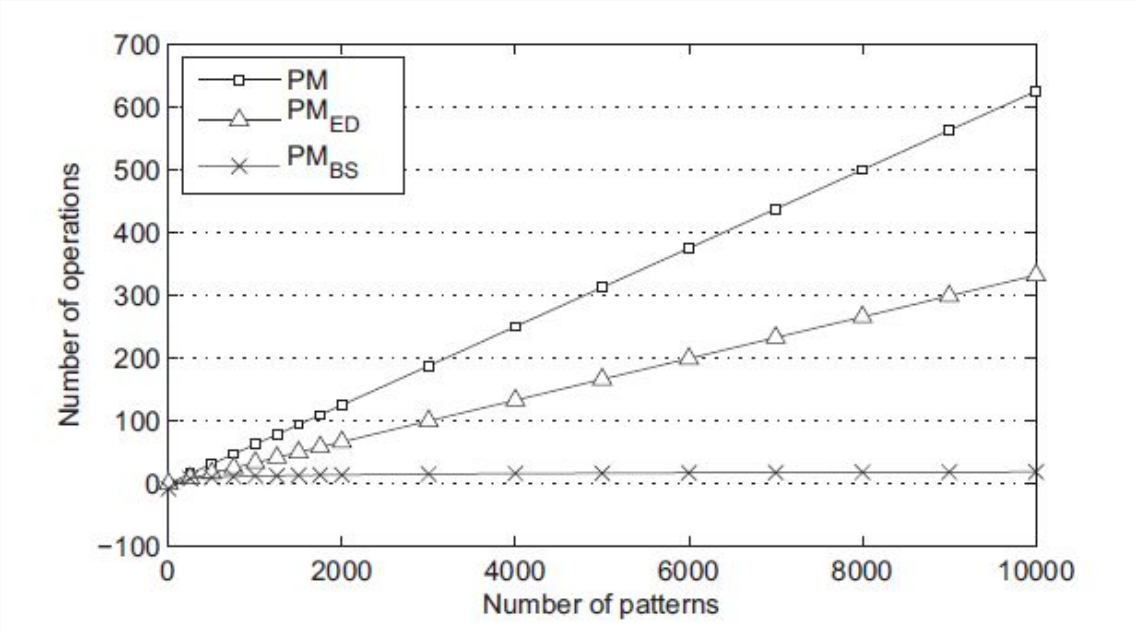
$$E\{CM_{ED}\} = \frac{N}{\Sigma^{B+B'}} \sum_{i=0}^{M-B'} \frac{1}{\Sigma^i} \sum_{i=1}^{\Sigma} \frac{i}{\Sigma^2}$$

$$O(N/\Sigma^B)$$

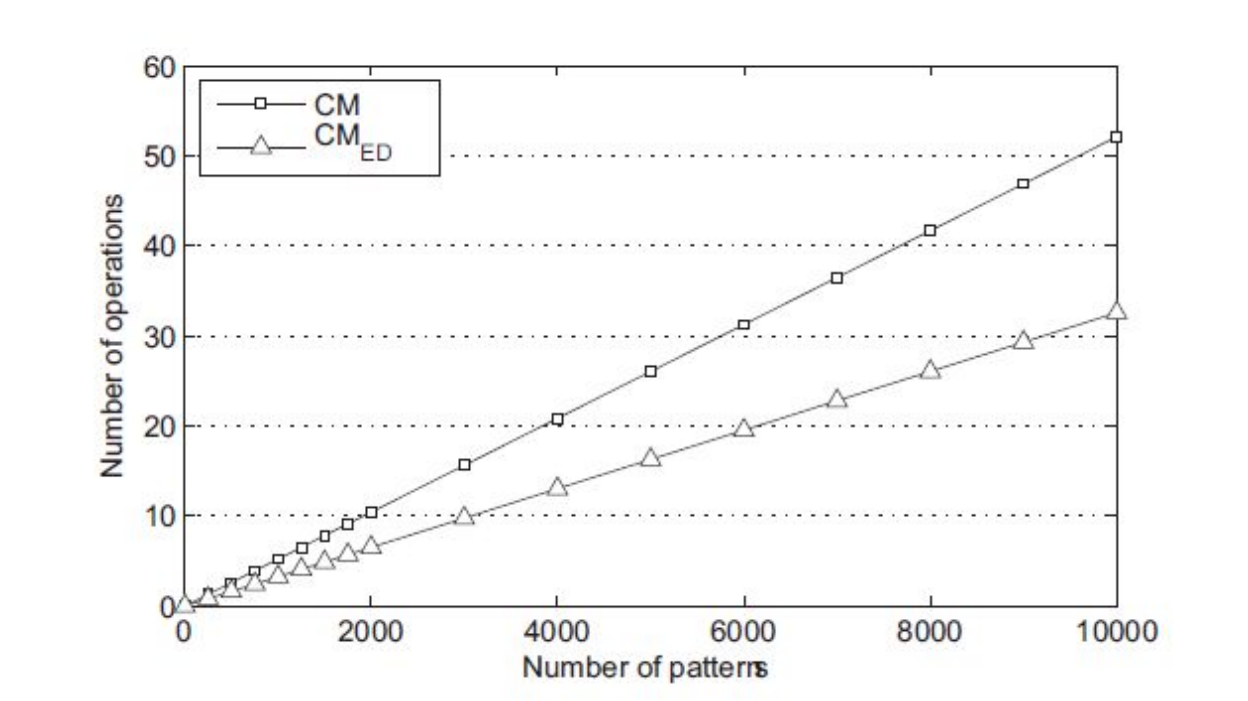
Búsqueda de Bordes:

$$E\{PM_{BS}\} = 2\log_2 \frac{N}{\Sigma^B}$$

$$O(N/(\Sigma^{B+B'}))$$



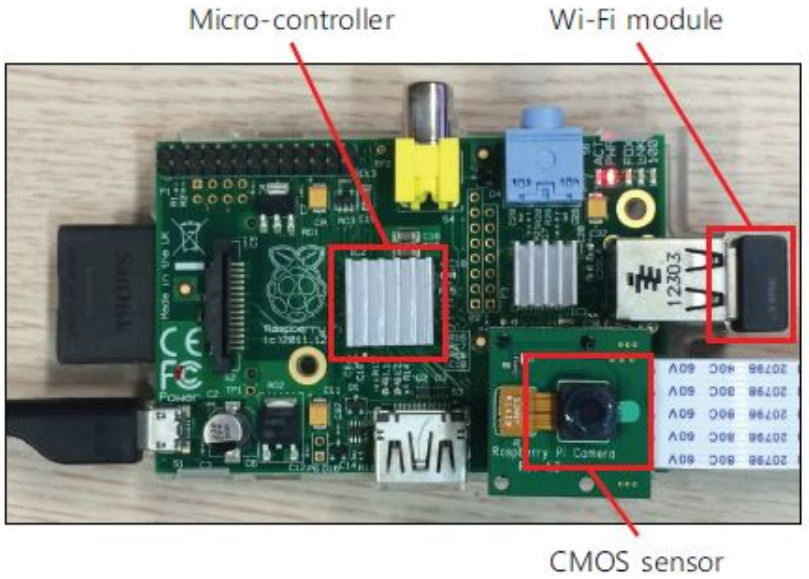
Ganancia de eficiencia en comparación de prefijos

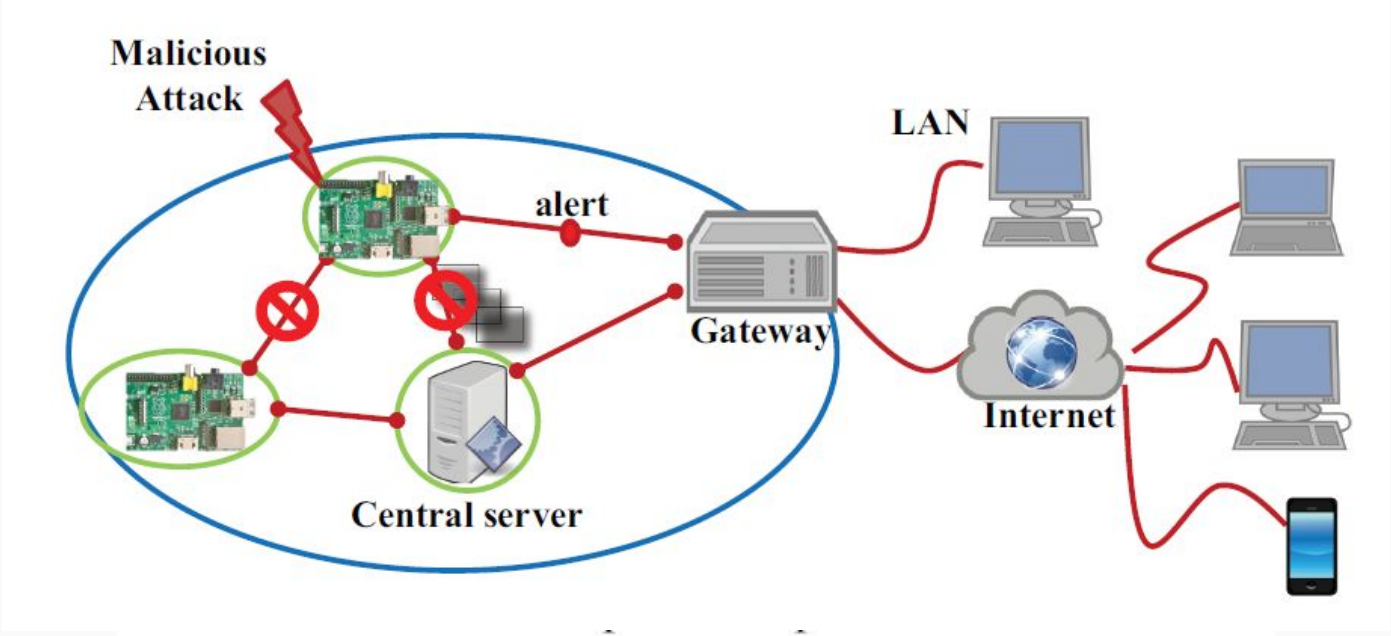


Ganancia de eficiencia en comparación de caracteres

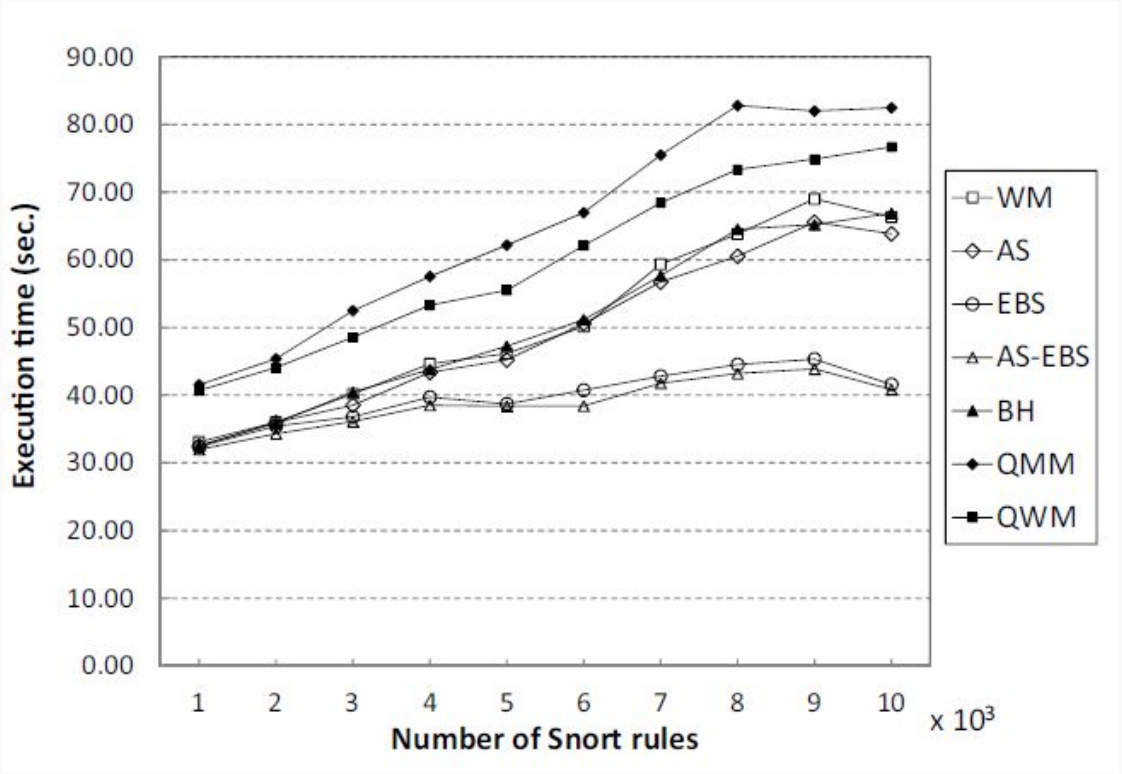
Implementación y Resultados

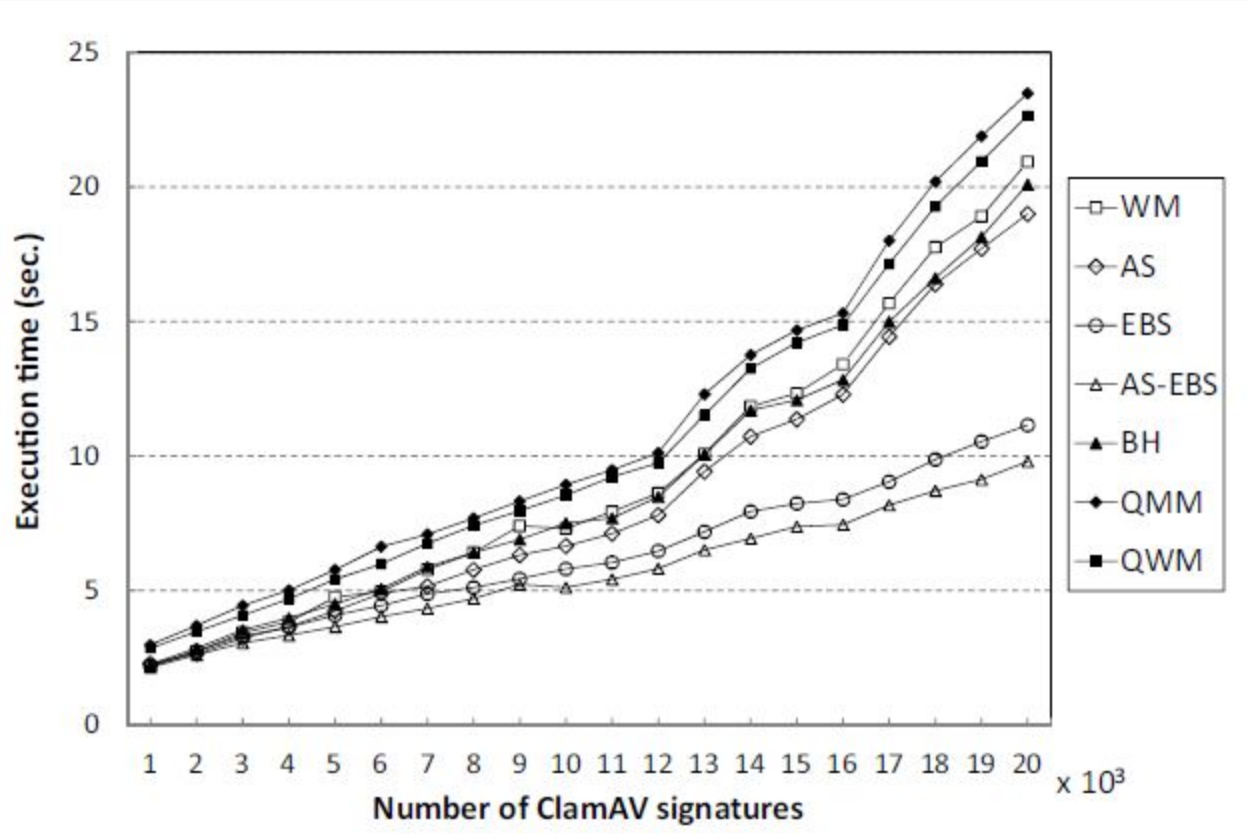
- Se utiliza una Raspberry Pi cuya tarea es transmitir video por la red.
- Se prueban 2 tipos de patrones, extraídos de Snort y ClamAV.
- Se compara el desempeño de las variaciones del algoritmo.
- Todo se compila con -O3.

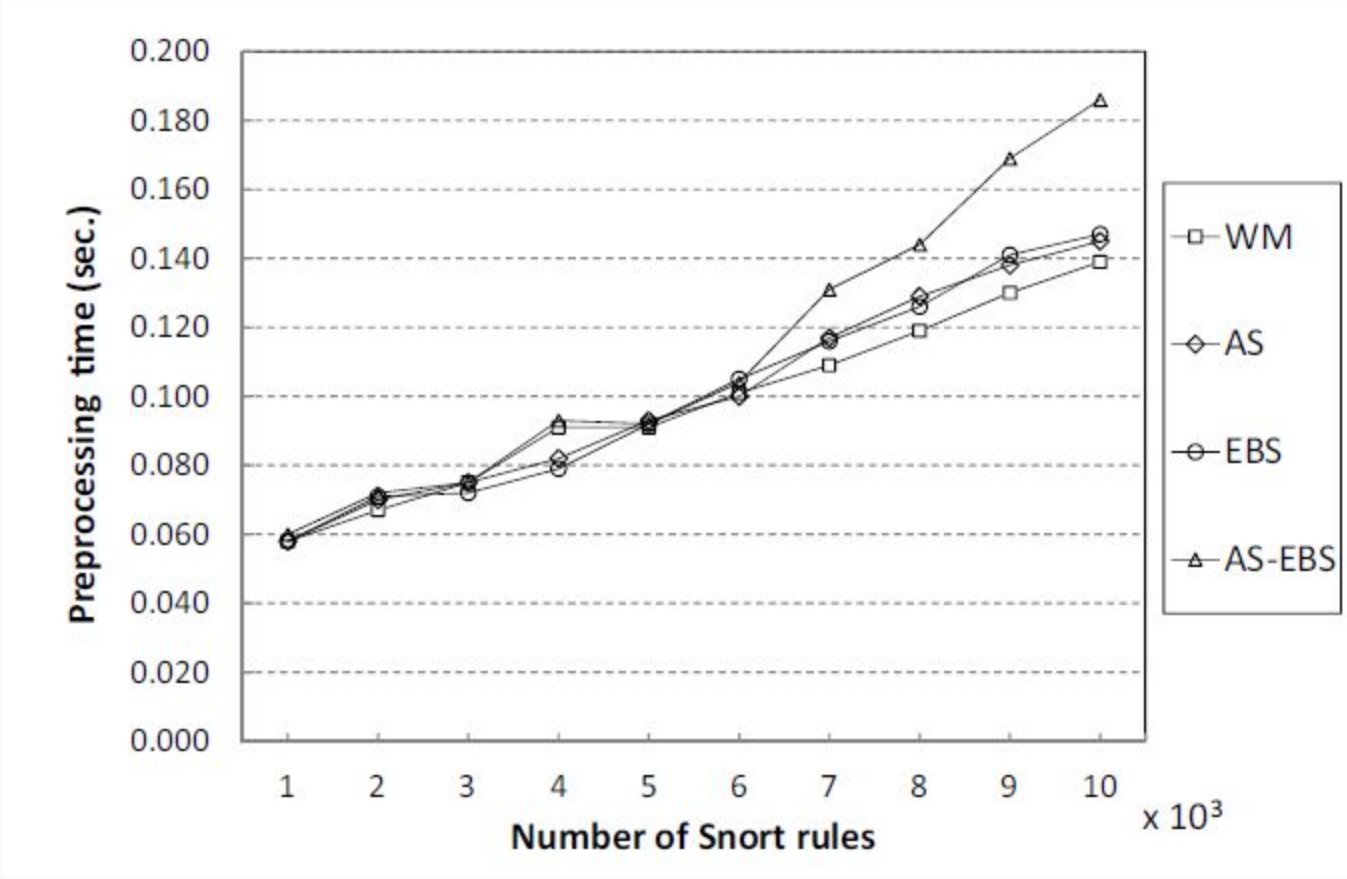


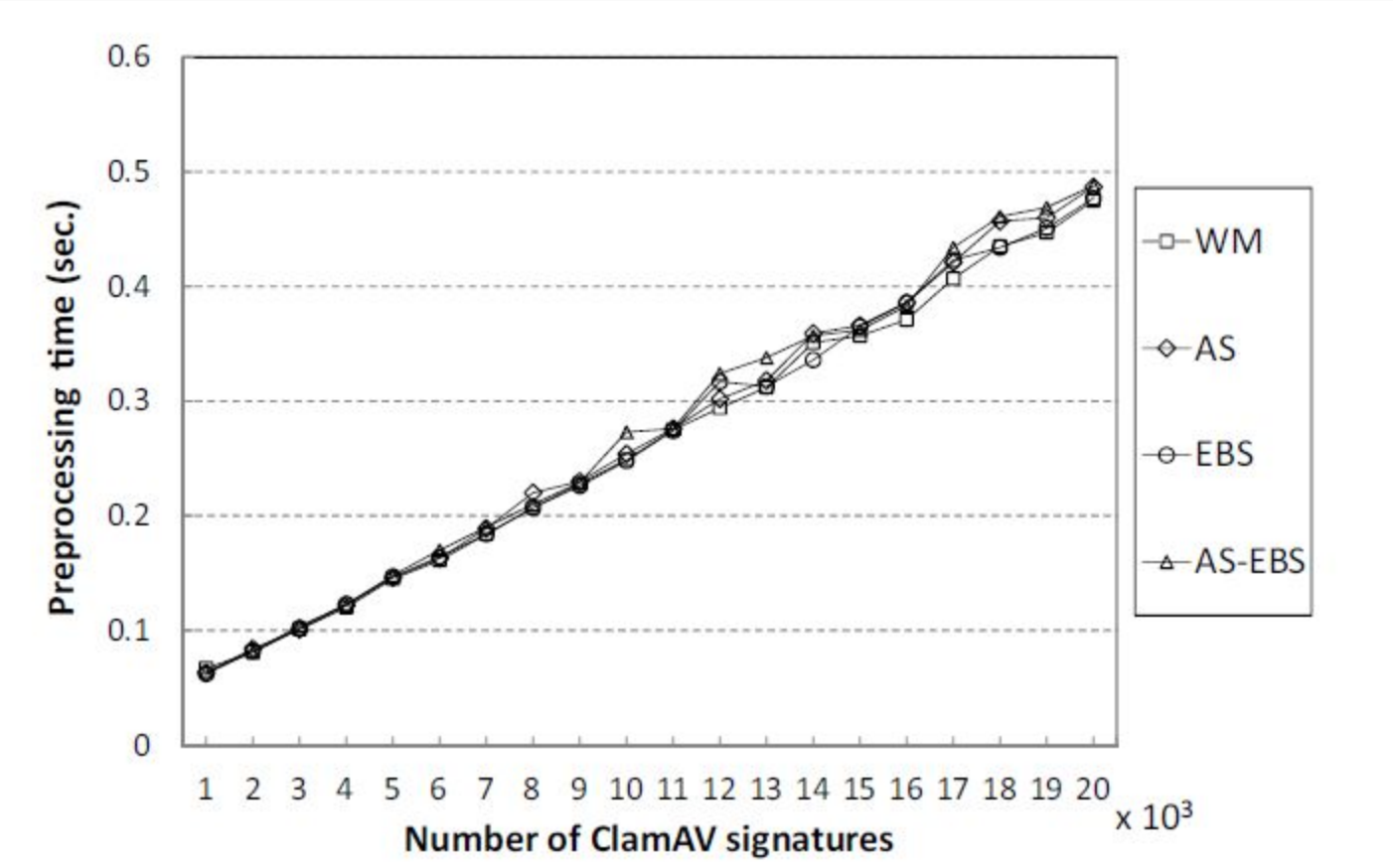


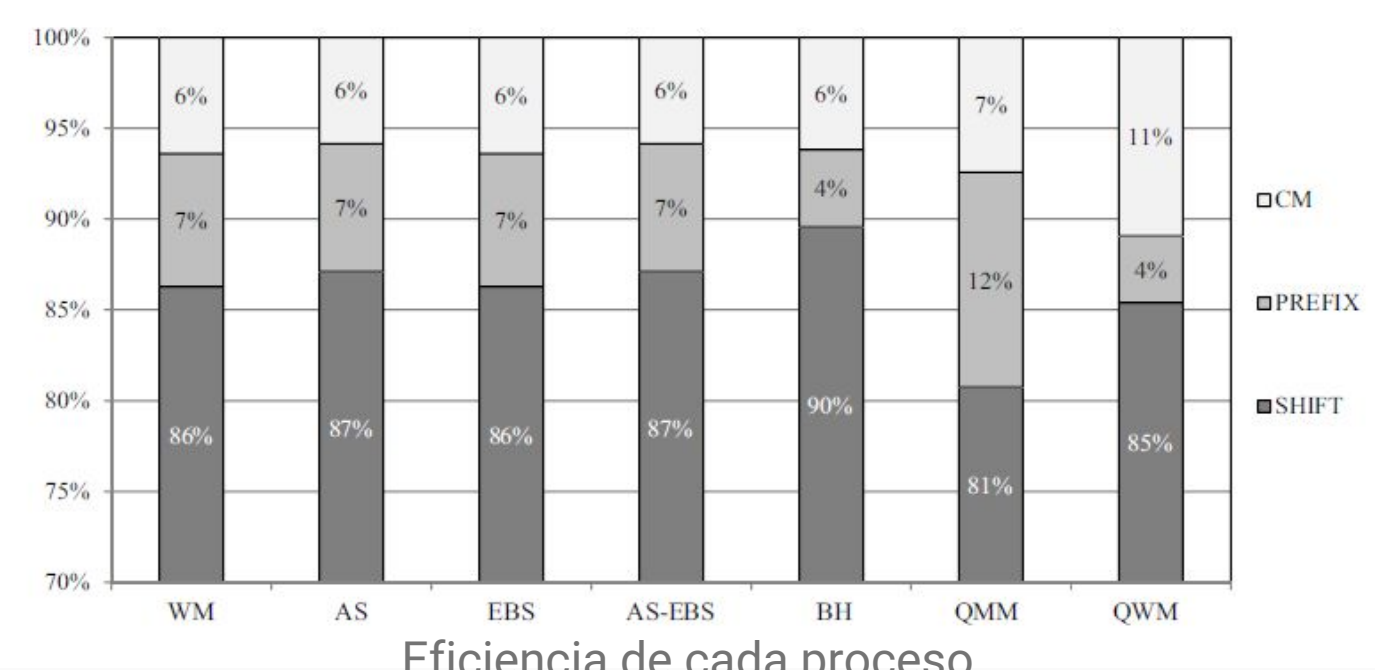
Pattern Set	$\|\Sigma\|$	Number	Shortest	Avg.	Longest
Snort	256	13,896	6	20	1021
ClamAV	256	20,000	16	204	694


90.00
80.00
70.00
60.00
50.00
40.00
30.00
20.00
10.00
0.00
Execution time (sec.)
1
2
3
4
5
6
7
8
9
10
x 10³
Number of Snort rules
WM
AS
EBS
AS-EBS
BH
QMM
QWM


25
20
15
10
5
0
Execution time (sec.)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
x 10³
Number of ClamAV signatures
WM
AS
EBS
AS-EBS
BH
QMM
QWM


0.200
0.180
0.160
0.140
0.120
0.100
0.080
0.060
0.040
0.020
0.000
Preprocessing time (sec.)
1
2
3
4
5
6
7
8
9
10
x 10³
Number of Snort rules
WM
AS
EBS
AS-EBS


0.6
0.5
0.4
0.3
0.2
0.1
0
Preprocessing time (sec.)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
x 10³
Number of ClamAV signatures
WM
AS
EBS
AS-EBS



Eficiencia de cada proceso

Concluyendo...

Gracias!