Deep Packet Inspection over Encrypted Traffic

Divyanshu, Romir, Mrittika

August 1, 2025

Privacy vs. Security

- **Encrypted traffic** hides data from eavesdroppers and also from security tools.
- **DPI tools** like IDS, parental filters, and DLP rely on inspecting **unencrypted** packet contents.

Privacy vs. Security

- **Encrypted traffic** hides data from eavesdroppers and also from security tools.
- **DPI tools** like IDS, parental filters, and DLP rely on inspecting **unencrypted** packet contents.
- With encryption, these tools can no longer operate, even on our own networks, violating security.
- Existing solutions violate privacy.

Privacy vs. Security

- **Encrypted traffic** hides data from eavesdroppers and also from security tools.
- **DPI tools** like IDS, parental filters, and DLP rely on inspecting **unencrypted** packet contents.
- With encryption, these tools can no longer operate, even on our own networks, violating security.
- Existing solutions violate privacy.

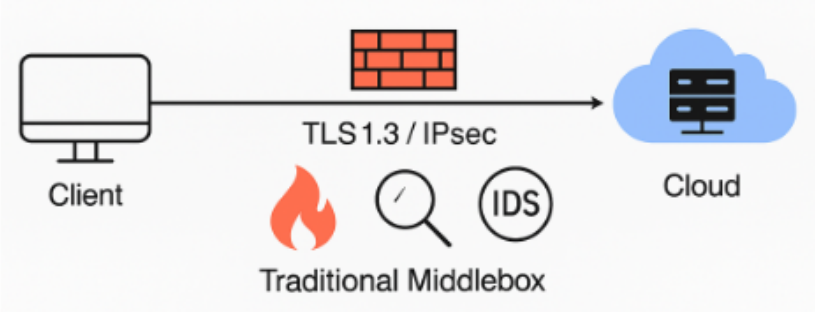
How to achieve both privacy and security?

Motivation

- Most enterprise networks use HTTPS connection to transfer all internal traffic.

Motivation

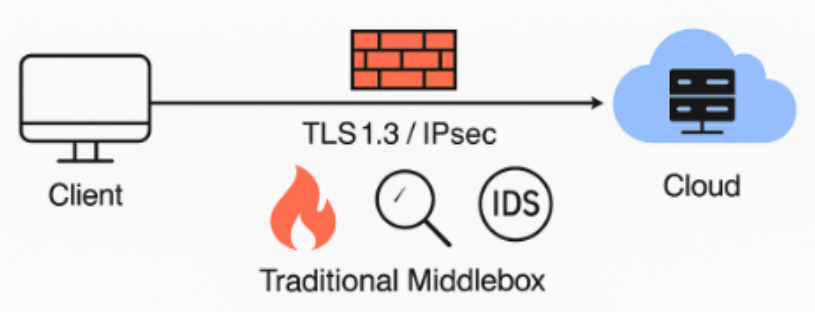
- Most enterprise networks use HTTPS connection to transfer all internal traffic.



- **Challenge:** Traditional middleboxes (firewalls, IDS/IPS) cannot inspect such encrypted payloads sent over a secure HTTPS/TLS connection.

Motivation

- Most enterprise networks use HTTPS connection to transfer all internal traffic.



- **Challenge:** Traditional middleboxes (firewalls, IDS/IPS) cannot inspect such encrypted payloads sent over a secure HTTPS/TLS connection.
- **Solution:** Inspect encrypted traffic **without decrypting it**, preserving both ***security*** and ***user privacy***.

Objective of the internship

- To design an efficient DPI system where the middlebox only has access to the encrypted traffic.

Objective of the internship

- To design an efficient DPI system where the middlebox only has access to the encrypted traffic.
- Such a system enables **selective traffic inspection** over encrypted traffic using **searchable encryption.**
- Searchable Encryption(SE) supports search with keywords over encrypted traffic.

What is DPI?

- Deep Packet Inspection (DPI) is a network traffic analysis technique that inspects the payload flowing through a network.
- DPI is performed by network middleboxes.
- It provides a wide range of services including IDS,IPS ,etc.

What is DPI?

- Deep Packet Inspection (DPI) is a network traffic analysis technique that inspects the payload flowing through a network.
- DPI is performed by network middleboxes.
- It provides a wide range of services including IDS,IPS ,etc.
- **DPI over encrypted traffic** enables deep packet inspection over an encrypted payload.

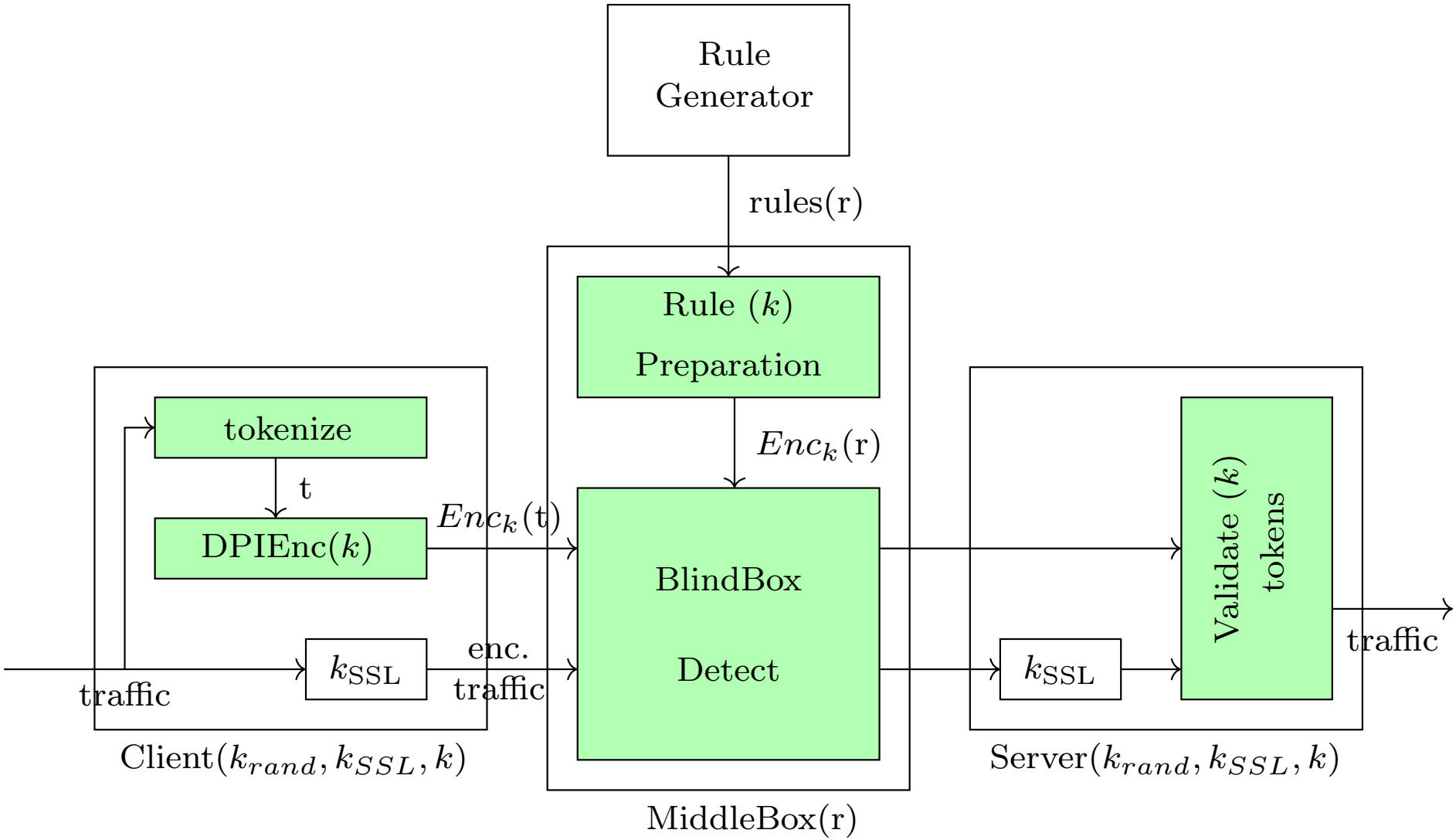
Acknowledgment

This presentation is primarily based on the work:

Sherry et al., *BlindBox: Deep Packet Inspection over Encrypted Traffic*, ACM, 2015.

All technical details and notations closely follow the above.

System Architecture



Protocol I

Protocol I enables matching a suspicious keyword against the encrypted traffic. An attack rule in this approach only consists of one keyword.

Protocol I

Protocol I enables matching a suspicious keyword against the encrypted traffic. An attack rule in this approach only consists of one keyword.

1. Tokenization
2. DPIEnc encryption scheme
3. BlindBox Detect
4. Rule Preparation
5. Validate Tokens

Tokenization

Two methods of tokenization can be used.

Tokenization

Two methods of tokenization can be used.

→ Window-based tokenization

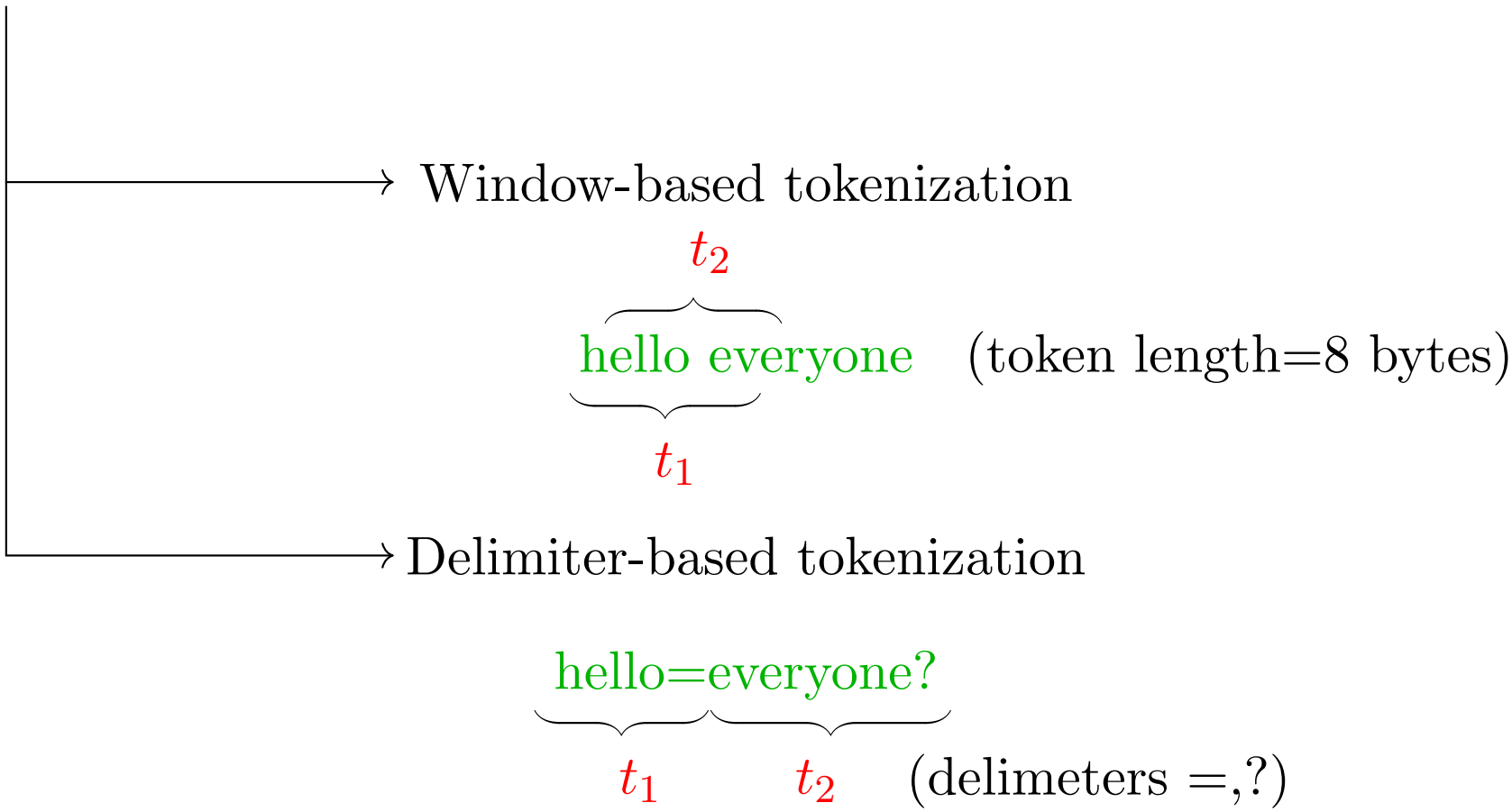
t_2

hello everyone (token length=8 bytes)

t_1

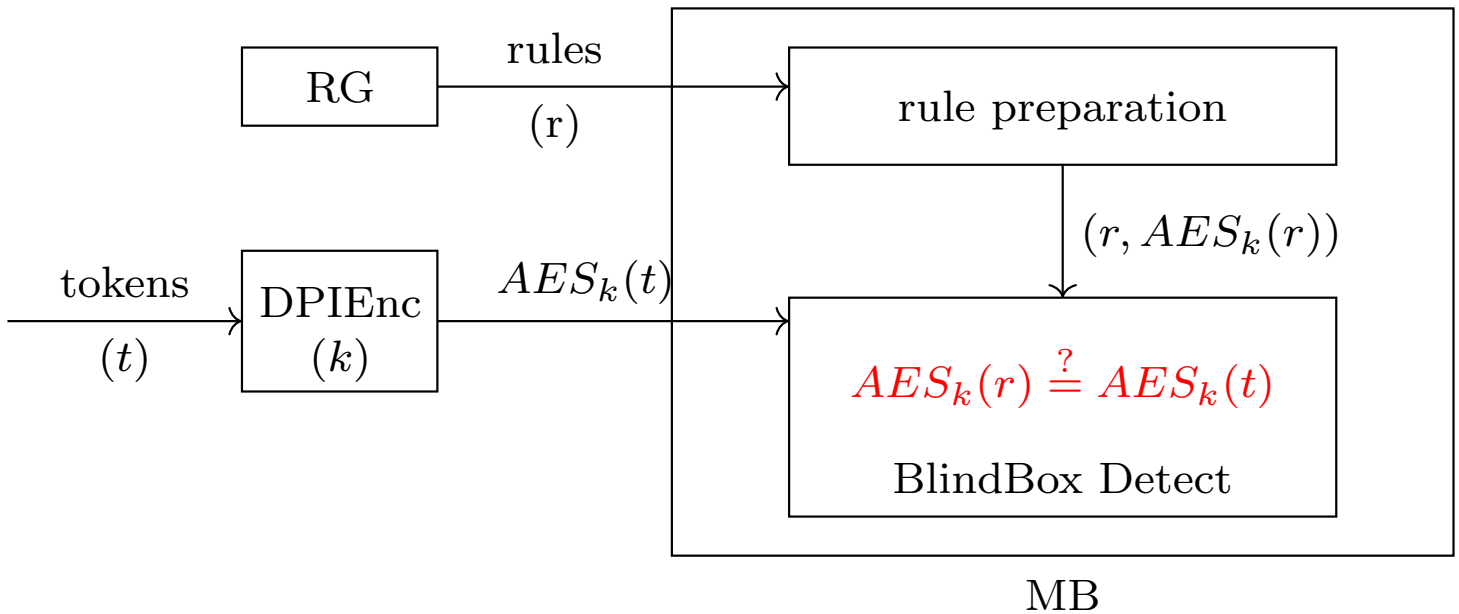
Tokenization

Two methods of tokenization can be used.



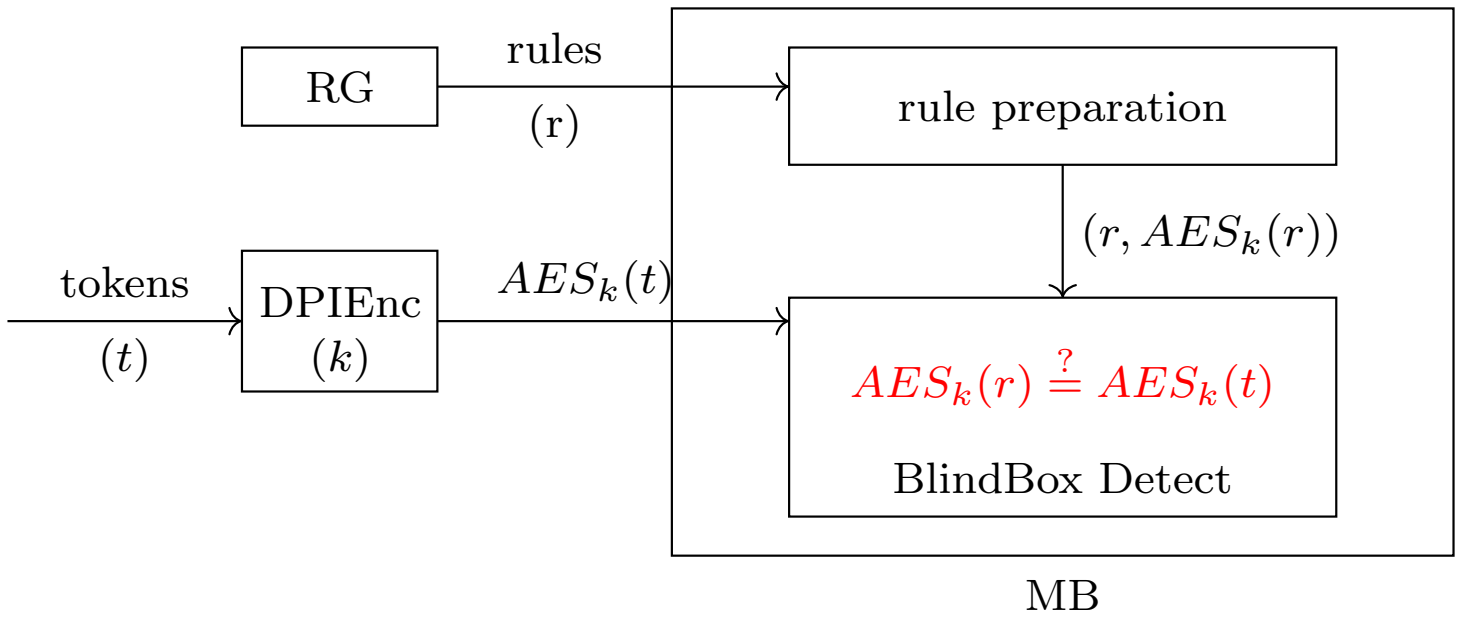
DPIEnc encryption scheme

For now , lets assume that MB has the key k, so it computes the encrypted rule $AES_k(r)$ for each rule r.



DPIEnc encryption scheme

For now , lets assume that MB has the key k, so it computes the encrypted rule $AES_k(r)$ for each rule r.



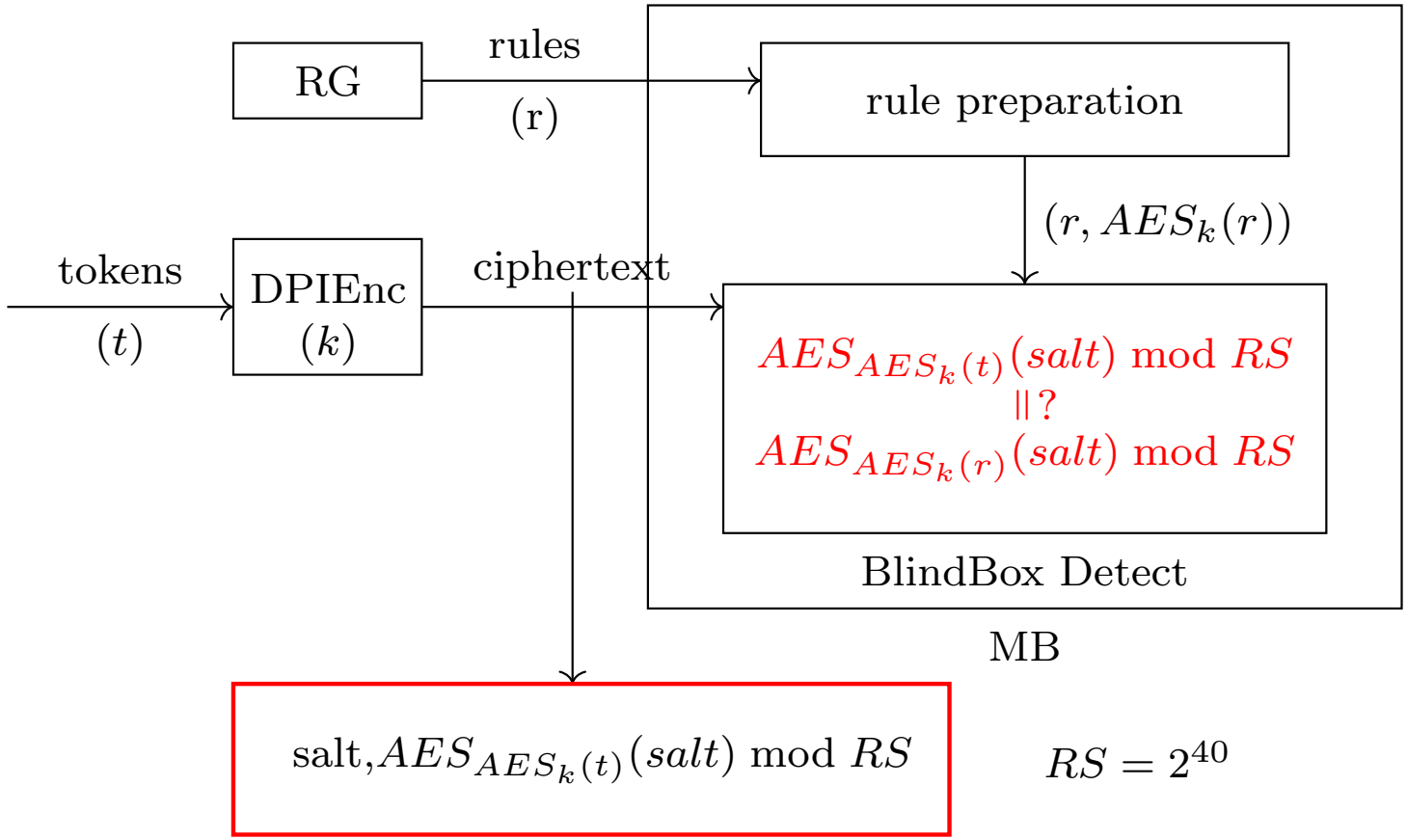
WEAK SECURITY!!

DPIEnc encryption scheme

$AES_{AES_k(t)}(salt)$ for some random $salt$.

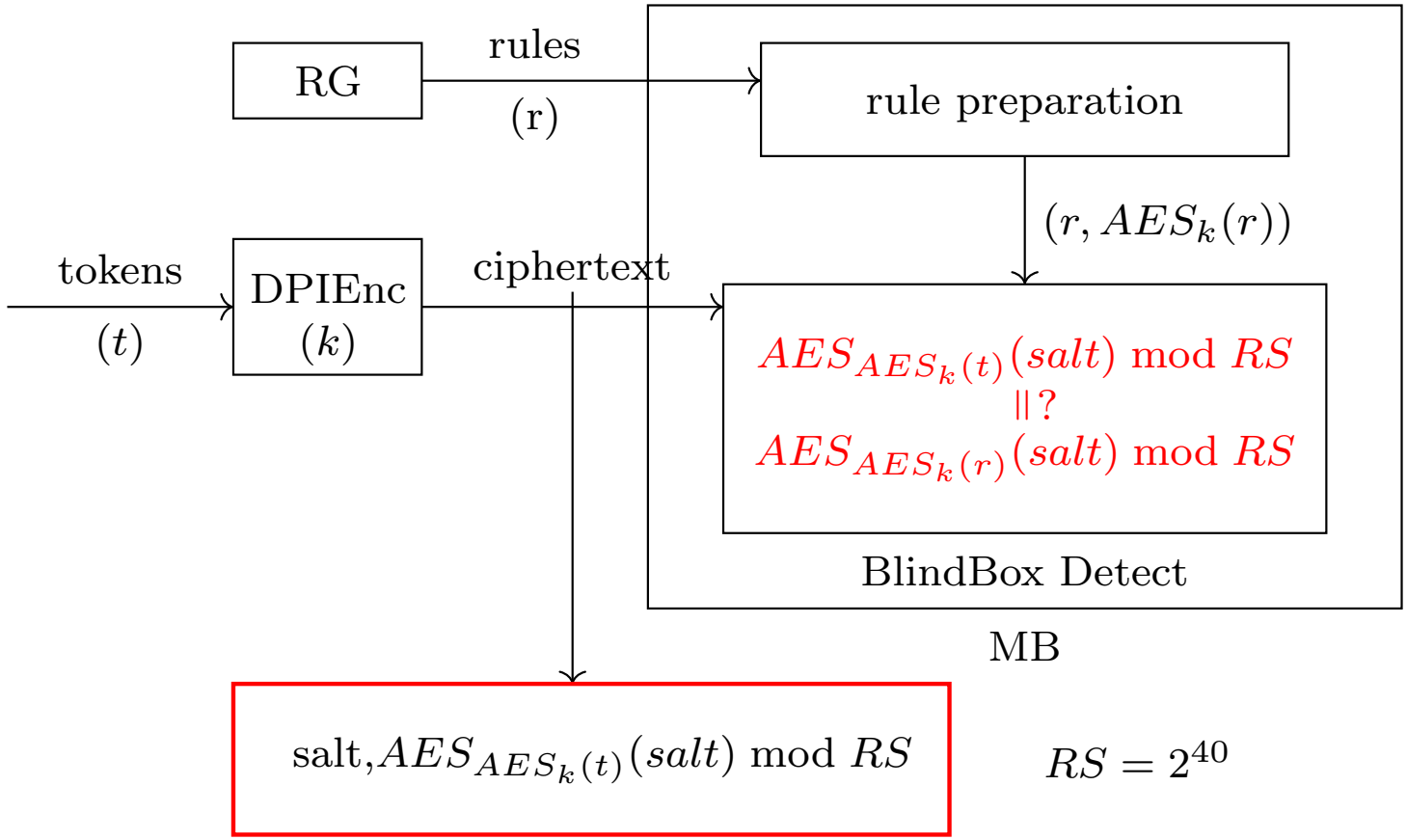
DPIEnc encryption scheme

$AES_{AES_k(t)}(salt)$ for some random $salt$.



DPIEnc encryption scheme

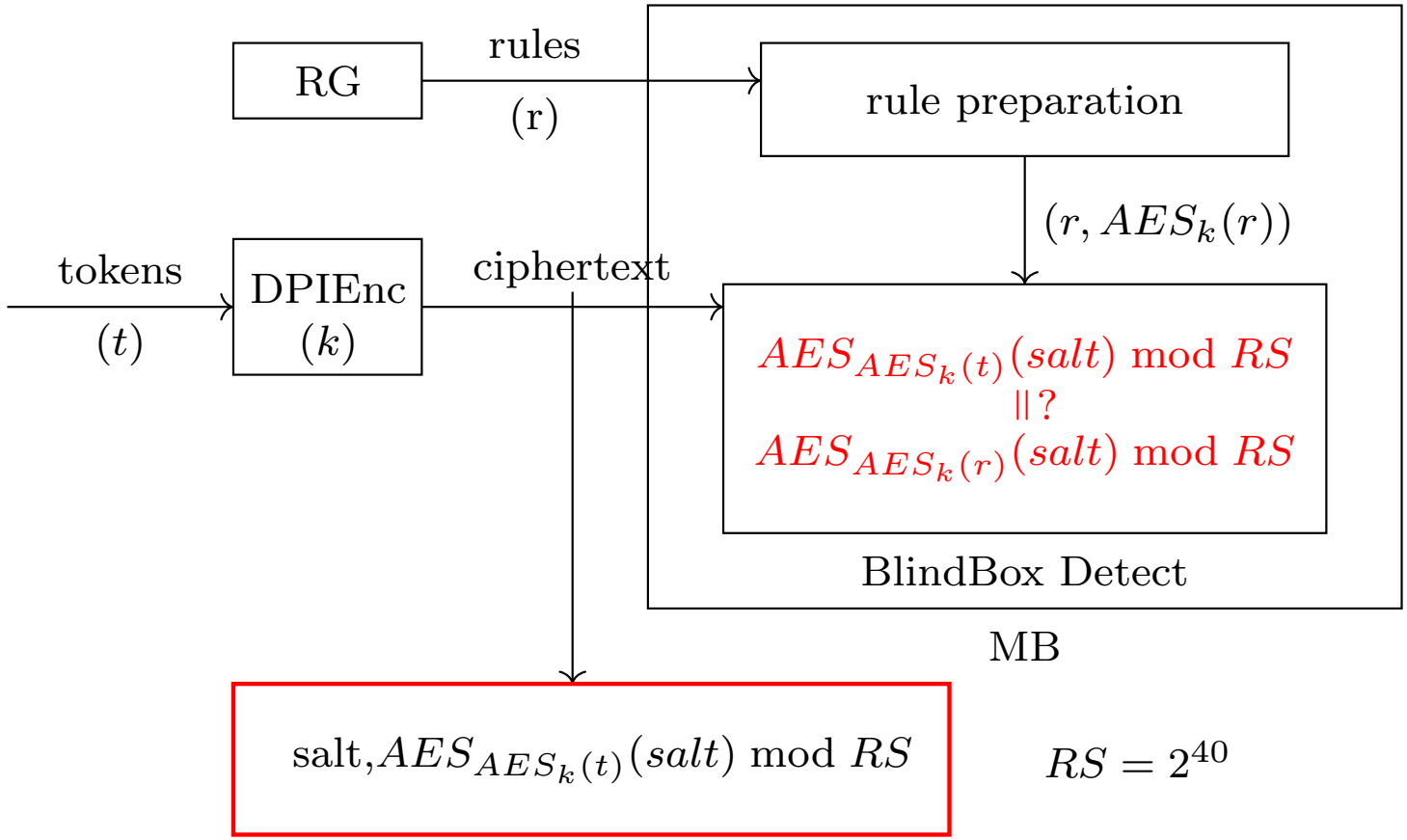
$AES_{AES_k(t)}(salt)$ for some random $salt$.



Performance per token linear in the number of rules.

DPIEnc encryption scheme

$AES_{AES_k(t)}(salt)$ for some random $salt$.



Performance per token linear in the number of rules.

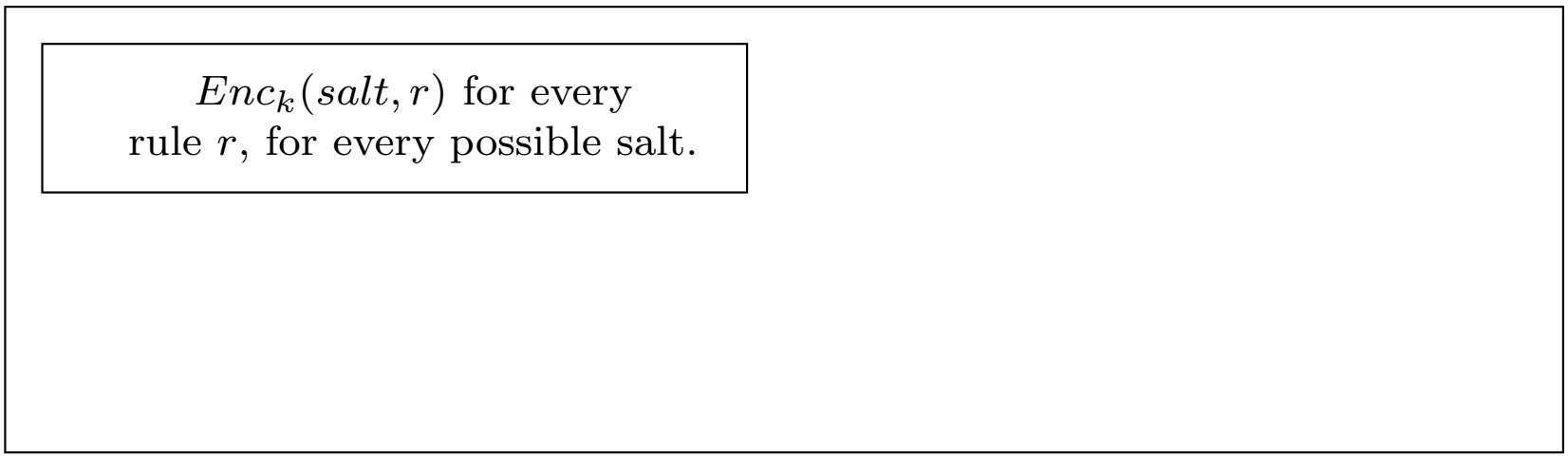
TOO SLOW!!

BlindBox Detect

Achieves logarithmic lookup times. HOW?

BlindBox Detect

Achieves logarithmic lookup times. HOW?
Arranges the encrypted rules in a search tree.

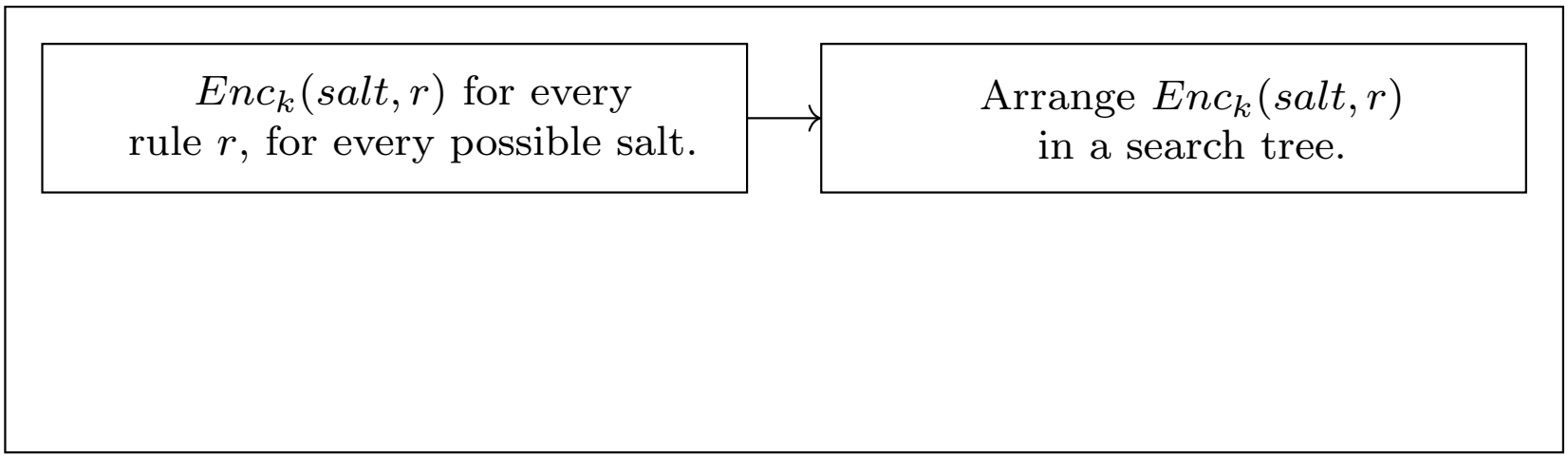


Need to compute $Enc_k(salt, t)$ for every posible salt !!

INFEASIBLE!!

BlindBox Detect

Achieves logarithmic lookup times. HOW?
Arranges the encrypted rules in a search tree.

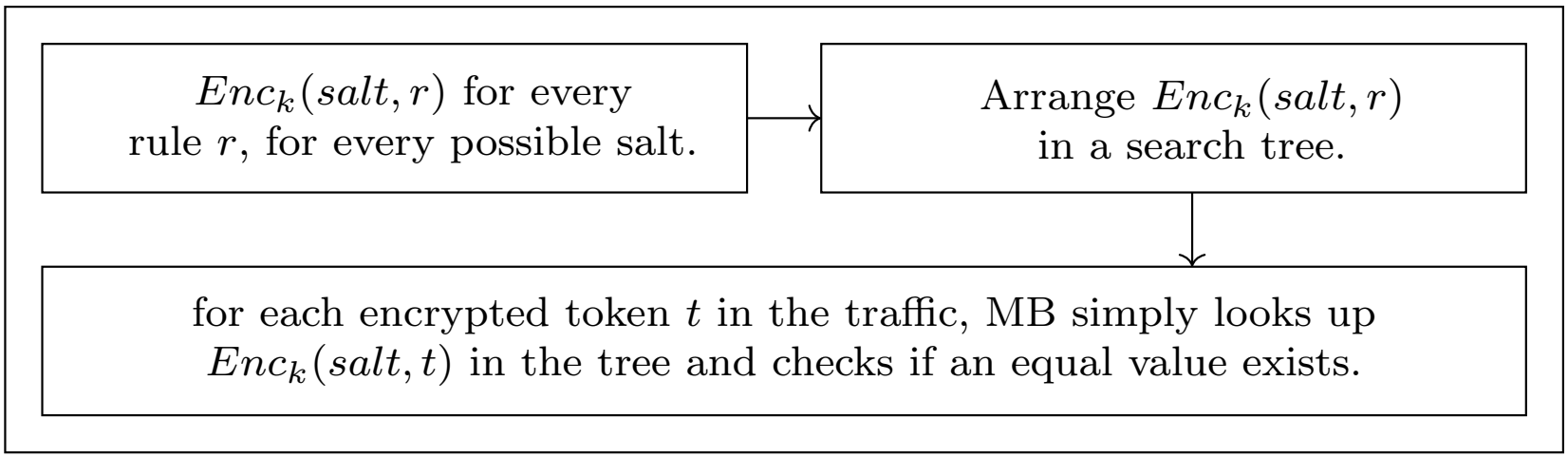


Need to compute $Enc_k(salt, t)$ for every posible salt !!

INFEASIBLE!!

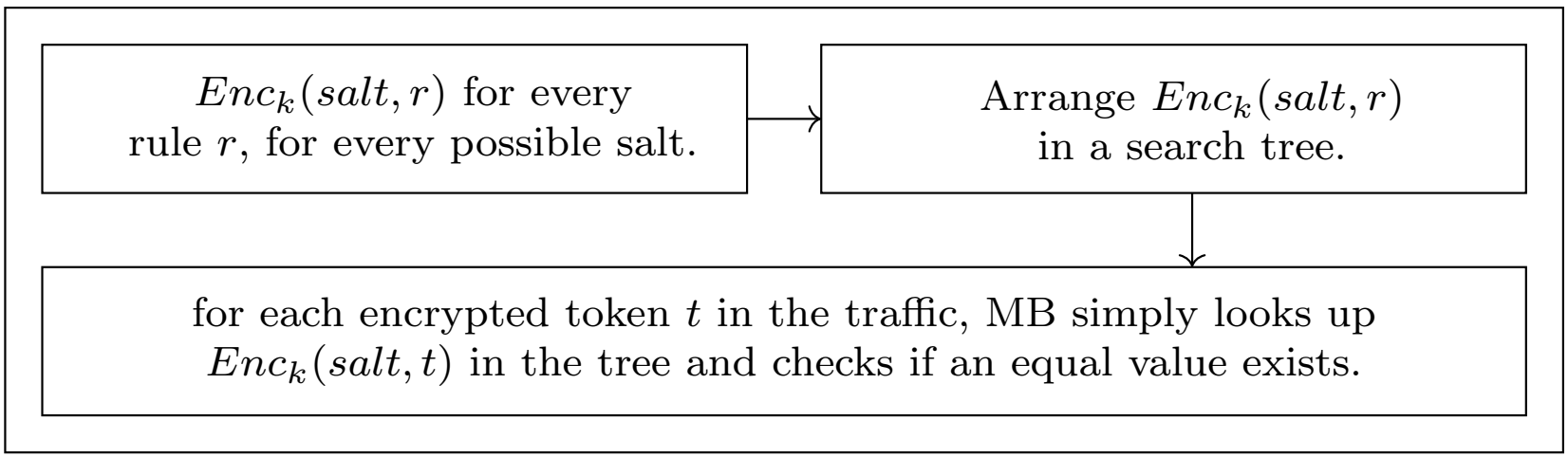
BlindBox Detect

Achieves logarithmic lookup times. HOW?
Arranges the encrypted rules in a search tree.



BlindBox Detect

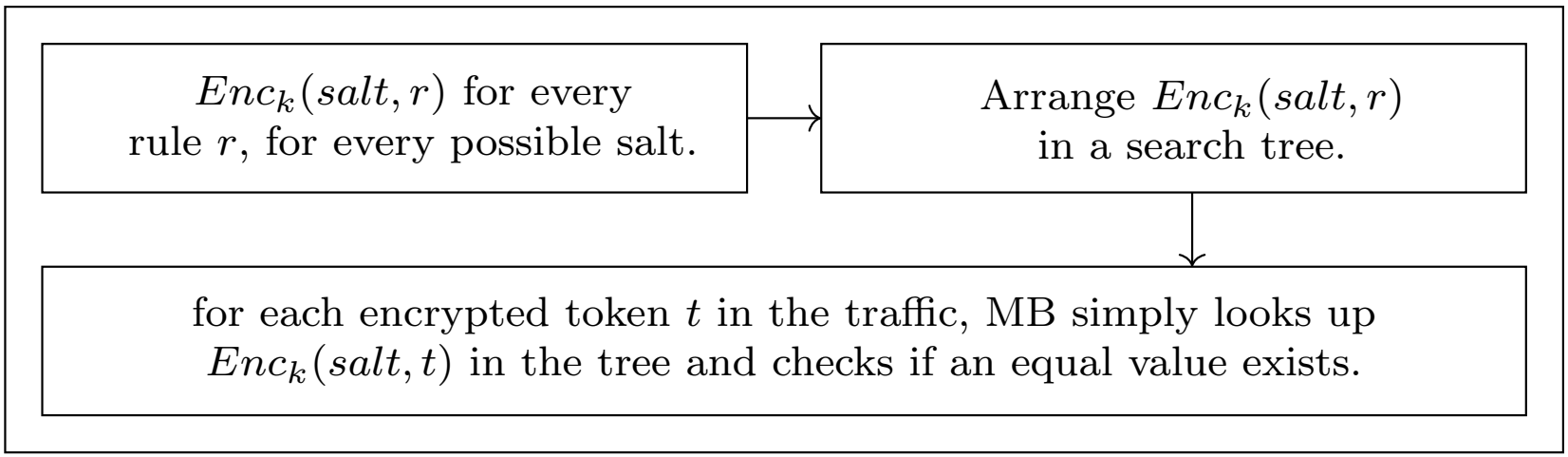
Achieves logarithmic lookup times. HOW?
Arranges the encrypted rules in a search tree.



Need to compute $Enc_k(salt, t)$ for every posible salt !!

BlindBox Detect

Achieves logarithmic lookup times. HOW?
Arranges the encrypted rules in a search tree.



Need to compute $Enc_k(salt, t)$ for every posible salt !!

INFEASIBLE!!

BlindBox Detect

What if we use the same salt for every token t?

BlindBox Detect

What if we use the same salt for every token t?

Problem. A salt cannot be reused for the same token.

Q. Can the same salt repeat across different tokens.

BlindBox Detect

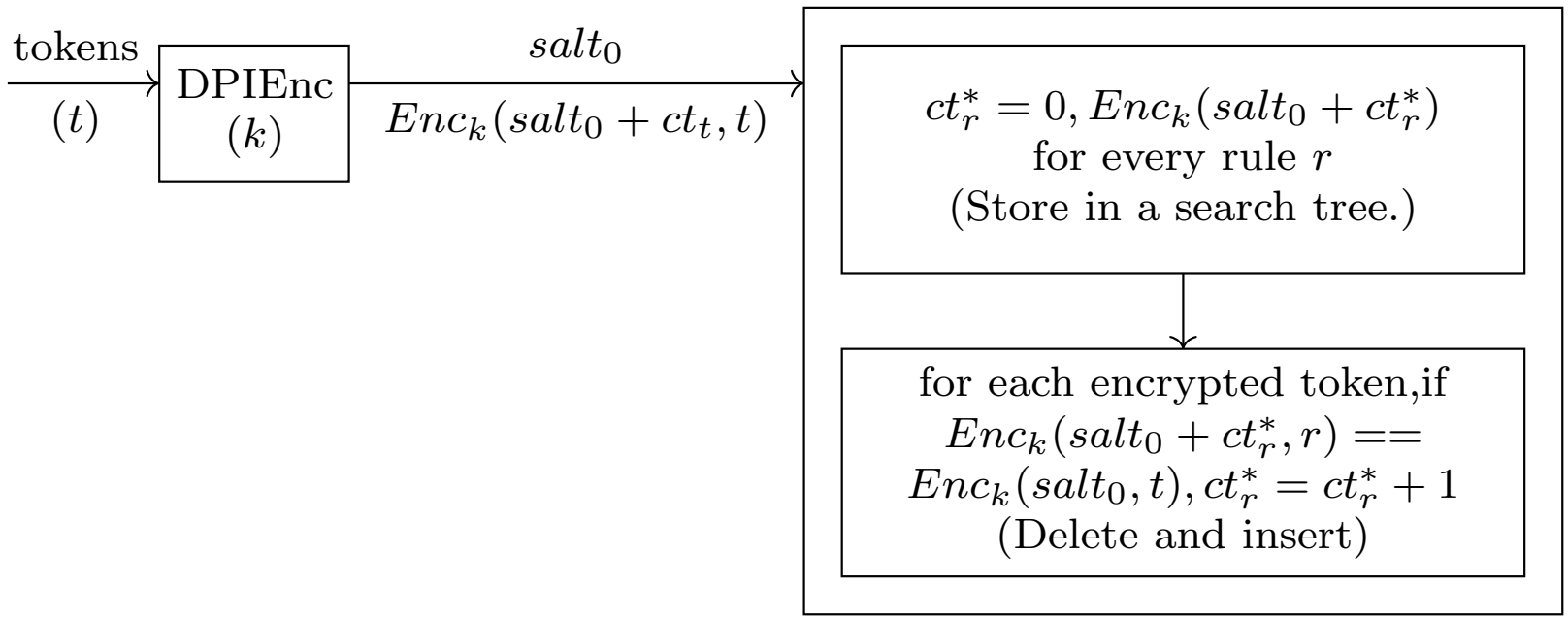
What if we use the same salt for every token t?

Problem. A salt cannot be reused for the same token.

Q. Can the same salt repeat across different tokens. YES.

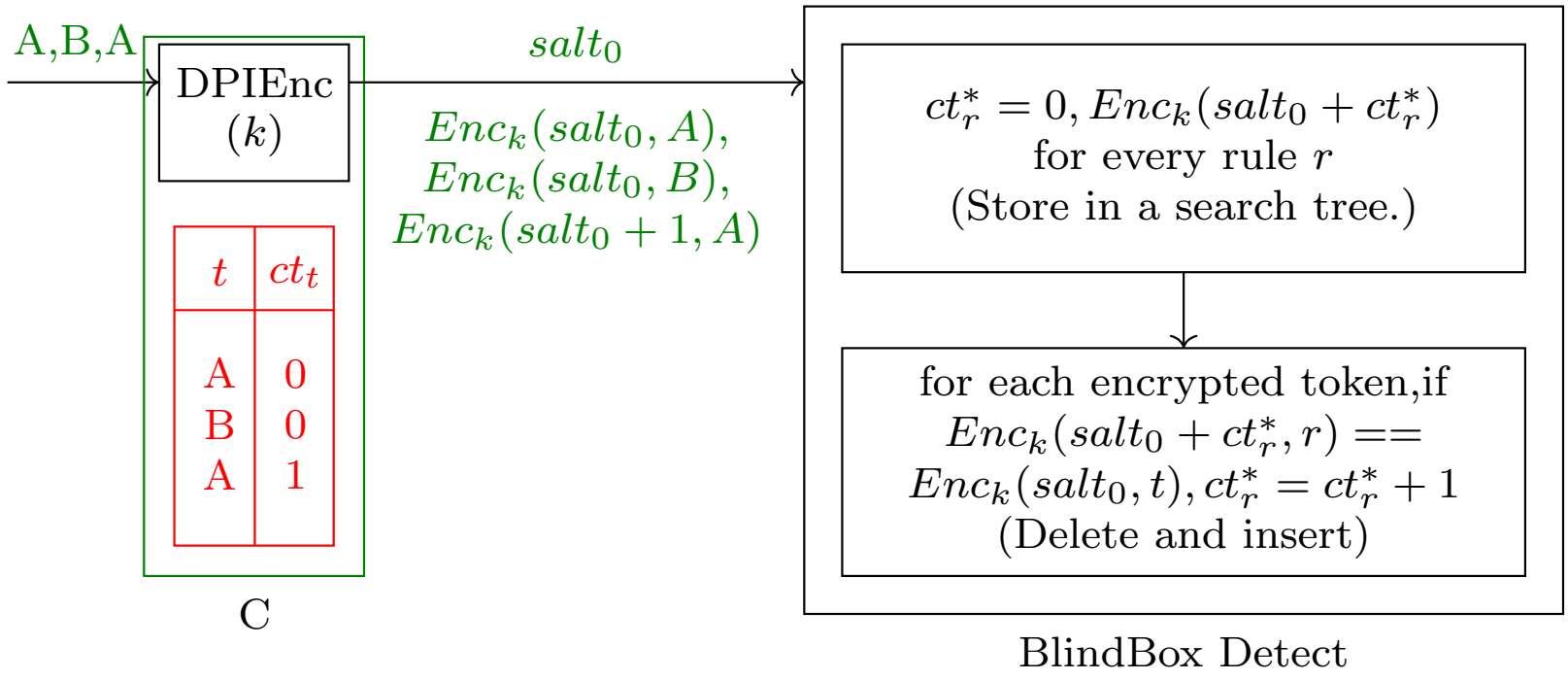
BlindBox Detect

What if we use the same salt for every token t?
Problem. A salt cannot be reused for the same token.
Q. Can the same salt repeat across different tokens. YES.

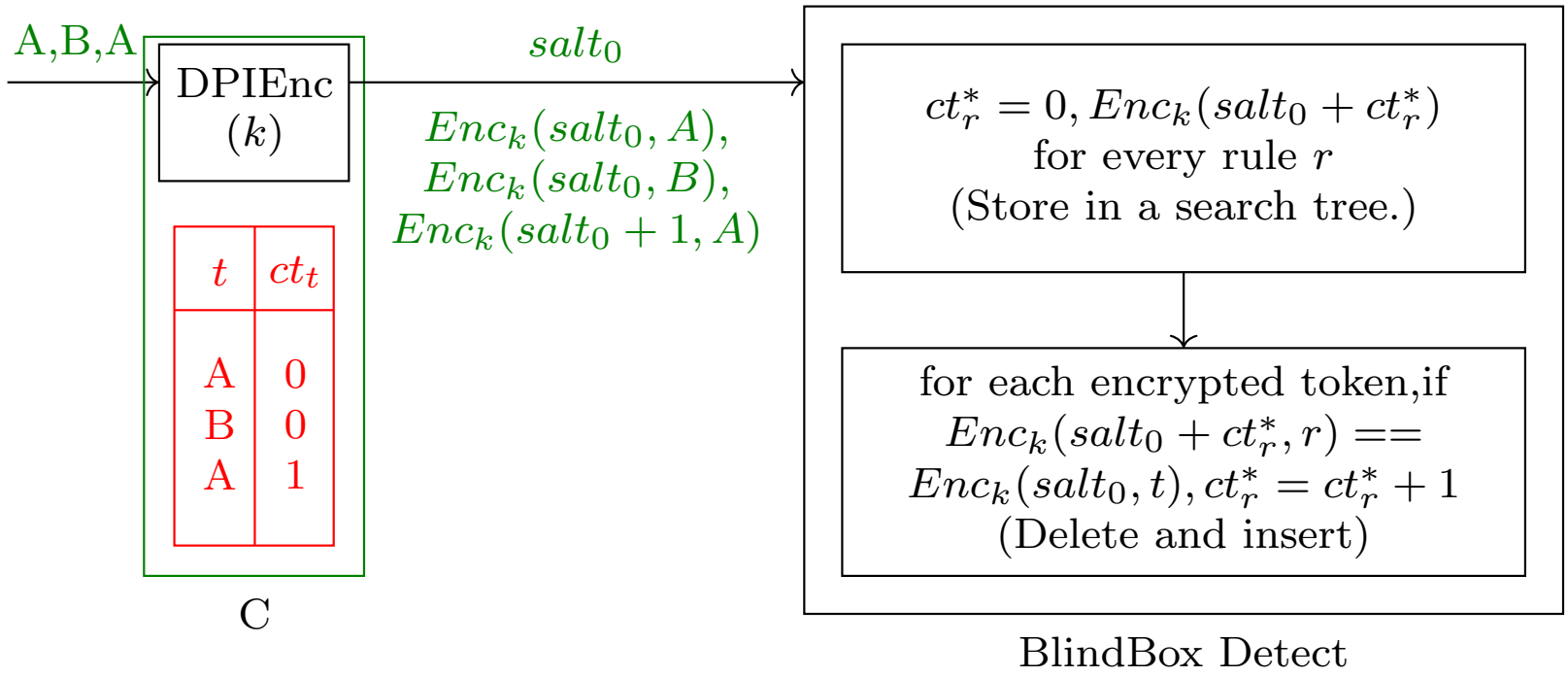


BlindBox Detect

BlindBox Detect



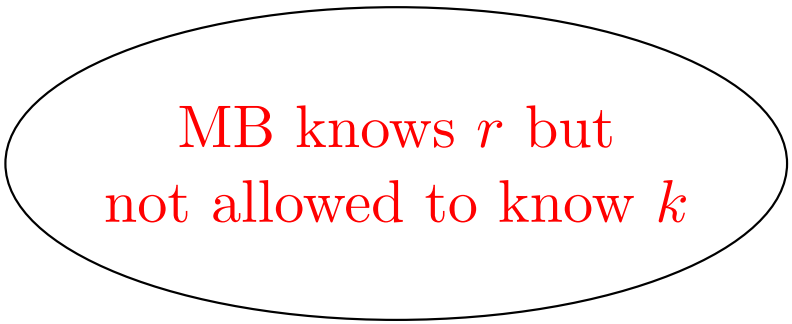
BlindBox Detect



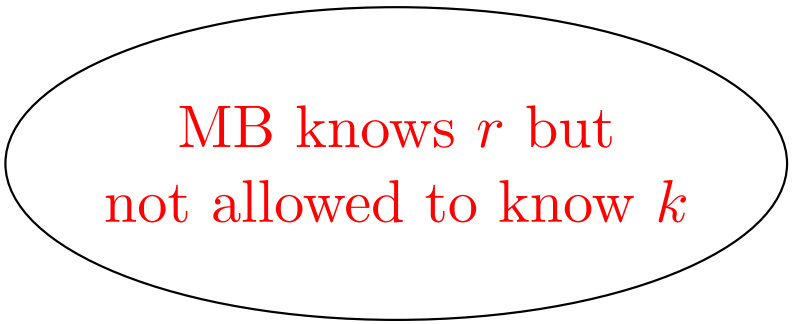
For every token t, MB performs a simple tree lookup, which is logarithmic in the number of rules.

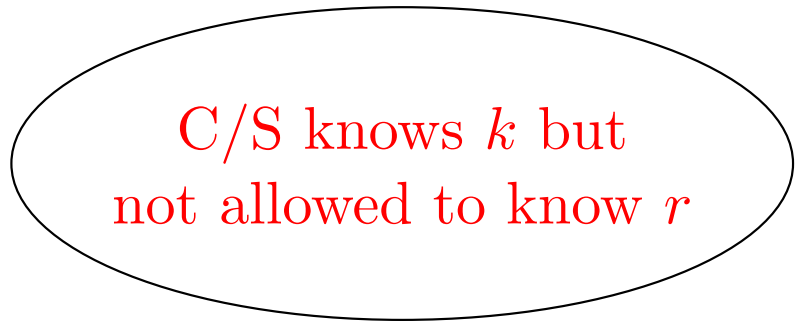
Rule Preparation

Rule Preparation



Rule Preparation





Rule Preparation

MB knows r but
not allowed to know k

C/S knows k but
not allowed to know r

How does MB compute $AES_k(r)$ for every rules r !!

Rule Preparation

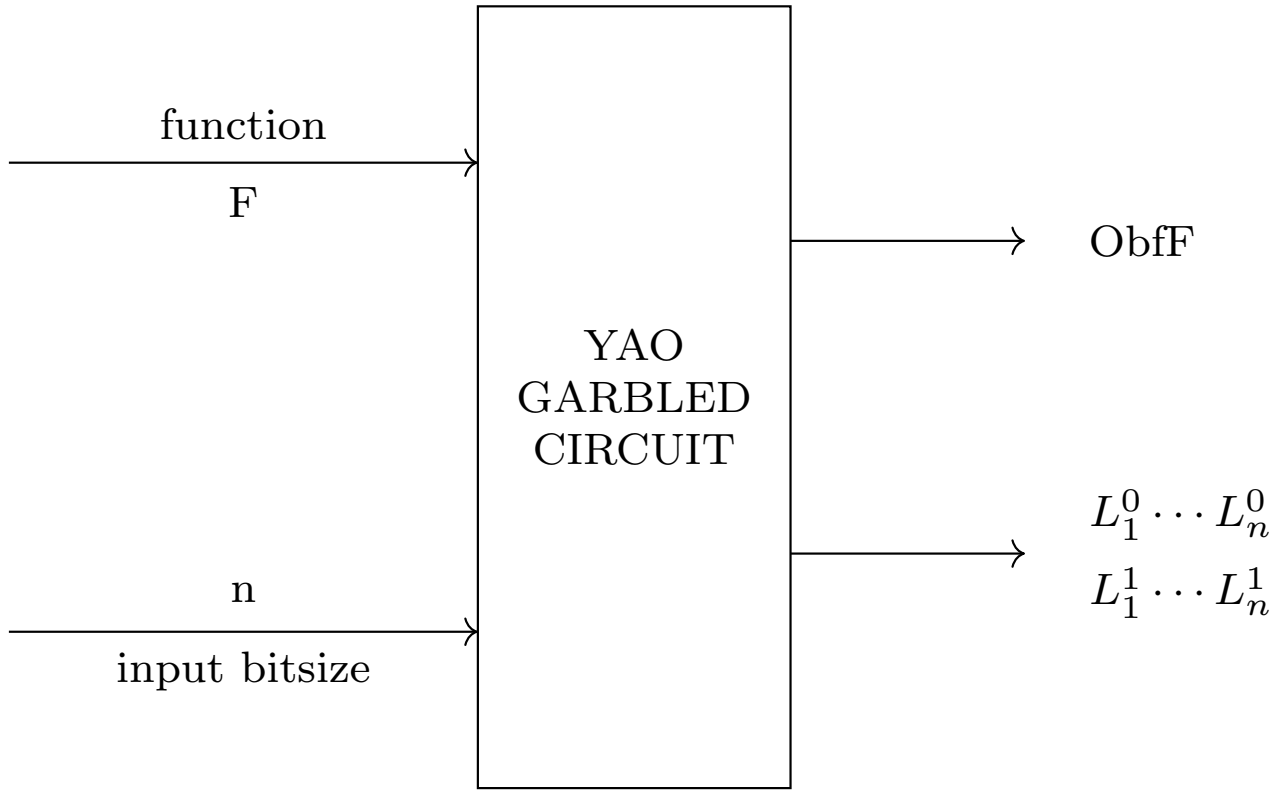
MB knows r but
not allowed to know k

C/S knows k but
not allowed to know r

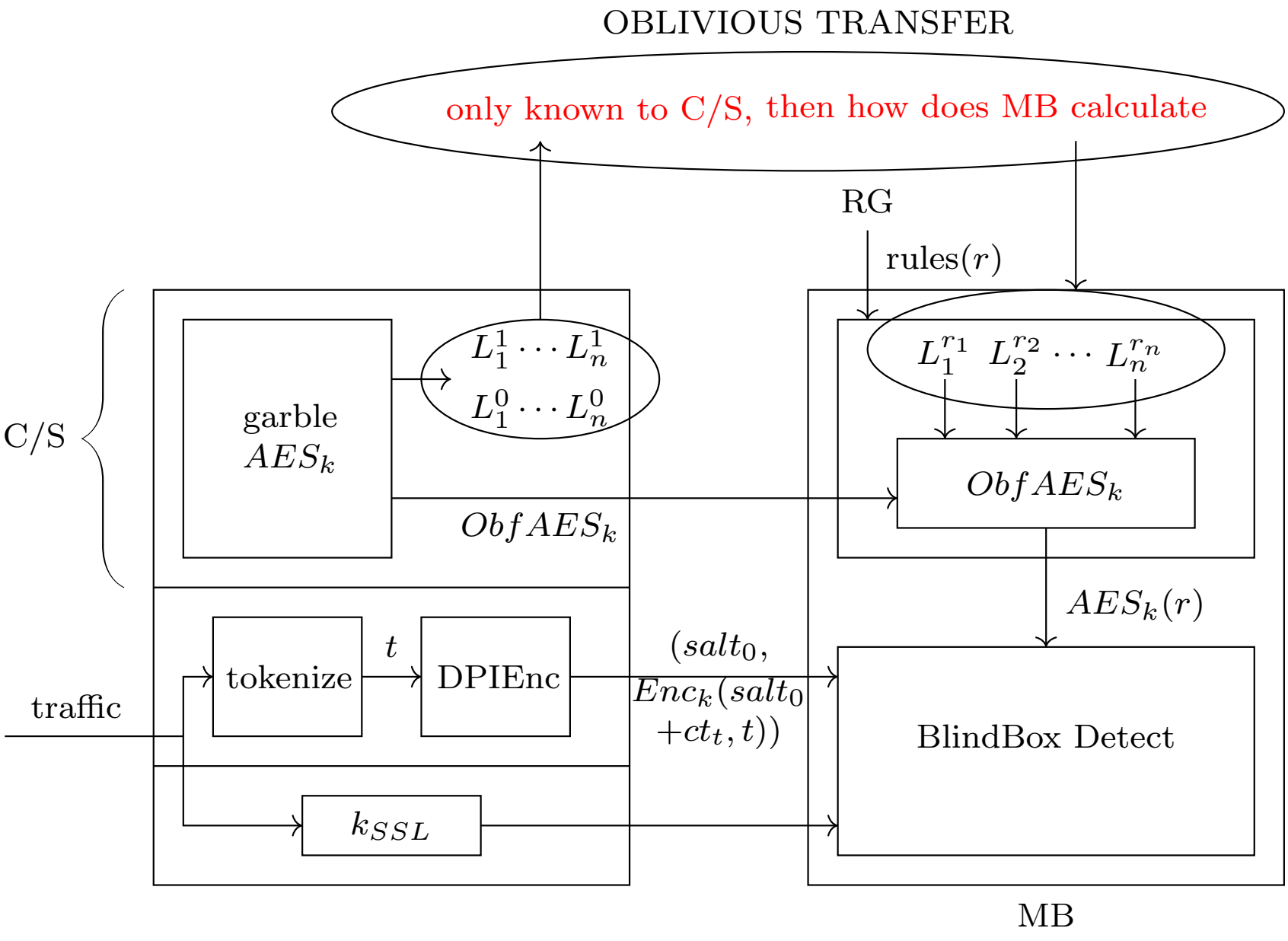
How does MB compute $AES_k(r)$ for every rules r !!

The sender provides the MB with an obfuscation of the function AES with key k hardcoded in it.

Obfuscated Rule Encryption

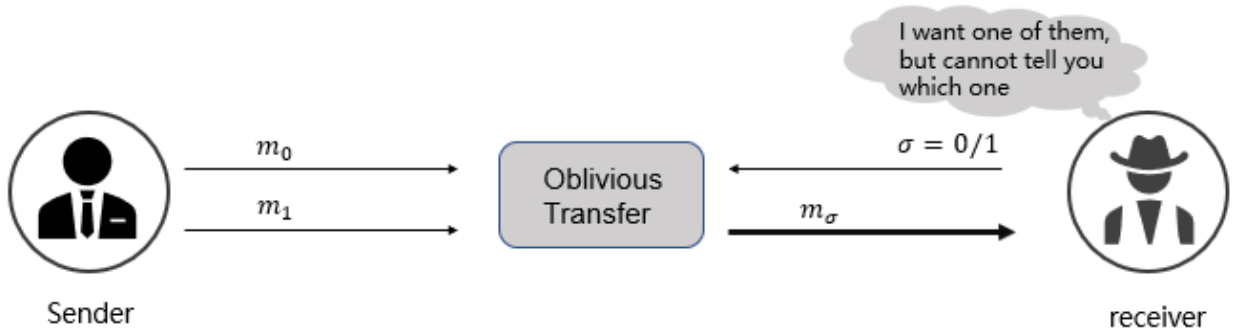


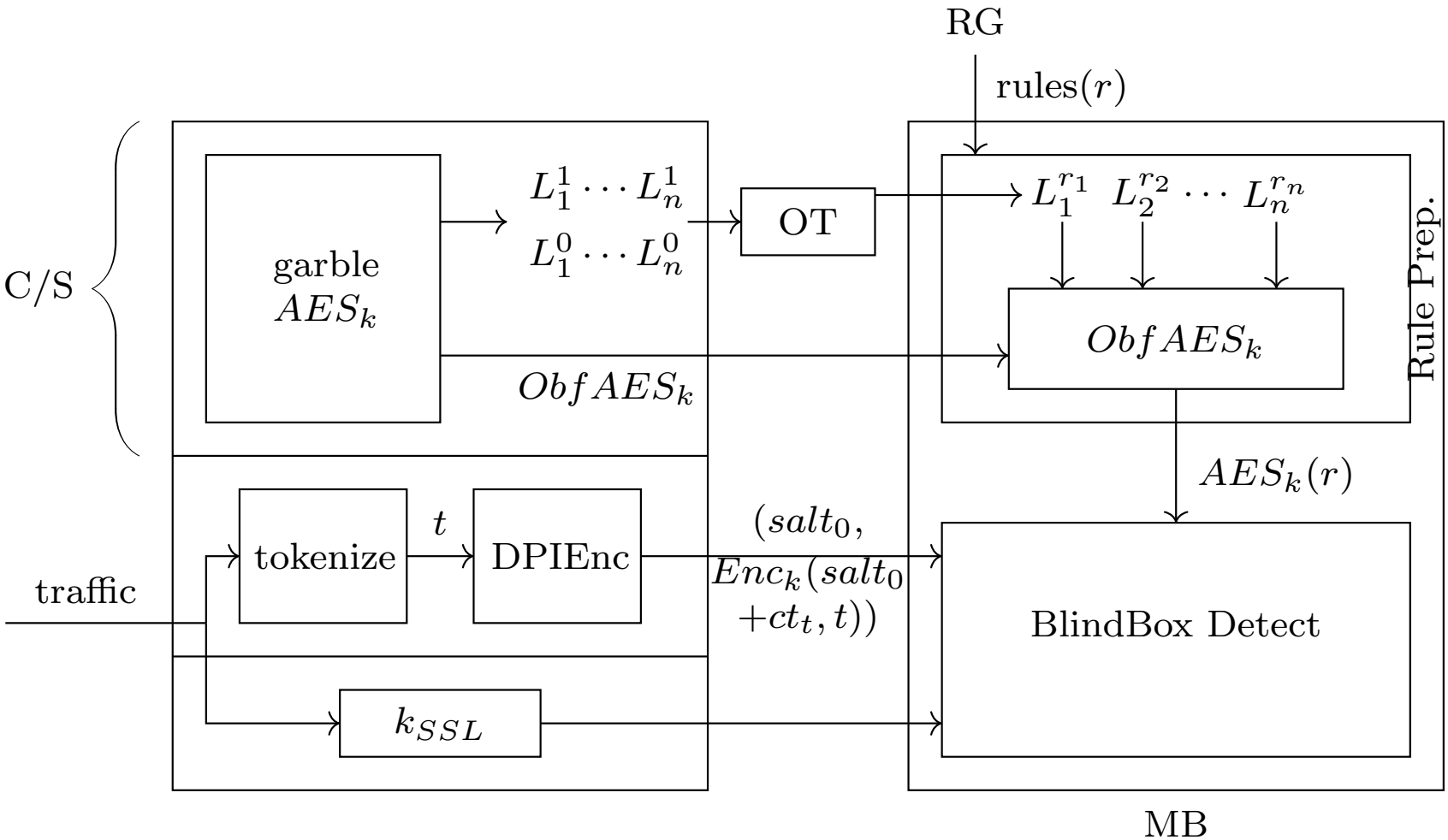
$\text{ObfF}(L_1^{x_1}, L_2^{x_2}, \ldots, L_n^{x_n})=\text{F(x)}$, $x = x_1 x_2 \ldots x_n$ (bits)


OBLIVIOUS TRANSFER
only known to C/S, then how does MB calculate
RG
rules(r)
C/S
garble
AES_k
$L_1^1 \cdots L_n^1$
$L_1^0 \cdots L_n^0$
$ObfAES_k$
$L_1^{r_1}$ $L_2^{r_2}$ $\cdots$ $L_n^{r_n}$
$ObfAES_k$
$AES_k(r)$
traffic
tokenize
t
DPIEnc
$(salt_0,$
$Enc_k(salt_0$
$+ct_t, t))$
BlindBox Detect
k_{SSL}
MB

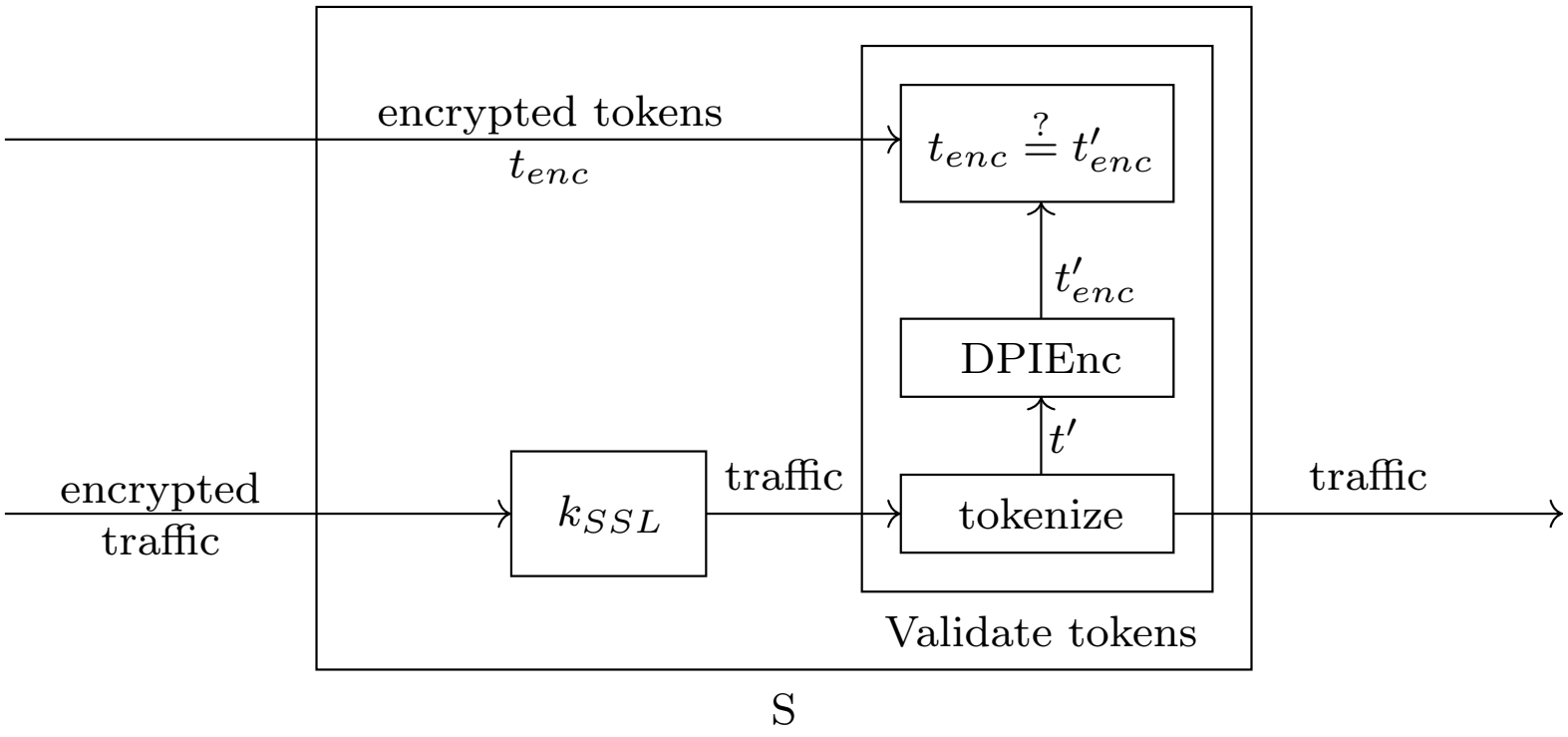
Oblivious Transfer

Oblivious transfer: B chooses $b \in 0, 1$. A has L^0, L^1. B can obtain L^b without learning L^{1-b} and A does not learn b.




RG
rules(r)
C/S
garble
AES_k
$L_1^1 \cdots L_n^1$
$L_1^0 \cdots L_n^0$
OT
$L_1^{r_1}$ $L_2^{r_2}$ $\cdots$ $L_n^{r_n}$
$ObfAES_k$
$ObfAES_k$
Rule Prep.
$AES_k(r)$
traffic
tokenize
t
DPIEnc
$(salt_0,$
$Enc_k(salt_0$
$+ct_t, t))$
BlindBox Detect
k_{SSL}
MB

Validate Tokens



Checks malicious endpoints.

Protocol II

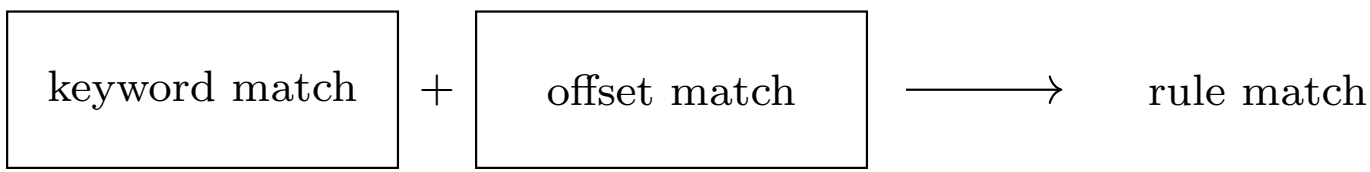
- Protocol II supports a limited form of an IDS.
- It allows a rule to contain :
  1. multiple keywords to be matched in the traffic
  2. absolute and relative offset information within the packet

Protocol II

- Protocol II supports a limited form of an IDS.
- It allows a rule to contain :
  1. multiple keywords to be matched in the traffic
  2. absolute and relative offset information within the packet
- Builds on Protocol I with one exception.
- The client attaches to each encrypted token the offset in the stream where it appeared.

Protocol II

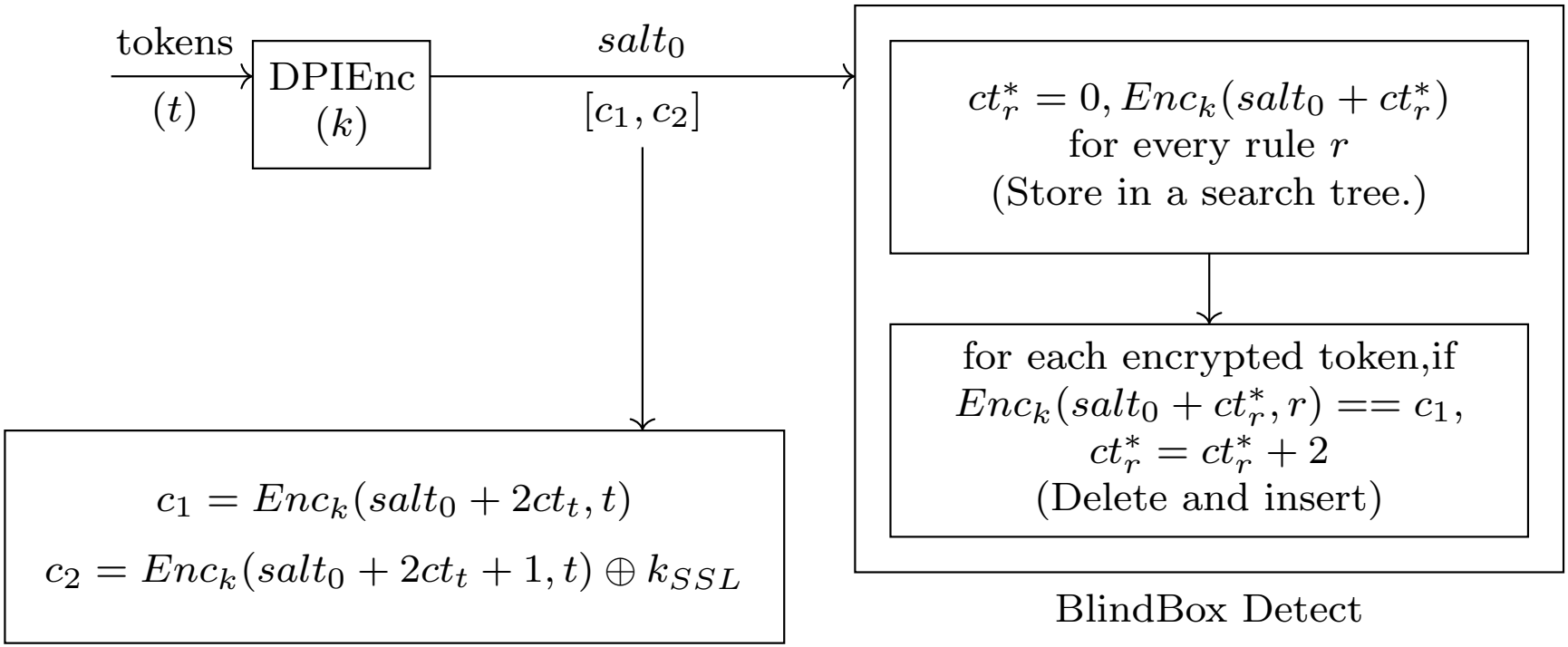
- Protocol II supports a limited form of an IDS.
- It allows a rule to contain :
  1. multiple keywords to be matched in the traffic
  2. absolute and relative offset information within the packet
- Builds on Protocol I with one exception.
- The client attaches to each encrypted token the offset in the stream where it appeared.



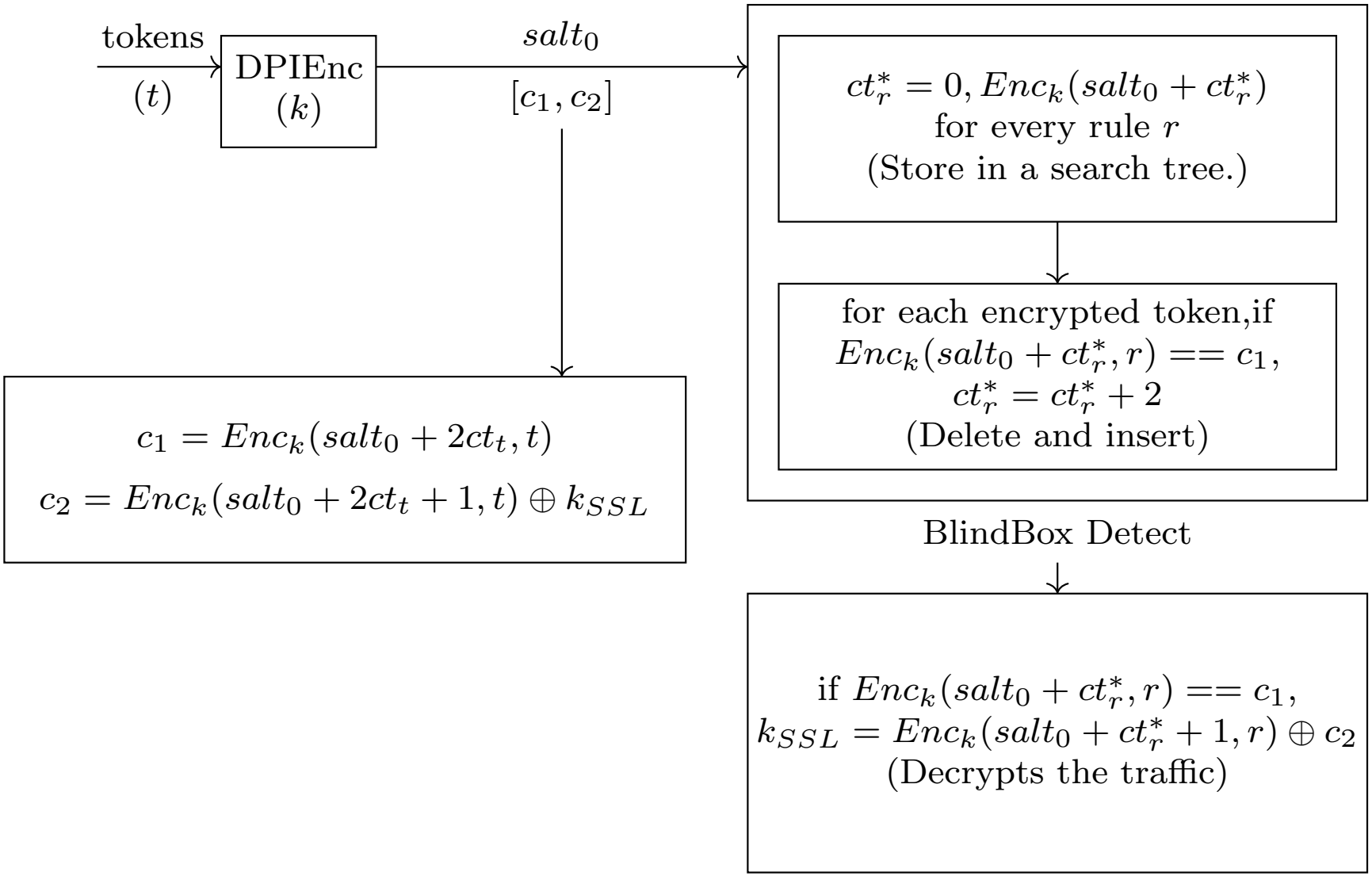
Protocol III

- Protocol III enables full IDS functionality.
- Compared to first approach, it ensures additional functionalities:
  1. If a keyword from a rule (a suspicious keyword) matches a stream of traffic, MB should be able to decrypt the traffic.
  2. If there is **no such match**, MB **cannot decrypt** the traffic.

Protocol III



Protocol III



Performance of Our BlindBox Model (2025 Benchmark)

Tokenization:

- Delimiter method: >1M tokens/sec even for large messages.
- Negligible overhead for practical use-cases.

Performance of Our BlindBox Model (2025 Benchmark)

Tokenization:

- Delimiter method: >1M tokens/sec even for large messages.
- Negligible overhead for practical use-cases.

Token Encryption:

- Protocol III: ∼21.8ms/token (∼46 tokens/sec).
- Now the main bottleneck for large messages.

Performance of Our BlindBox Model (2025 Benchmark)

Tokenization:

- Delimiter method: >1M tokens/sec even for large messages.
- Negligible overhead for practical use-cases.

Token Encryption:

- Protocol III: ∼21.8ms/token (∼46 tokens/sec).
- Now the main bottleneck for large messages.

System Components:

- AES message encryption: ∼5.7ms/message (∼176 messages/sec).
- AVL tree detection: 15,000–25,000 searches/sec; tree rebuild: ∼0.5 sec (4,939 rules).

Performance of Our BlindBox Model (2025 Benchmark)

Tokenization:

- Delimiter method: >1M tokens/sec even for large messages.
- Negligible overhead for practical use-cases.

Token Encryption:

- Protocol III: ∼21.8ms/token (∼46 tokens/sec).
- Now the main bottleneck for large messages.

System Components:

- AES message encryption: ∼5.7ms/message (∼176 messages/sec).
- AVL tree detection: 15,000–25,000 searches/sec; tree rebuild: ∼0.5 sec (4,939 rules).

Network Overhead:

- Each token: 37 bytes; ∼50–100% larger than plaintext.

Performance of Our BlindBox Model (2025 Benchmark)

Tokenization:

- Delimiter method: >1M tokens/sec even for large messages.
- Negligible overhead for practical use-cases.

Token Encryption:

- Protocol III: ∼21.8ms/token (∼46 tokens/sec).
- Now the main bottleneck for large messages.

System Components:

- AES message encryption: ∼5.7ms/message (∼176 messages/sec).
- AVL tree detection: 15,000–25,000 searches/sec; tree rebuild: ∼0.5 sec (4,939 rules).

Network Overhead:

- Each token: 37 bytes; ∼50–100% larger than plaintext.

Optimization Focus: Parallel token encryption, hardware acceleration (AES-NI).

Performance of BlindBox (Original Paper, 2015)

Rule Encryption:

- 1,000 rules: 9.5s; 10,000 rules: 97s (∼100 rules/sec).
- Significant bottleneck for large rulesets.

Performance of BlindBox (Original Paper, 2015)

Rule Encryption:

- 1,000 rules: 9.5s; 10,000 rules: 97s ($\sim$100 rules/sec).
- Significant bottleneck for large rulesets.

Tokenization:

- Qualitatively fast; negligible overhead claimed (no rate given).

Performance of BlindBox (Original Paper, 2015)

Rule Encryption:

- 1,000 rules: 9.5s; 10,000 rules: 97s ($\sim$100 rules/sec).
- Significant bottleneck for large rulesets.

Tokenization:

- Qualitatively fast; negligible overhead claimed (no rate given).

Encryption Overhead:

- 128-bit block: 69ns (BlindBox) vs 13ns (HTTPS).
- 1,400-byte packet: 90μs (BlindBox) vs 3μs (HTTPS).

Performance of BlindBox (Original Paper, 2015)

Rule Encryption:

- 1,000 rules: 9.5s; 10,000 rules: 97s ($\sim$100 rules/sec).
- Significant bottleneck for large rulesets.

Tokenization:

- Qualitatively fast; negligible overhead claimed (no rate given).

Encryption Overhead:

- 128-bit block: 69ns (BlindBox) vs 13ns (HTTPS).
- 1,400-byte packet: 90μs (BlindBox) vs 3μs (HTTPS).

Detection Throughput:

- 166–186 Mbps/core (middlebox), similar to or faster than Snort.

Performance of BlindBox (Original Paper, 2015)

Rule Encryption:

- 1,000 rules: 9.5s; 10,000 rules: 97s ($\sim$100 rules/sec).
- Significant bottleneck for large rulesets.

Tokenization:

- Qualitatively fast; negligible overhead claimed (no rate given).

Encryption Overhead:

- 128-bit block: 69ns (BlindBox) vs 13ns (HTTPS).
- 1,400-byte packet: 90μs (BlindBox) vs 3μs (HTTPS).

Detection Throughput:

- 166–186 Mbps/core (middlebox), similar to or faster than Snort.

Network Overhead:

- Delimiter tokenization: 2.5$\times$ increase; window: up to 4$\times$.

Performance of BlindBox (Original Paper, 2015)

Rule Encryption:

- 1,000 rules: 9.5s; 10,000 rules: 97s ($\sim$100 rules/sec).
- Significant bottleneck for large rulesets.

Tokenization:

- Qualitatively fast; negligible overhead claimed (no rate given).

Encryption Overhead:

- 128-bit block: 69ns (BlindBox) vs 13ns (HTTPS).
- 1,400-byte packet: 90μs (BlindBox) vs 3μs (HTTPS).

Detection Throughput:

- 166–186 Mbps/core (middlebox), similar to or faster than Snort.

Network Overhead:

- Delimiter tokenization: 2.5$\times$ increase; window: up to 4$\times$.

Main Bottleneck: Rule encryption/setup time for large IDS.

Performance of BlindBox (Original Paper, 2015)

Rule Encryption:

- 1,000 rules: 9.5s; 10,000 rules: 97s ($\sim$100 rules/sec).
- Significant bottleneck for large rulesets.

Tokenization:

- Qualitatively fast; negligible overhead claimed (no rate given).

Encryption Overhead:

- 128-bit block: 69ns (BlindBox) vs 13ns (HTTPS).
- 1,400-byte packet: 90μs (BlindBox) vs 3μs (HTTPS).

Detection Throughput:

- 166–186 Mbps/core (middlebox), similar to or faster than Snort.

Network Overhead:

- Delimiter tokenization: 2.5$\times$ increase; window: up to 4$\times$.

Main Bottleneck: Rule encryption/setup time for large IDS.
Optimization Focus: Cryptographic optimization, reducing setup, parallelism.

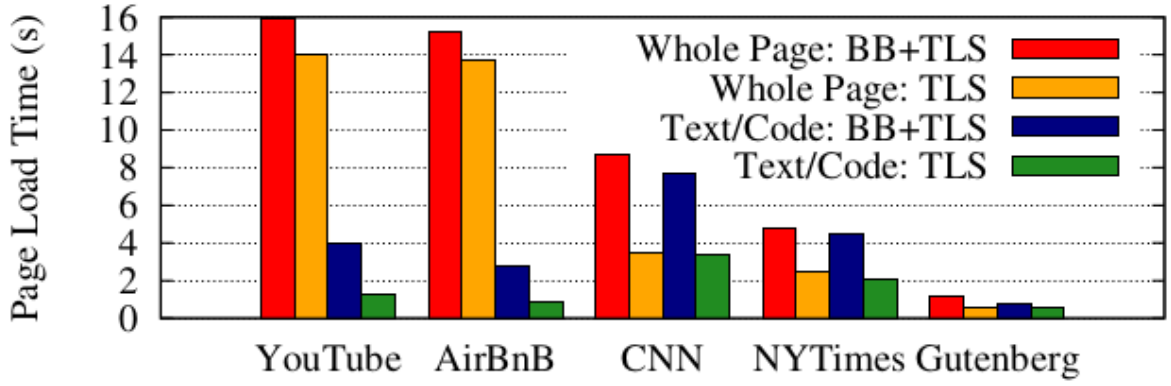


Figure 4: Download time for TLS and BlindBox (BB) + TLS at 20Mbps×10ms.

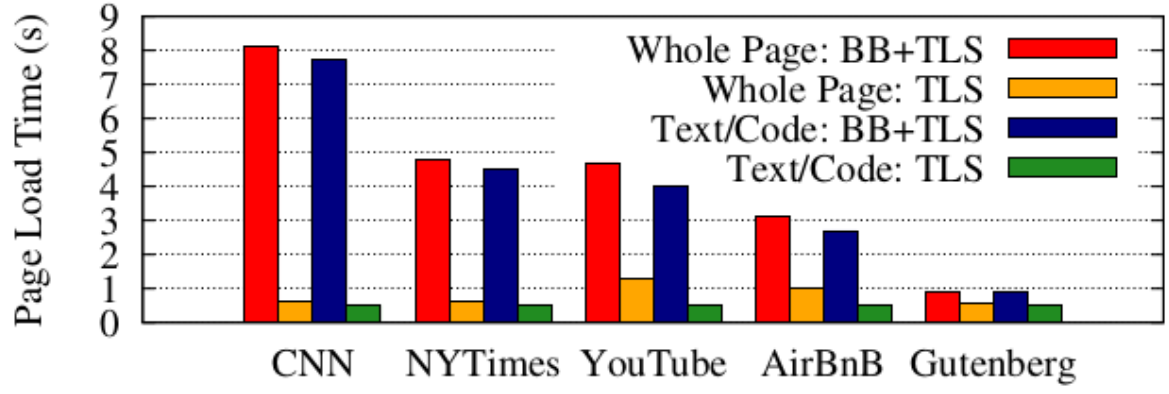


Figure 5: Download time for TLS and BlindBox (BB) + TLS at 1Gbps×10ms.

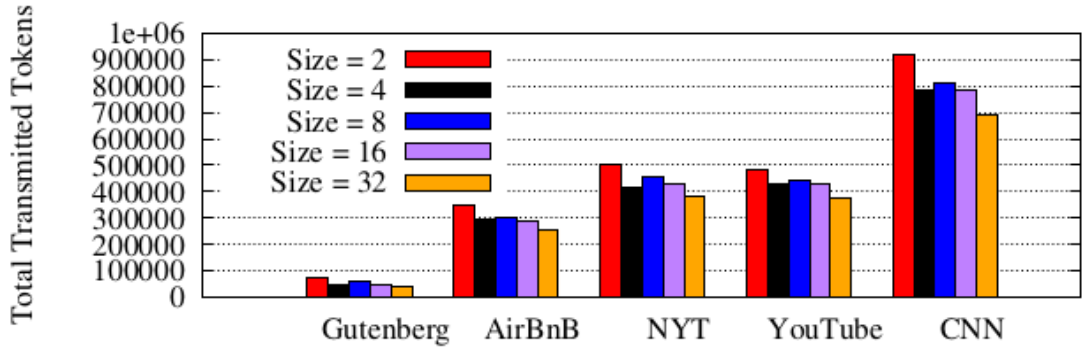


Figure 7: Tokens generated for each of six popular websites using delimiter-based tokenization and a minimum token size between 1-32 bytes.

Future Plans

Token encryption optimization

- Hardware acceleration (AES-NI)
- Parallel processing
- Algorithm optimization

Future Plans

Token encryption optimization

- Hardware acceleration (AES-NI)
- Parallel processing
- Algorithm optimization

Network optimization

- 37+ bytes per token overhead
- Compression and serialization improvements

Future Plans

Token encryption optimization

- Hardware acceleration (AES-NI)
- Parallel processing
- Algorithm optimization

Network optimization

- 37+ bytes per token overhead
- Compression and serialization improvements

Rule encryption (already efficient)

- 4,939 rules encrypt in 1.33 seconds
- Suitable for production use

Thank You!

Appendix

```
alert tcp $EXTERNAL_NET $HTTP_PORTS
                                    -> $HOME_NET 1025:5000 (
  flow: established,from_server;
  content: "Server|3a| nginx/0.";
  offset: 17; depth: 19;
  content: "Content-Type|3a| text/html";
  content: "|3a|80|3b|255.255.255.255"; )
```

- This rule is triggered if :
  1. the flow is from the server
  2. first keyword at an offset between 17 and 19.
  3. second and third keywords.
- If all the fields of the relevant rule are satisfied, MB takes the action indicated by the rule.