DynPRE: Protocol Reverse Engineering via Dynamic Inference

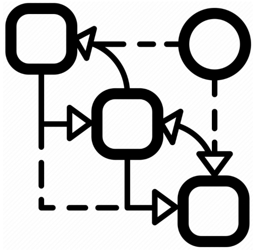
Zhengxiong Luo[1], Kai Liang[2], Yanyang Zhao[1], Feifan Wu[1], Junze Yu[1], Heyuan Shi[2], and Yu Jiang[1]

[1]Tsinghua University

[2]Central South University

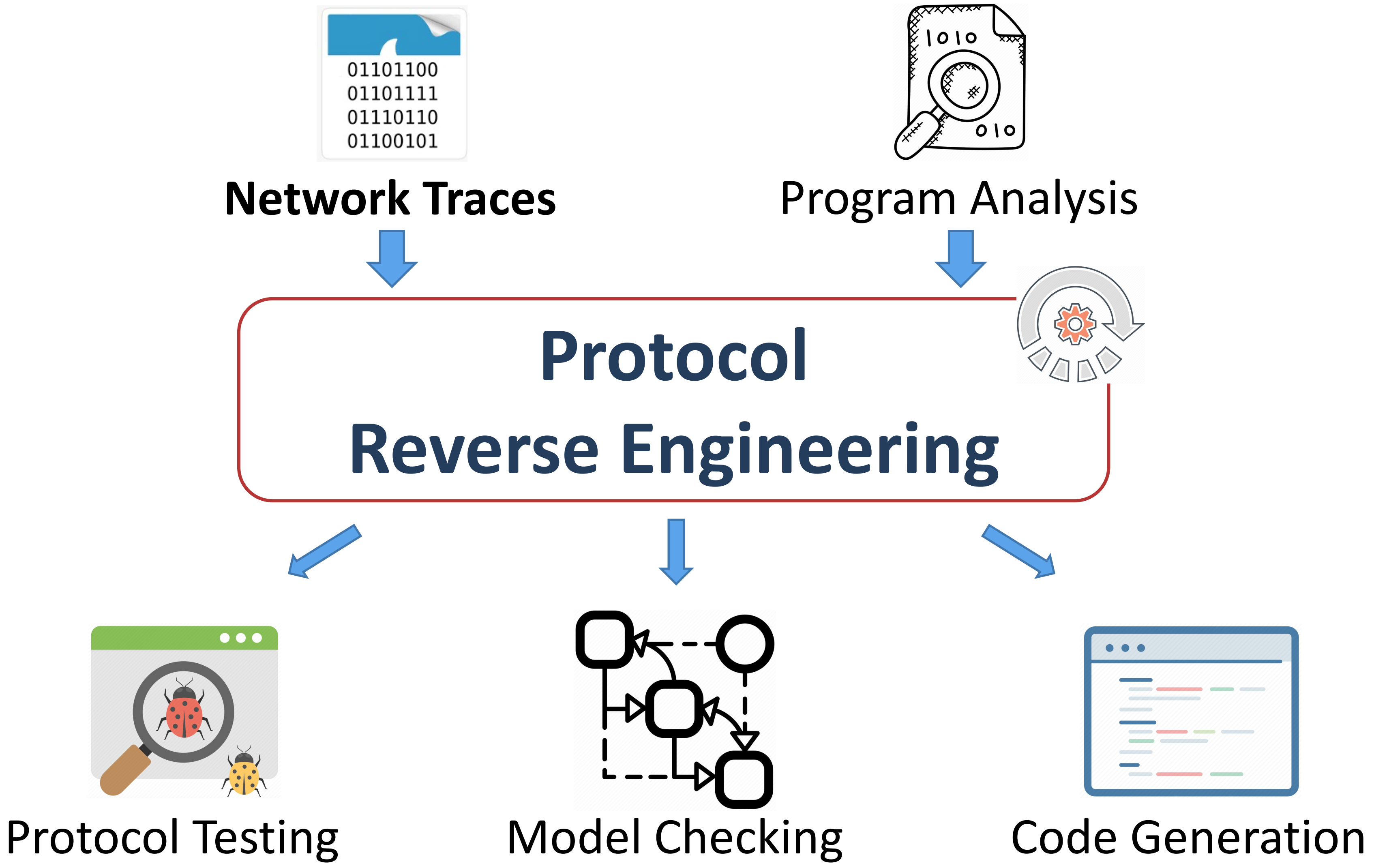
Protocol Reverse Engineering

Protocol Testing



Model Checking

Code Generation


01101100
01101111
01110110
01100101
Network Traces
1010
010
Program Analysis
Protocol
Reverse Engineering
Protocol Testing
Model Checking
Code Generation

Traditional Network Trace based Method

Employ **statistical analysis** on the input network traces

Traditional Network Trace based Method

Employ **statistical analysis** on the input network traces

- e.g., alignment-based method

Msg1 00 0D 00 00 00 14 FF 0F 00 01 00 62 0D 80 D6 3D 8F 15 A4 92 9A 09

Msg2 00 0E 00 00 00 10 FF 01 0D 80 D6 3D 8F 15 A4 92 9A 09

Align

Msg1	00	0D	00	00	00	14	FF	0F	00	01	00	62	0D	80	D6	3D	8F	15	A4	92	9A	09
Msg2	00	0E	00	00	00	10	FF	--	--	01	--	--	0D	80	D6	3D	8F	15	A4	92	9A	09

Traditional Network Trace based Method

Employ **statistical analysis** on the input network traces

- e.g., alignment-based method

Msg1 00 0D 00 00 00 14 FF 0F 00 01 00 62 0D 80 D6 3D 8F 15 A4 92 9A 09

Msg2 00 0E 00 00 00 10 FF 01 0D 80 D6 3D 8F 15 A4 92 9A 09

Align

Msg1	00	0D	00	00	00	14	FF	0F	00	01	00	62	0D	80	D6	3D	8F	15	A4	92	9A	09
Msg2	00	0E	00	00	00	10	FF	--	--	01	--	--	0D	80	D6	3D	8F	15	A4	92	9A	09

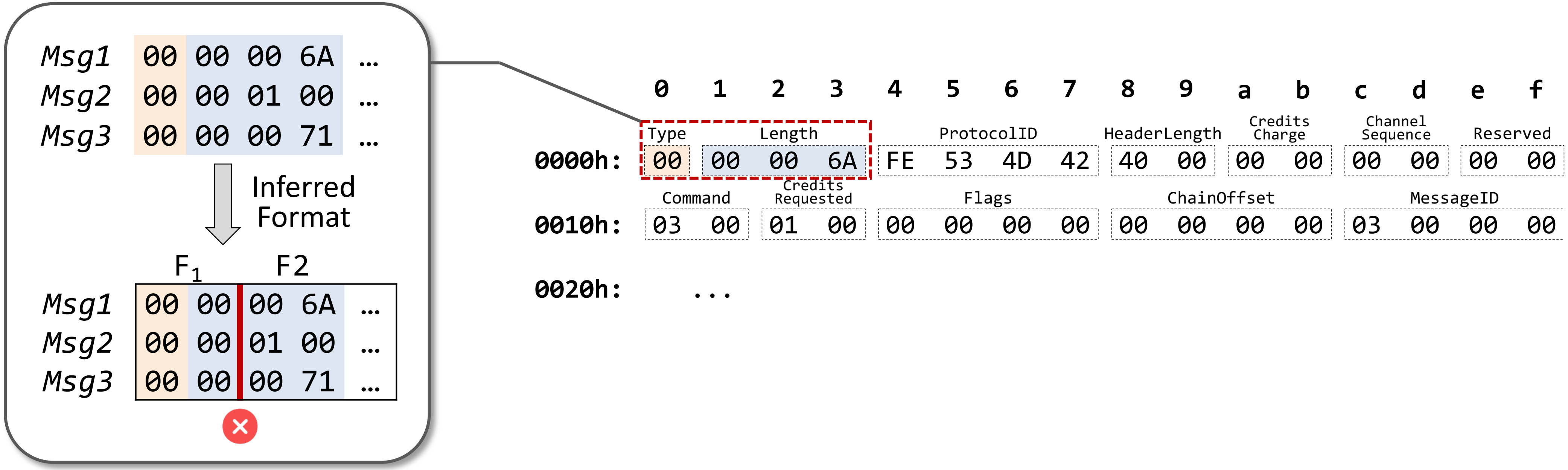
Infer Format

	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
Msg1	00	0D	00 00 00	14	FF	0F 00	01	00 62	0D 80 D6 3D 8F 15 A4 92 9A 09
Msg2	00	0E	00 00 00	10	FF	-- --	01	-- --	0D 80 D6 3D 8F 15 A4 92 9A 09

Constant Variable

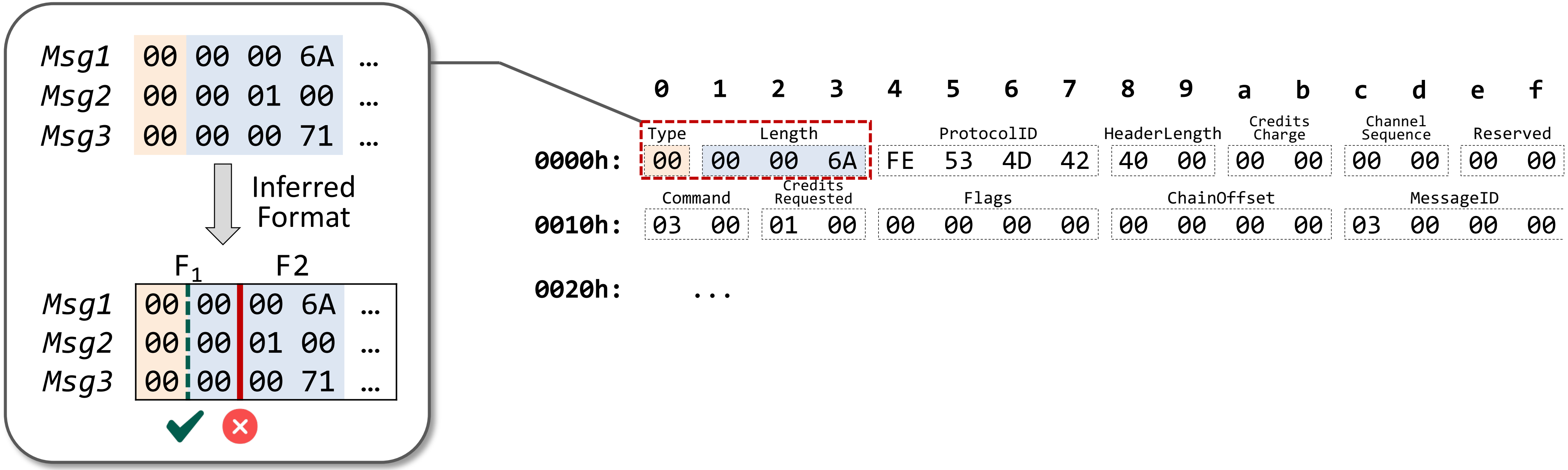
Weakness of the Traditional Method

- **Require high-quality network traces** that contain diverse messages and cover most protocol features



Weakness of the Traditional Method

- **Require high-quality network traces** that contain diverse messages and cover most protocol features



Weakness of the Traditional Method

- Require high-quality network traces
- **Lacks precision in capturing field semantics**

Msg1 … FE **53** 00 00 …

Msg2 00 4D 01 **53** …

Equal Value → ? → *Equal Semantic*

Insights

In protocol reverse engineering applications like fuzzing, **servers' interactive capabilities are exploitable**

Insights

In protocol reverse engineering applications like fuzzing, **servers' interactive capabilities are exploitable**

By establishing active communication with the server, we can:

- **Acquire additional message samples** as needed by interaction
- **Extract insights from the server** as it already encodes the protocol logic and understands the messages

Challenges

C1: How to **interact with the server** without protocol specifications

- Proper interaction requires **sequential, well-formed messages**
- Input traces serve as a reference, but require **resolving session-specific identifiers**

Challenges

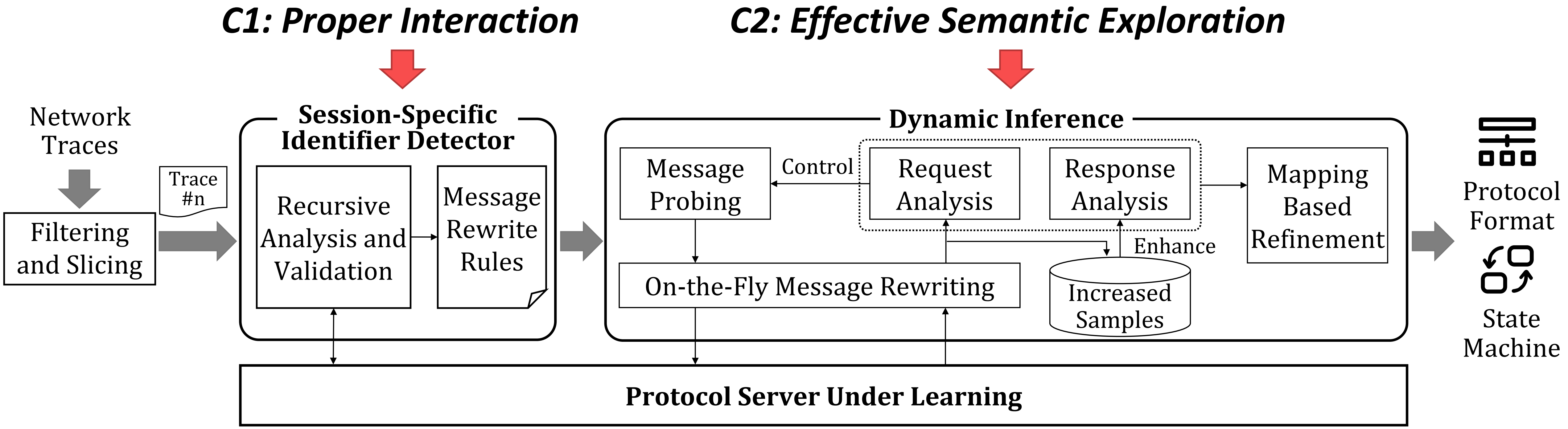
C1: How to **interact with the server** without protocol specifications

- Proper interaction requires **sequential, well-formed messages**
- Input traces serve as a reference, but require **resolving session-specific identifiers**

C2: **Effectively explore the interactive server** for protocol understanding

- Applicable across protocols, **inducing diverse server behaviors**

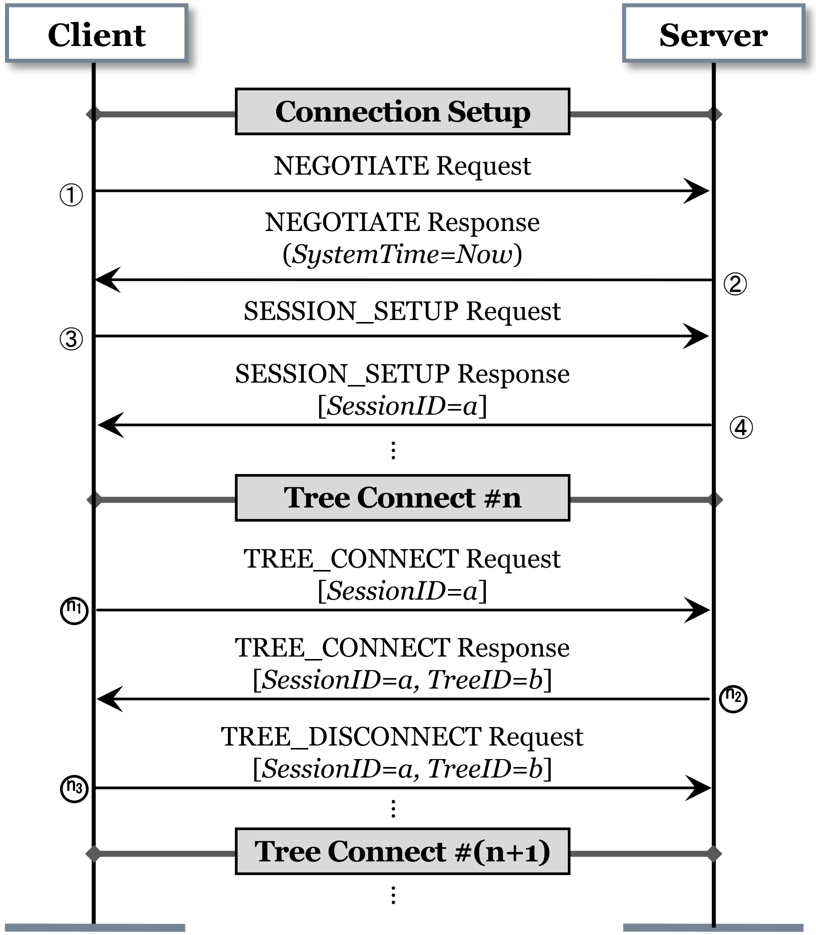
DynPRE Overview



Session-Specific Identifier

Feature

- **Dynamically assigned by the server** to keep track of contextual information **for each session**
- **Subsequent requests should carry valid values** for these fields

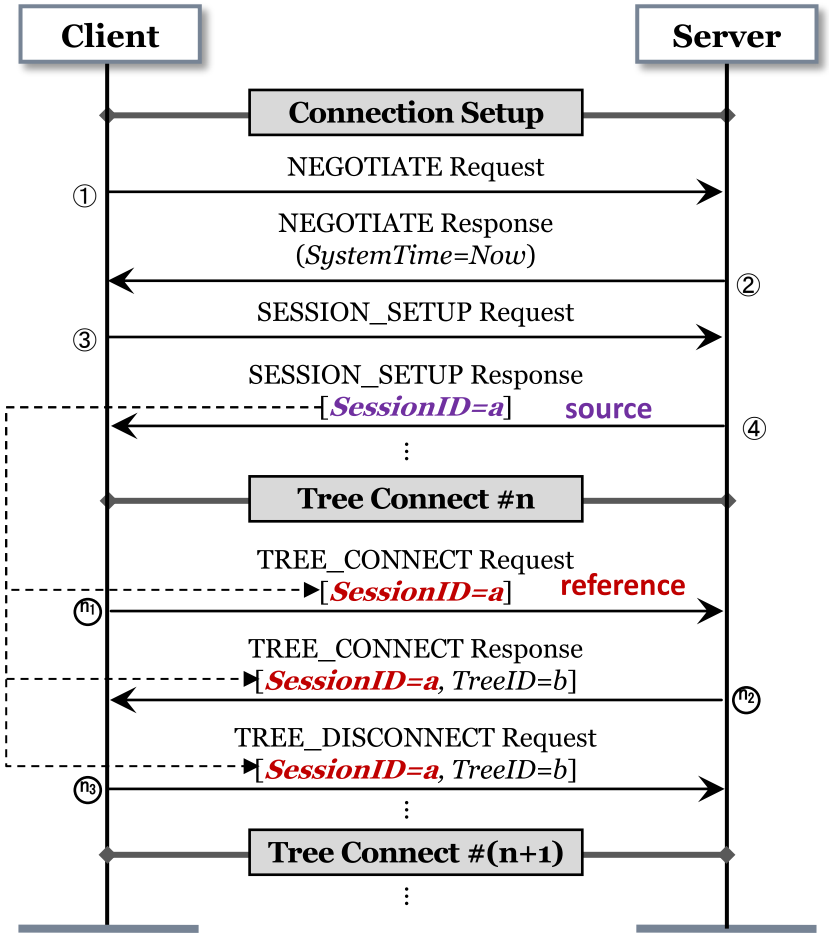


[Session-Specific Identifier]

Session-Specific Identifier

Feature

- **Dynamically assigned by the server** to keep track of contextual information **for each session**
- **Subsequent requests should carry valid values** for these fields

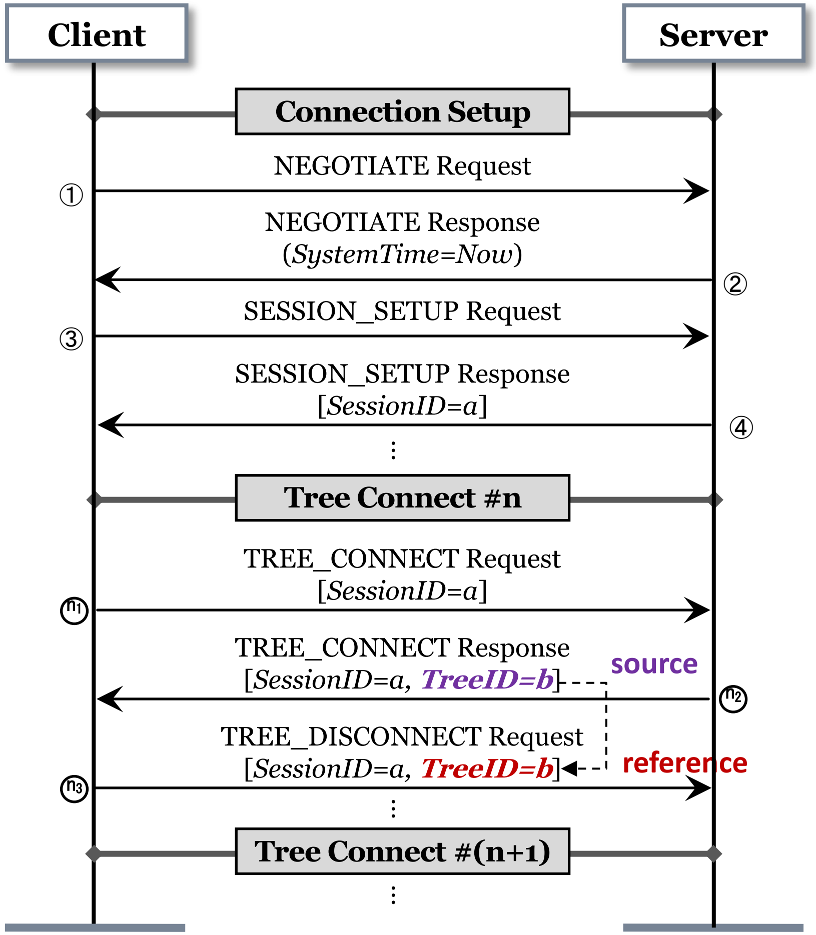


[Session-Specific Identifier]

Session-Specific Identifier

Feature

- **Dynamically assigned by the server** to keep track of contextual information **for each session**
- **Subsequent requests should carry valid values** for these fields

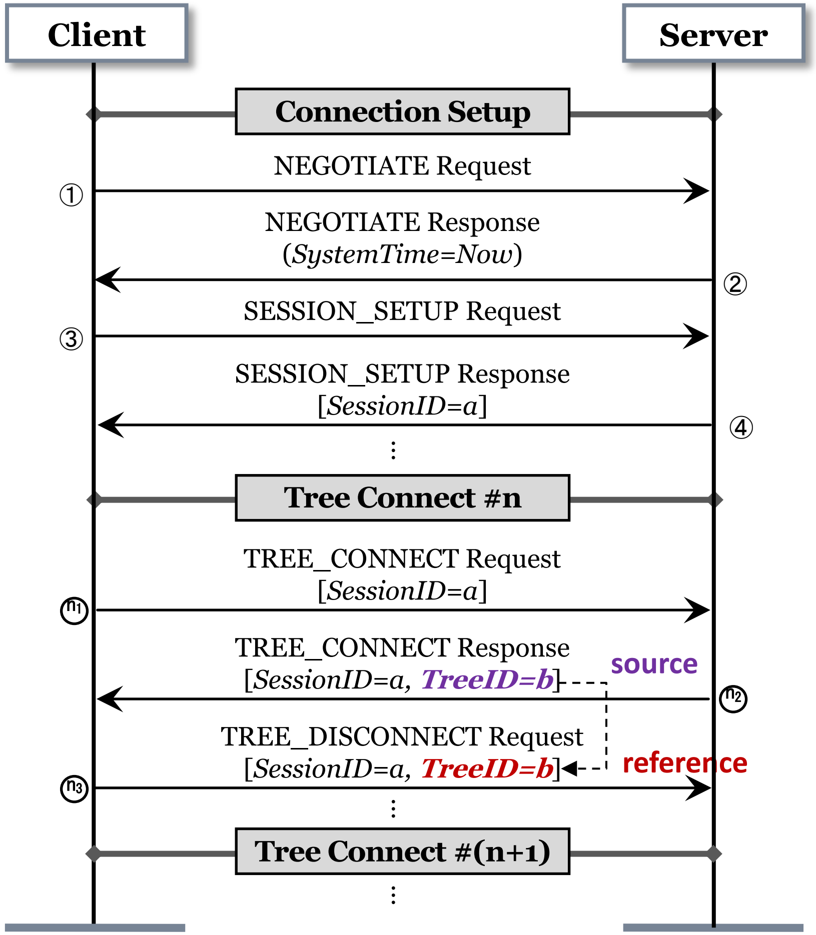


[Session-Specific Identifier]

Session-Specific Identifier

Feature

- **Dynamically assigned by the server** to keep track of contextual information **for each session**
- **Subsequent requests should carry valid values** for these fields
- **Constraint relationships** between sources and references can be **diverse**

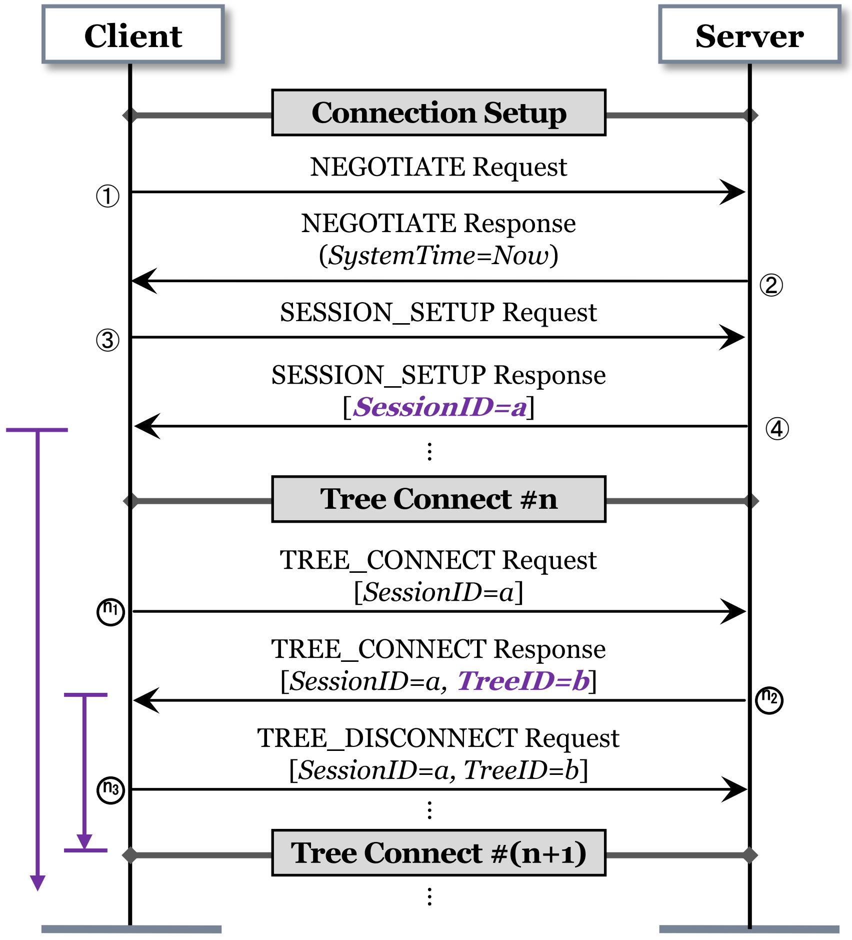


[Session-Specific Identifier]

Session-Specific Identifier

Feature

- **Dynamically assigned by the server** to keep track of contextual information **for each session**
- **Subsequent requests should carry valid values** for these fields
- **Constraint relationships** between sources and references can be diverse
- Different identifiers may have **different lifetimes**

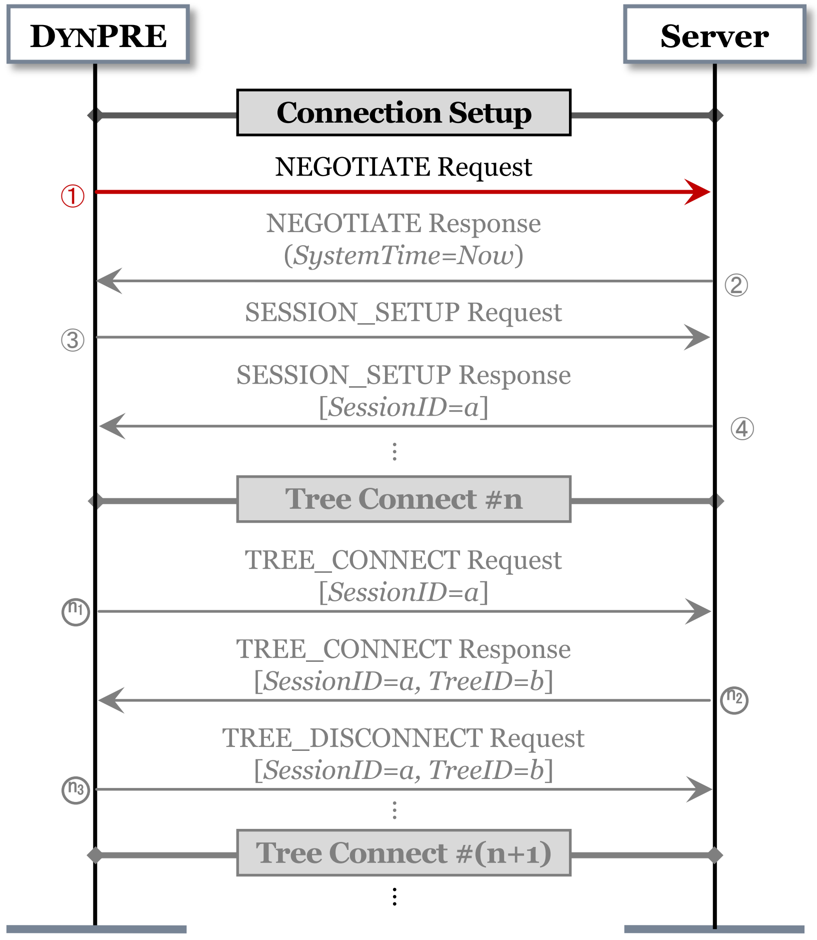


[Session-Specific Identifier]

Session-Specific Identifier Detection

Recursive Analysis and Validation

① Try to replay each request in the input network trace sequentially



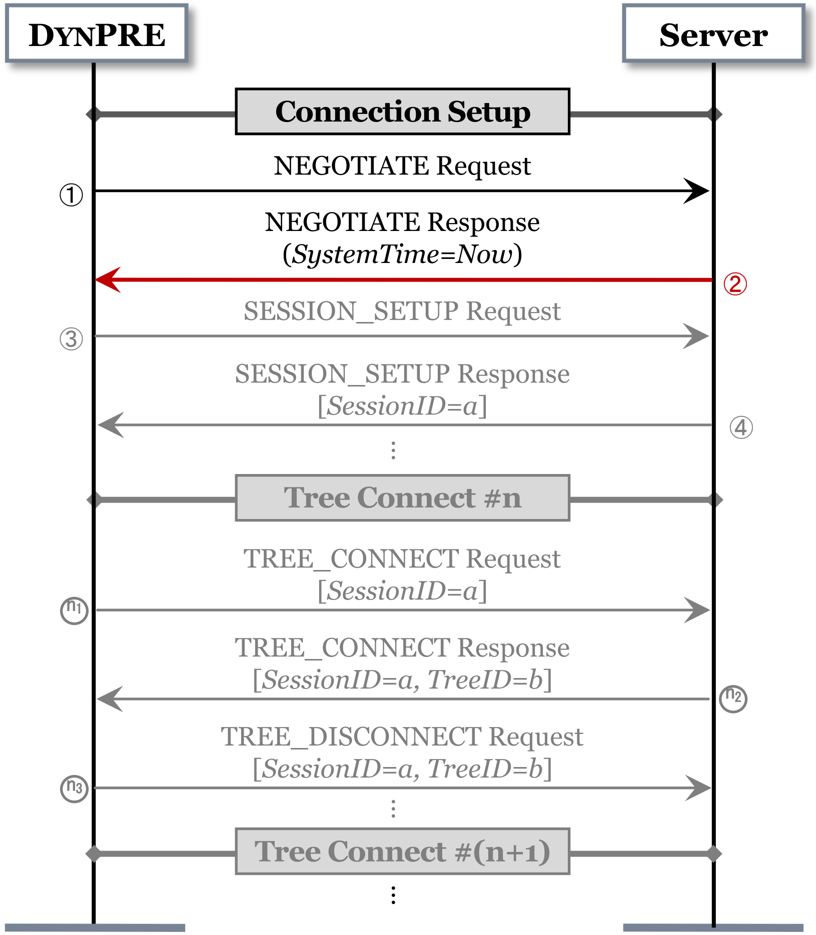
[Session-Specific Identifier]

Session-Specific Identifier Detection

Recursive Analysis and Validation

① Try to replay each request in the input network trace sequentially

② Obtain the live response and compare it with the original response in the input network trace



[Session-Specific Identifier]

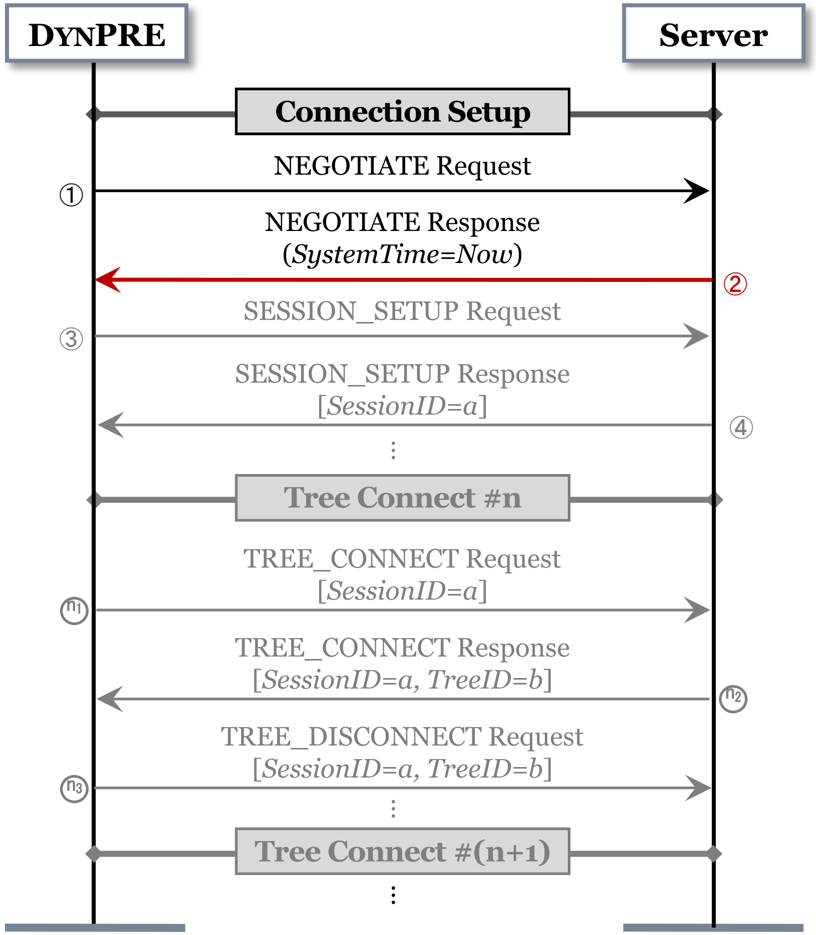
Session-Specific Identifier Detection

Recursive Analysis and Validation

① Try to replay each request in the input network trace sequentially

② Obtain the live response and compare it with the original response in the input network trace

- ➢ If they are the same, continue to replay the next request



[Session-Specific Identifier]

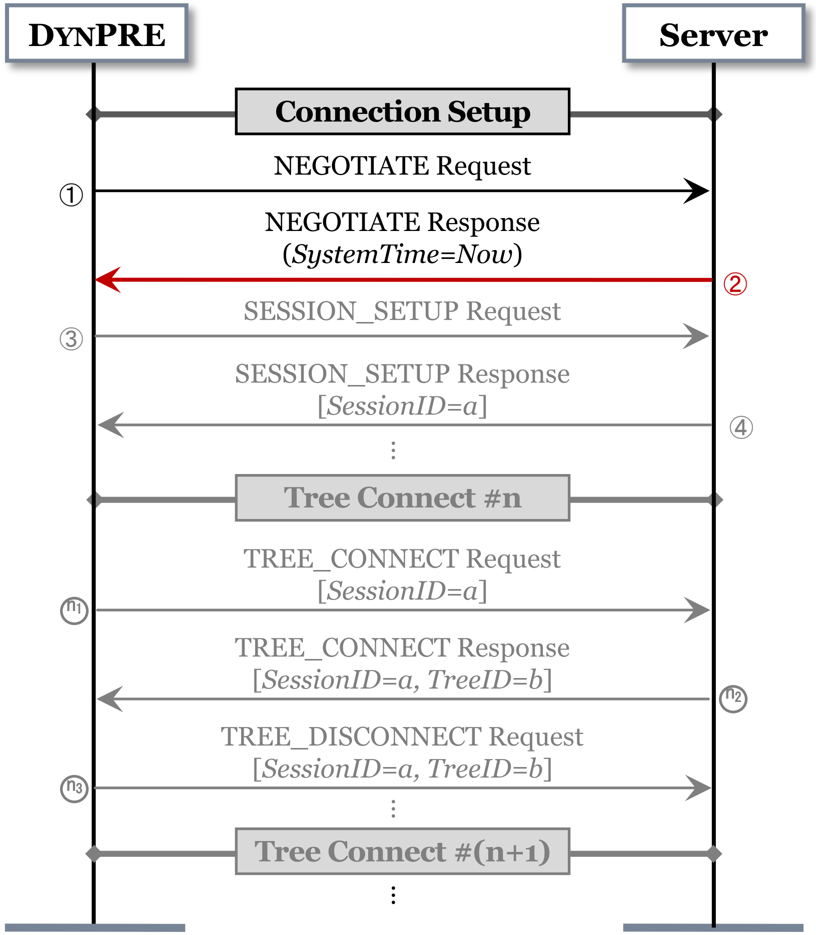
Session-Specific Identifier Detection

Recursive Analysis and Validation

① Try to replay each request in the input network trace sequentially

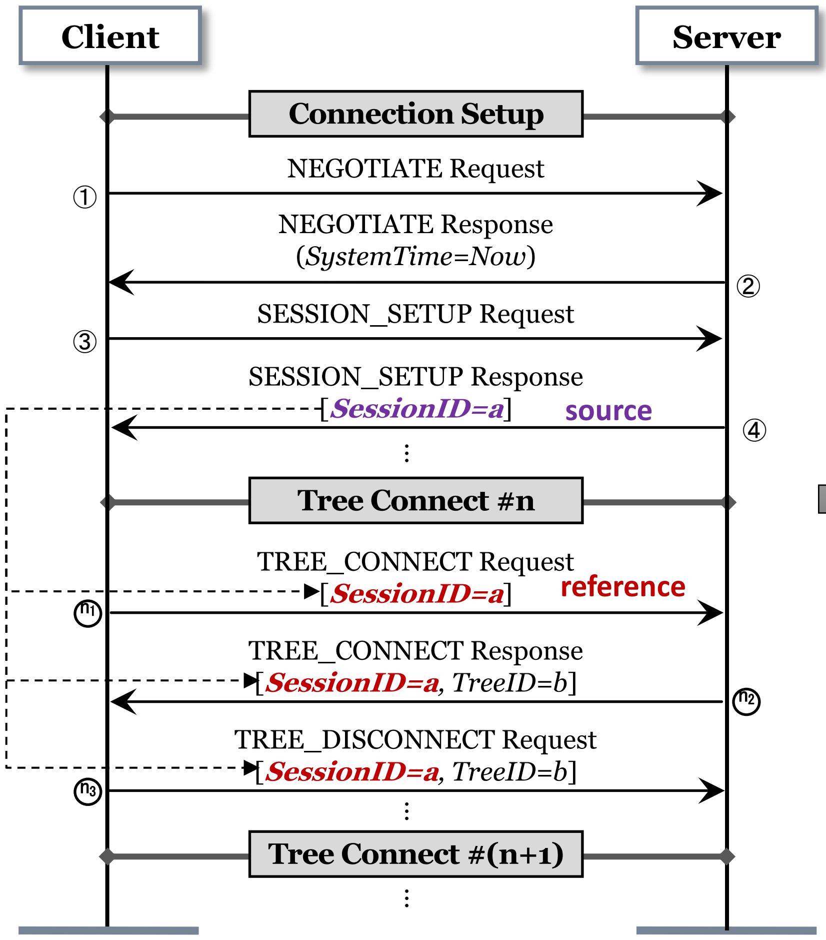
② Obtain the live response and compare it with the original response in the input network trace

- If they are the same, continue to replay the next request
- If not, this means that the response may contain session-specific identifiers. Get the differing byte regions and try to use constraint-solving list [x, x+1, px, px+1, $null$] to identify



[Session-Specific Identifier]

Session-Specific Identifier Detection

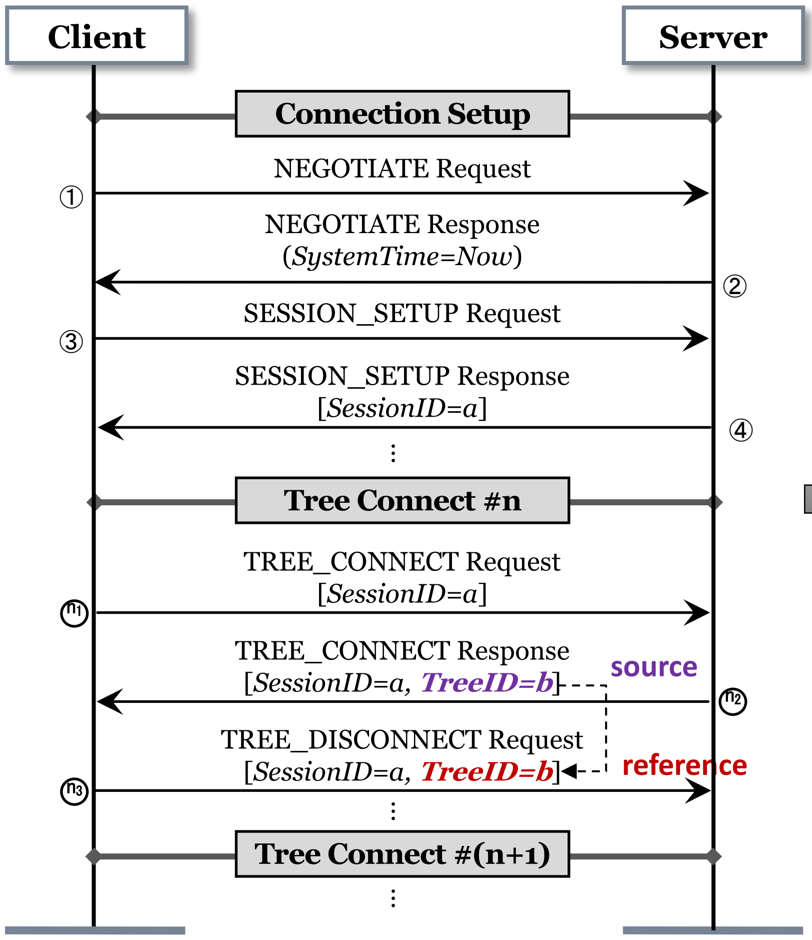


[Session-Specific Identifier]

Message Rewrite Rules

No.	Source	References	Constraint
1	4: [44..51]	n1: [44..51], n2: [44..51], n3:[44..51]	$y = x$
2	n2: [40..43]	n3: [40..43]	$y = x$

Session-Specific Identifier Detection

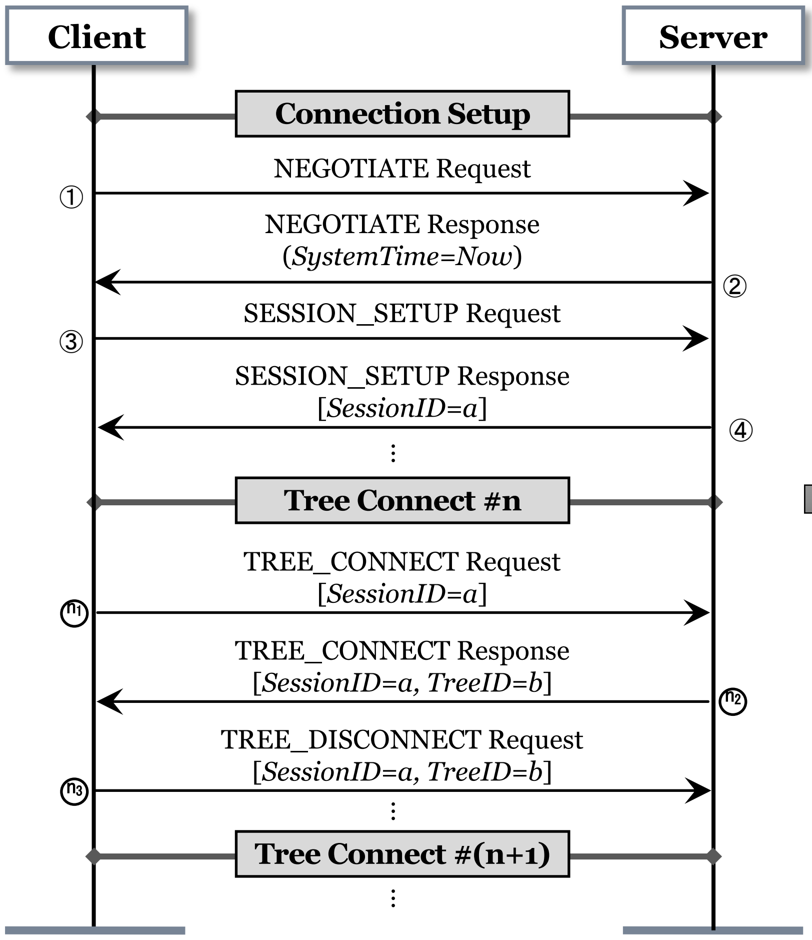


[Session-Specific Identifier]

Message Rewrite Rules

No.	Source	References	Constraint
1	④: [44..51]	n1: [44..51], n2: [44..51], n3:[44..51]	$y = x$
2	n2: [40..43]	n3: [40..43]	$y = x$

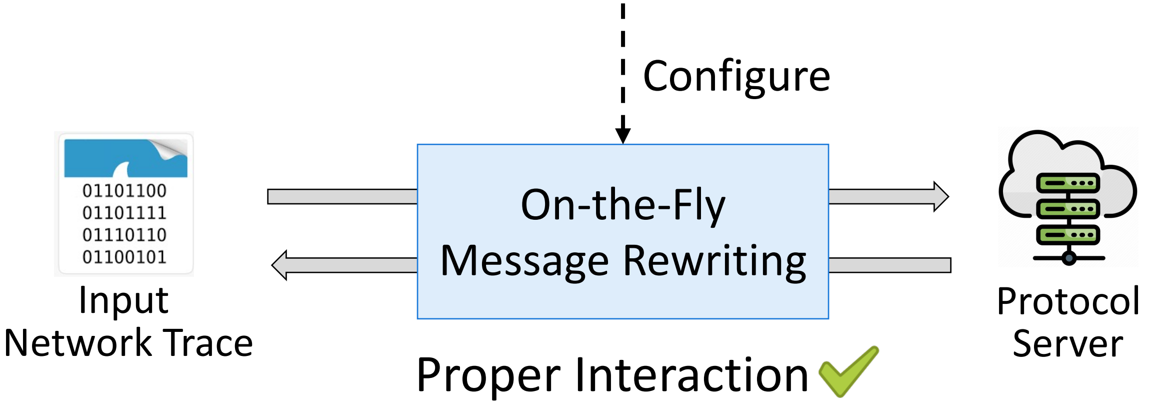
Session-Specific Identifier Detection



[Session-Specific Identifier]

Message Rewrite Rules

No.	Source	References	Constraint
1	4: [44..51]	n1: [44..51], n2: [44..51], n3:[44..51]	$y = x$
2	n2: [40..43]	n3: [40..43]	$y = x$



Dynamic Inference

Based on request message probing

Network Trace

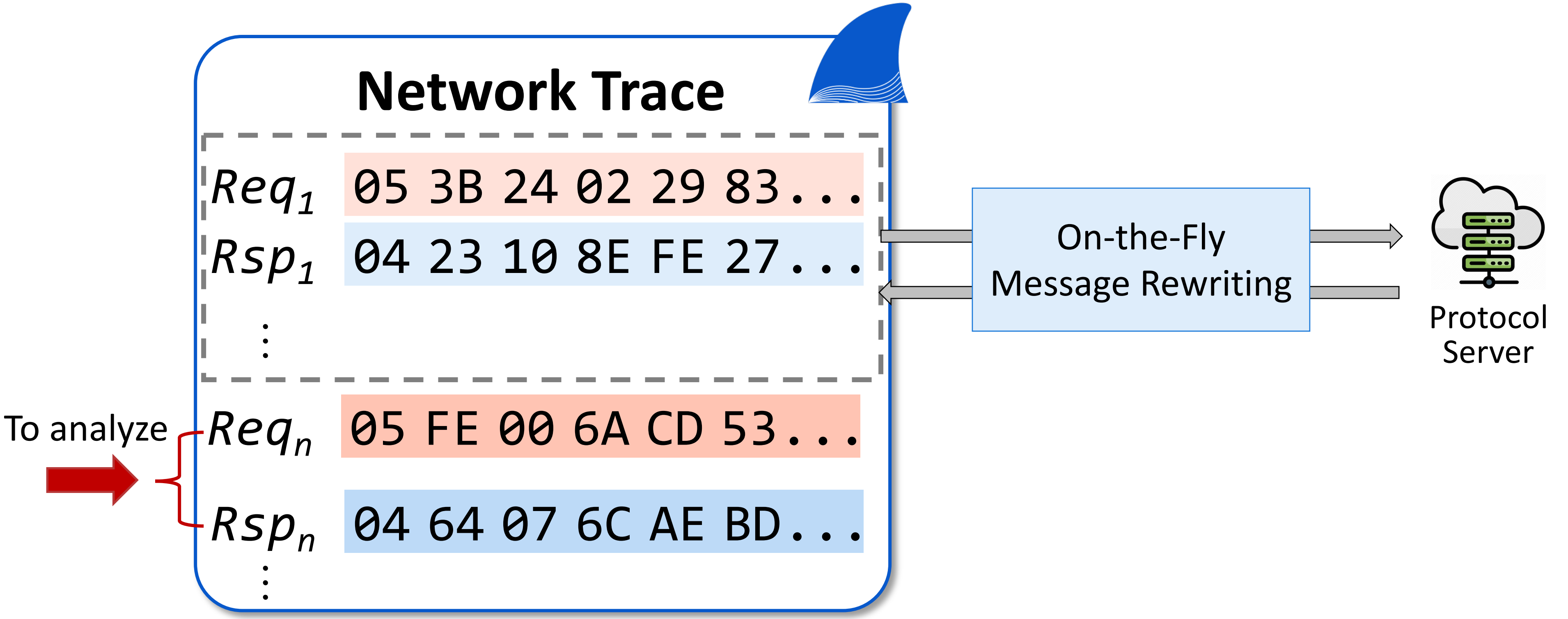
Req_1	`05 3B 24 02 29 83...`
Rsp_1	`04 23 10 8E FE 27...`
⋮	
Req_n	`05 FE 00 6A CD 53...`
Rsp_n	`04 64 07 6C AE BD...`
⋮	

To analyze $\left\{ Req_n, Rsp_n \right.$

Dynamic Inference

Based on request message probing

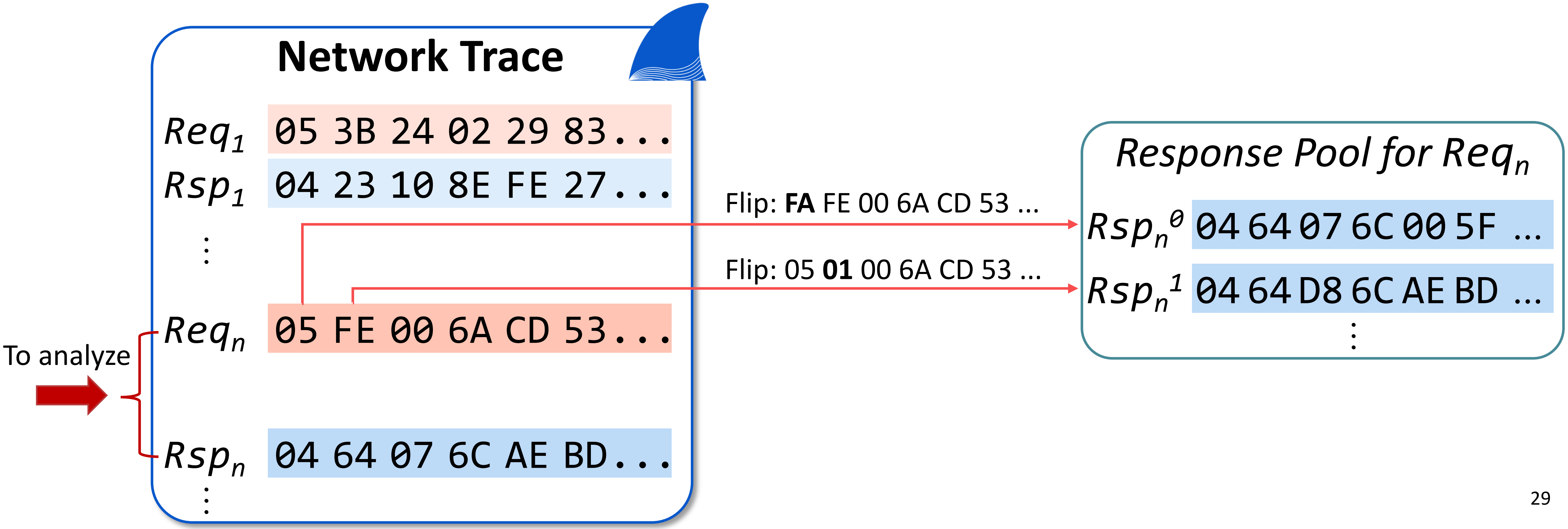
1. Replay preceding requests to drive the server into an appreciate state



Dynamic Inference

Based on request message probing

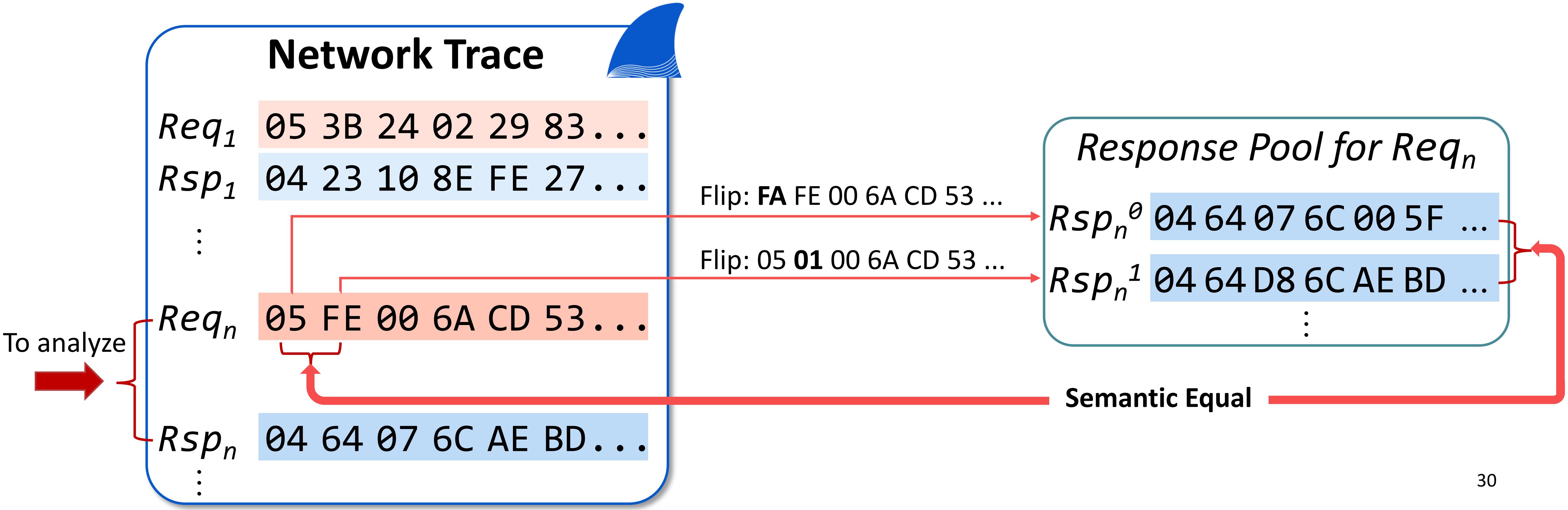
2. Flip each byte individually and scrutinize the corresponding responses



Dynamic Inference

Based on request message probing

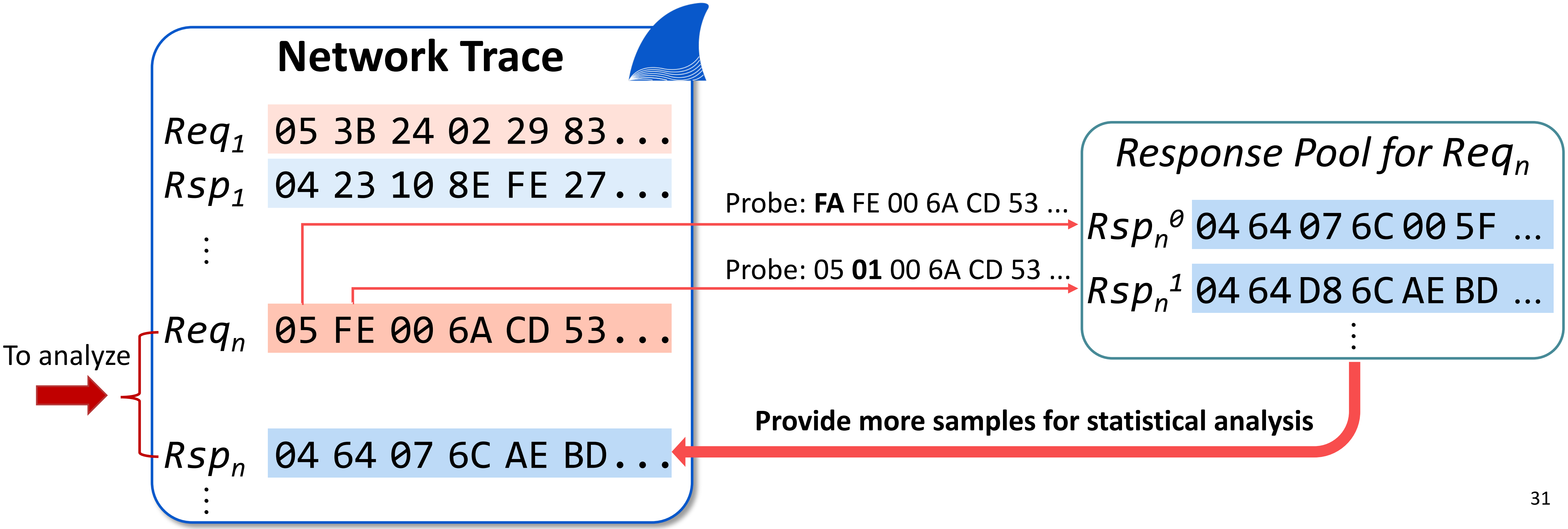
3. Request analysis: each byte's corresponding response indicates its semantic



Dynamic Inference

Based on request message probing

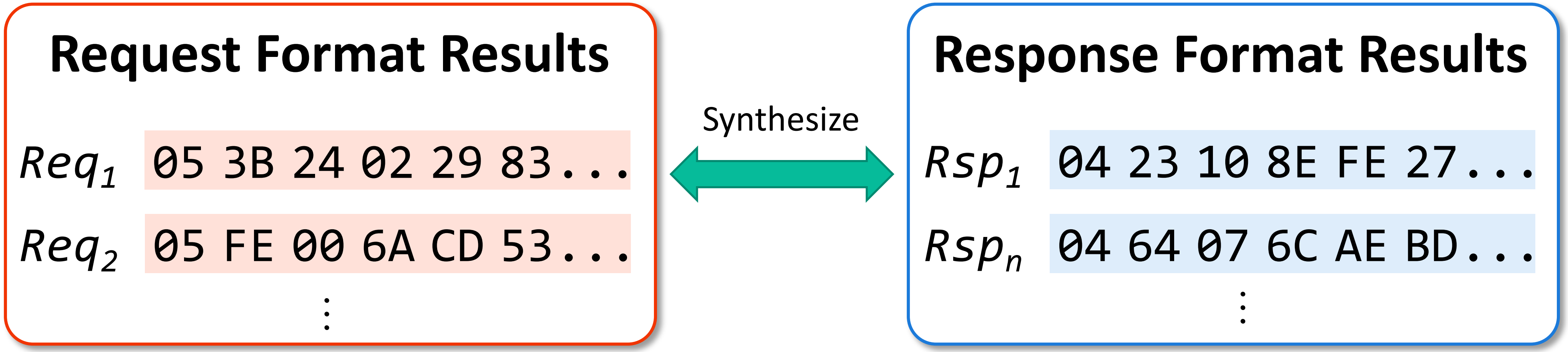
4. Response analysis: responses in the pool are likely to be roughly similar



Dynamic Inference

Based on request message probing

5. Mapping based refinement: synthesize request and response results to enhance outcomes and identify message types



Analyze based on dynamic inference

Analyze based on statistical anlysis

Evaluation

Compared Tools

- Netplier [NDSS’21]
- BinaryInferno [NDSS’23]
- Netzob [AsiaCCS’14]
- Nemesys [WOOT’18]
- FieldHunter [IFIP’15]

Public Protocols

- IEC61850-MMS
- S7comm
- Modbus
- MQTT-QoS1/2
- AMQP
- SMB
- SMB2
- HTTP
- NTP
- DNS
- BGP
- TFTP

Comparison on Static Dataset — Format Inference Result

Outperforms existing tools on all metrics on different-sized datasets

- Average **perfectly inferred field ratio**:

  DYNPRE — 50%

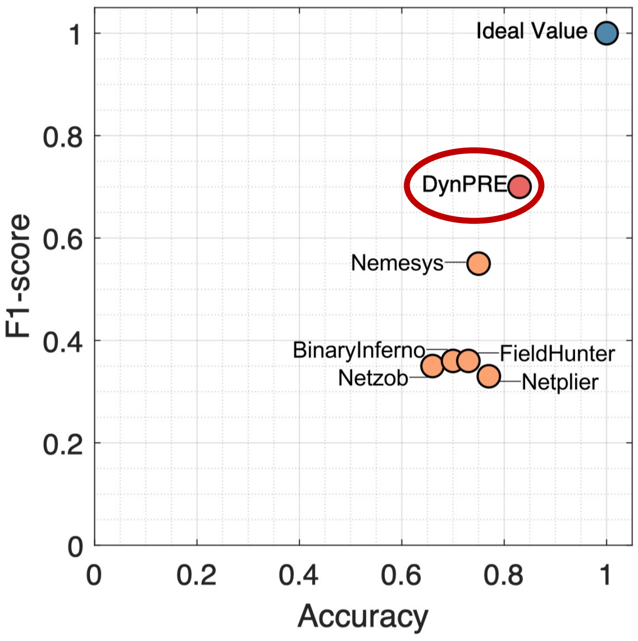
  BinaryInferno — 15% (**3.3**×)

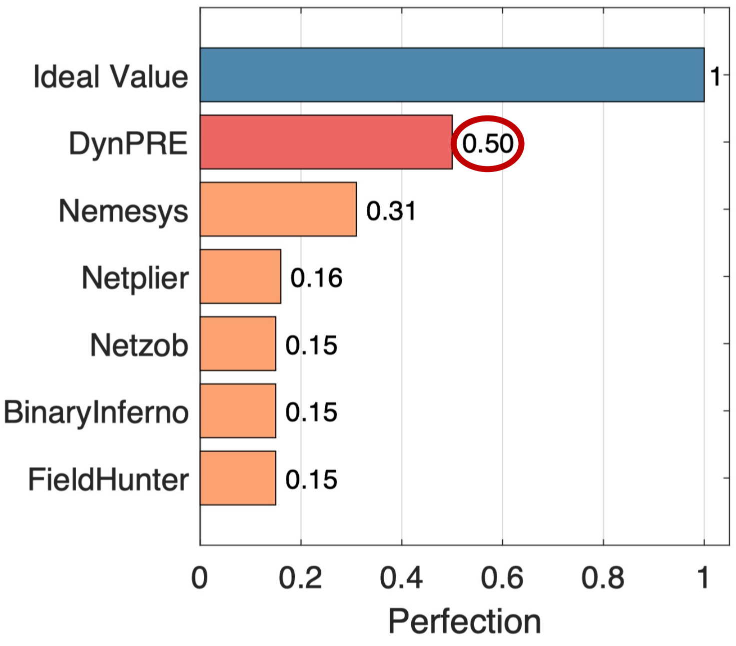
  Netzob — 15% (**3.3**×)

  FieldHunter — 15% (**3.3**×)

  Netplier — 16% (**3.1**×)

  Nemesys — 31% (**1.6**×)





Comparison on Static Dataset — State Machine Inference Result

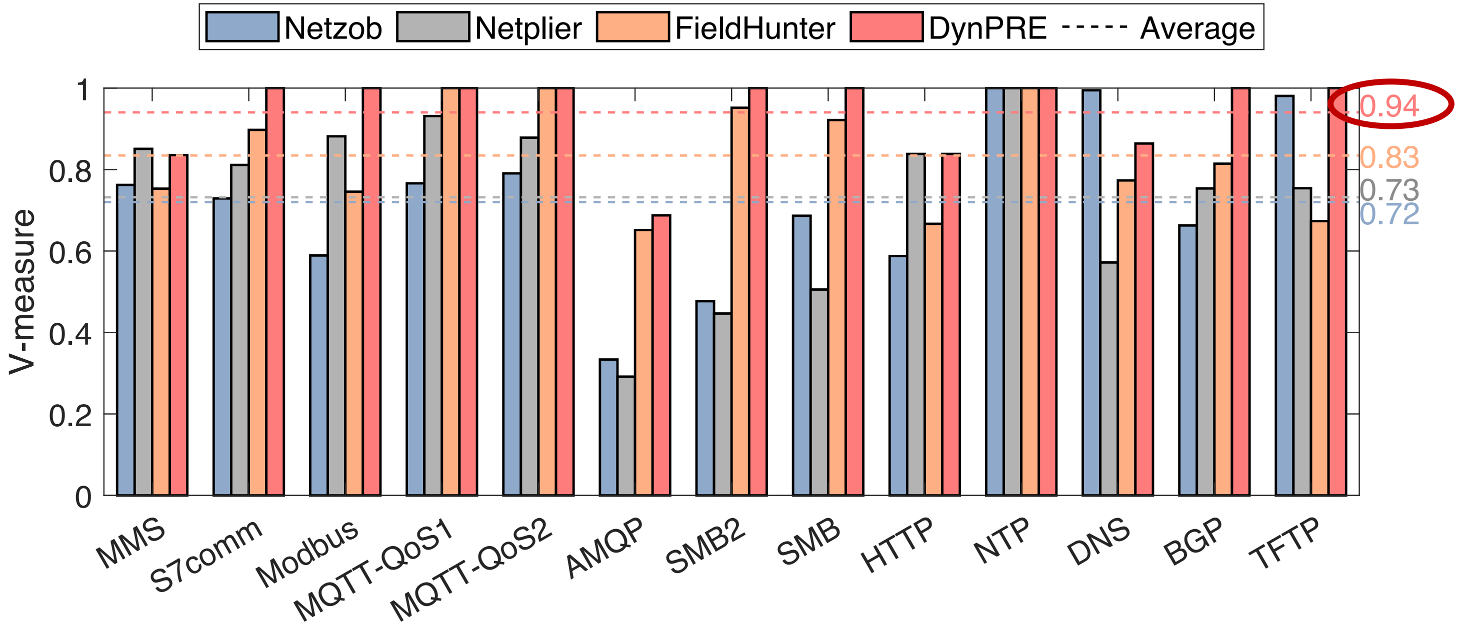
Metrics

- **Message type inference** is the key to protocol automata construction
- **Treat message type inference as a clustering problem**
- Use **widely accepted metrics**: homogeneity, completeness, and **V-measure**

Comparison on Static Dataset — State Machine Inference Result

Average V-measure Results

- **DYNPRE — 94%**
- Netzob — 72% (**+22%**)
- Netplier — 73% (**+21%**)
- FieldHunter — 83% (**+11%**)
- Nemesys — Not supported
- BinaryInferno — Not supported



Comparison on Enhenced Dataset

Enhance original datasets **by dynamic interaction** for compared tools

- $S_{initial}$: Initial static message dataset
- S_{DynPRE}: Additional message samples derived by DYNPRE
- $S_{BooFuzz}$: Use $S_{initial}$ to initialize fuzzer BooFuzz and obtain the derived additional message samples

Comparison on Enhenced Dataset

Results: **improvement is limited**, and certain results are even worse.
DynPRE is still superior.

		DynPRE	Netplier	BinaryInferno	Netzob	Nemesys	FieldHunter
Format Inference	Accuracy	**0.84**	0.77, 0.78 (0.78)	0.71, 0.69 (0.69)	0.70, 0.70 (0.67)	0.76, 0.77 (0.76)	0.71, 0.74 (0.74)
	F1-score	**0.72**	0.44, 0.44 (0.37)	0.34, 0.29 (0.35)	0.42, 0.42 (0.34)	0.56, 0.57 (0.55)	0.27, 0.34 (0.35)
	Perfection	**0.51**	0.25, 0.28 (0.21)	0.14, 0.13 (0.14)	0.23, 0.23 (0.15)	0.34, 0.35 (0.32)	0.09, 0.14 (0.13)
Message Type Inference	V-measure	**0.88**	0.62, 0.68 (0.62)	-	0.72, 0.72 (0.66)	-	0.79, 0.79 (0.81)

Format: S_{DynPRE}, $S_{BooFuzz}$ ($S_{initial}$) . Underlined for decreases, bold for **best**

Comparison on Enhenced Dataset

Results: **improvement is limited**, and certain results are even worse. DynPRE is still superior.

		DynPRE	Netplier	BinaryInferno	Netzob	Nemesys	FieldHunter
Format Inference	Accuracy	**0.84**	0.77, 0.78 (0.78)	0.71, 0.69 (0.69)	0.70, 0.70 (0.67)	0.76, 0.77 (0.76)	0.71, 0.74 (0.74)
	F1-score	**0.72**	0.44, 0.44 (0.37)	0.34, 0.29 (0.35)	0.42, 0.42 (0.34)	0.56, 0.57 (0.55)	0.27, 0.34 (0.35)
	Perfection	**0.51**	0.25, 0.28 (0.21)	0.14, 0.13 (0.14)	0.23, 0.23 (0.15)	0.34, 0.35 (0.32)	0.09, 0.14 (0.13)
Message Type Inference	V-measure	**0.88**	0.62, 0.68 (0.62)	-	0.72, 0.72 (0.66)	-	0.79, 0.79 (0.81)

Format: S_{DynPRE}, $S_{BooFuzz}$ ($S_{initial}$) . Underlined for decreases, bold for **best**

Reason — different strategies for exploring interactive traffic

- DYNPRE: correlates the modification operations with the server feedback
- Other tools: rely exclusively on statistical analysis

Proprietary Protocol Analysis

A **three-step evaluation process** derived from protocols' application

❶ Input message construction

Device	Behaviors of Input Messages	Message Format	# Triggered Behaviors
Yeelight LED Screen Light Bar Pro	Turn On	V(18) **V(9)** C(13) **V(2)** V(3) V(6) V(7) C(2)	Turn On, Brighten, Turn Off, Dim
	Brighten	V(18) **V(10)** C(12) **V(2)** V(2) C(6) V(7) C(2)	
Philips Hue Bridge	Create Group	**V(7)** C(28) V(15) **V(9)** V(57) C(1) V(11) V(7) V(1) V(3) **V(36)**	Set Name, Create Group, Output Name, Delete Group
	Set Name	**V(57)** V(57) C(1) V(7) V(1) V(5) V(5) V(1) V(1) V(2) **V(21)**	
Broadlink Smart Plug	Turn Off	V(32) V(4) C(2) C(2) V(2) C(10) C(1) V(1) V(2) V(4) V(2) **V(26)**	Turn On, Turn Off
Xiaomi Mijia Smart Camera*	Turn On	C(12) V(4) V(9) V(7) V(21) C(9) V(4) V(1) V(2) V(3) C(5) **V(2)** C(4)	Turn On, Turn Off
Tplink Router	Add Forbidden Domain	C(2) C(10) C(12) C(13) C(10) V(15) C(11) C(6) C(3) **V(13)** C(14) **V(3)** C(2)	Add Forbidden Domain, Clear Forbidden Domains, Output Forbidden Domains

Proprietary Protocol Analysis

A **three-step evaluation process** derived from protocols' application

❶ Input message construction

❷ Protocol reverse engineering

Device	Behaviors of Input Messages	Message Format	# Triggered Behaviors
Yeelight LED Screen Light Bar Pro	Turn On	V(18) **V(9)** C(13) **V(2)** V(3) V(6) V(7) C(2)	Turn On, Brighten, Turn Off, Dim
	Brighten	V(18) **V(10)** C(12) **V(2)** V(2) C(6) V(7) C(2)	
Philips Hue Bridge	Create Group	**V(7)** C(28) V(15) **V(9)** V(57) C(1) V(11) V(7) V(1) V(3) **V(36)**	Set Name, Create Group, Output Name, Delete Group
	Set Name	**V(57)** V(57) C(1) V(7) V(1) V(5) V(5) V(1) V(1) V(2) **V(21)**	
Broadlink Smart Plug	Turn Off	V(32) V(4) C(2) C(2) V(2) C(10) C(1) V(1) V(2) V(4) V(2) **V(26)**	Turn On, Turn Off
Xiaomi Mijia Smart Camera*	Turn On	C(12) V(4) V(9) V(7) V(21) C(9) V(4) V(1) V(2) V(3) C(5) **V(2)** C(4)	Turn On, Turn Off
Tplink Router	Add Forbidden Domain	C(2) C(10) C(12) C(13) C(10) V(15) C(11) C(6) C(3) **V(13)** C(14) **V(3)** C(2)	Add Forbidden Domain, Clear Forbidden Domains, Output Forbidden Domains

Proprietary Protocol Analysis

A **three-step evaluation process** derived from protocols' application

❶ Input message construction

❷ Protocol reverse engineering

❸ Application of inferred formats

Device	Behaviors of Input Messages	Message Format	# Triggered Behaviors
Yeelight LED Screen Light Bar Pro	Turn On	V(18) **V(9)** C(13) **V(2)** V(3) V(6) V(7) C(2)	Turn On, Brighten, Turn Off, Dim
	Brighten	V(18) **V(10)** C(12) **V(2)** V(2) C(6) V(7) C(2)	
Philips Hue Bridge	Create Group	**V(7)** C(28) V(15) **V(9)** V(57) C(1) V(11) V(7) V(1) V(3) **V(36)**	Set Name, Create Group, Output Name, Delete Group
	Set Name	**V(57)** V(57) C(1) V(7) V(1) V(5) V(5) V(1) V(1) V(2) **V(21)**	
Broadlink Smart Plug	Turn Off	V(32) V(4) C(2) C(2) V(2) C(10) C(1) V(1) V(2) V(4) V(2) **V(26)**	Turn On, Turn Off
Xiaomi Mijia Smart Camera*	Turn On	C(12) V(4) V(9) V(7) V(21) C(9) V(4) V(1) V(2) V(3) C(5) **V(2)** C(4)	Turn On, Turn Off
Tplink Router	Add Forbidden Domain	C(2) C(10) C(12) C(13) C(10) V(15) C(11) C(6) C(3) **V(13)** C(14) **V(3)** C(2)	Add Forbidden Domain, Clear Forbidden Domains, Output Forbidden Domains

Proprietary Protocol Analysis

A three-step evaluation process derived from protocols' application

❶ Input message construction

❷ Protocol reverse engineering

❸ Application of inferred formats

Device	Behaviors of Input Messages	Message Format	# Triggered Behaviors
Yeelight LED Screen Light Bar Pro	Turn On	V(18) **V(9)** C(13) **V(2)** V(3) V(6) V(7) C(2)	Turn On, Brighten, Turn Off, Dim
	Brighten	V(18) **V(10)** C(12) **V(2)** V(2) C(6) V(7) C(2)	
Philips Hue Bridge	Create Group	**V(7)** C(28) V(15) **V(9)** V(57) C(1) V(11) V(7) V(1) V(3) **V(36)**	Set Name, Create Group, Output Name, Delete Group
	Set Name	**V(57)** V(57) C(1) V(7) V(1) V(5) V(5) V(1) V(1) V(2) **V(21)**	
Broadlink Smart Plug	Turn Off	V(32) V(4) C(2) C(2) V(2) C(10) C(1) V(1) V(2) V(4) V(2) **V(26)**	Turn On, Turn Off
Xiaomi Mijia Smart Camera*	Turn On	C(12) V(4) V(9) V(7) V(21) C(9) V(4) V(1) V(2) V(3) C(5) **V(2)** C(4)	Turn On, Turn Off
Tplink Router	Add Forbidden Domain	C(2) C(10) C(12) C(13) C(10) V(15) C(11) C(6) C(3) **V(13)** C(14) **V(3)** C(2)	Add Forbidden Domain, Clear Forbidden Domains, Output Forbidden Domains

Based on formats inferred from the **original traffic with 7 behavior types**, the newly generated messages can **trigger 15 different behaviors** on the selected devices

Summary

- DynPRE exploits the server's interactive capability for protocol reverse engineering
- DynPRE supports adaptive message rewriting to allow proper interaction with the server
- DynPRE applies an intelligent request crafting method to obtain semantic information and supplementary samples for analysis
- DynPRE outperforms the state-of-the-art and proves effective in real-world applications

luozx19@mails.tsinghua.edu.cn