

# Informal specification-based performance evaluation of security protocols

---

Bela Genge<sup>1</sup>, Piroska Haller<sup>1</sup>, Iosif Ignat<sup>2</sup>, Ovidiu Ratoi<sup>1</sup>

<sup>1</sup>"Petru Maior" University of Târgu Mureş, Romania  
{bgenge, phaller, oratoi}@upm.ro

<sup>2</sup>Technical University of Cluj Napoca, Romania  
Iosif.Ignat@cs.utcluj.ro

# Presentation outline

---

- Performance evaluation context
- Protocol model
- Performance model
- Model validation
- Implementation
- Conclusions

# Main goals

---

- Constructing and accessing composed resources
  - Functionality composition
  - Security protocol composition:
    - Precondition effect composition
    - Term composition
  - Distribution and execution of composed protocols

# Security protocol composition

---

- Goal: create new protocols based on several smaller protocols
- Process:
  - Precondition-effect composition
  - Term (message component) composition
- Open problem:
  - Compose protocols with equal effects
  - Solutions:
    - Simply chose one protocol
    - Chose protocol with a reduced number of encryption/decryption requirements
    - Chose a protocol based on the protocol performances

# Performance evaluation related problems

---

- Regular performance evaluation:
  - Chose several cryptographic libraries (e.g. Cryptlib, OpenSSL, Crypto++)
  - Chose hardware, operating system, ...
  - Implement protocols
  - Execute protocols and measure performance
- Shortcomings:
  - User intervention required
  - Detailed aspects related to implementation must be known
  - Can not be used as an on-line procedure

# Performance evaluation related problems

- continued -

---

- ❑ On-line performance evaluation environment:
  - Unknown information:
    - ❑ Algorithms, keys, encryption/decryption modes (e.g. CBC, ECB, ...)
    - ❑ Exact message size
    - ❑ Cryptographic library
  - Known information:
    - ❑ Algorithm types
    - ❑ Message component types
    - ❑ Message structures

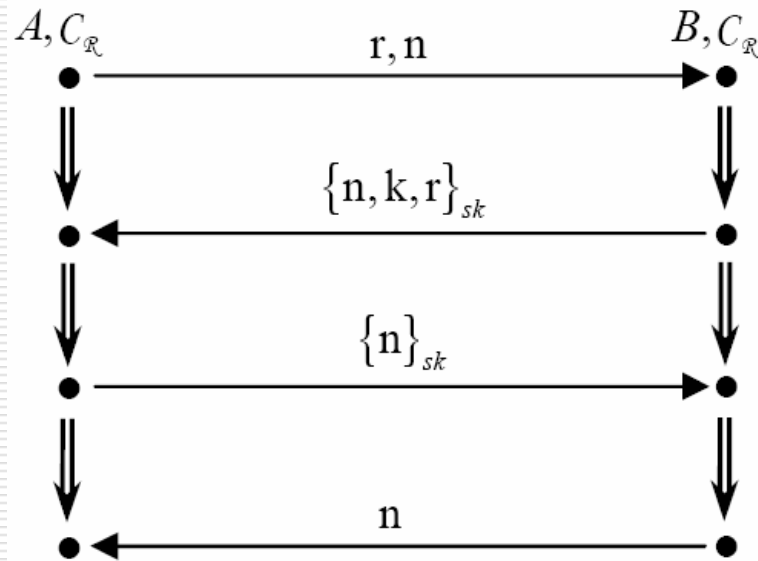
# Proposed solution

---

- Comparative performance evaluation:
  - Exact protocol performance is not required
  - Comparatively analyze the protocol performances
- Protocol performance evaluation:
  - Based on the performance of cryptographic algorithms
  - Model cryptographic operations
  - Construct a general performance model of cryptographic algorithms

# Protocol model

- Strand-based canonical model
- Model cryptographic operations



$$S_{tA} = \langle C_{\mathcal{R}}, A, \langle +(r, n), -\{n, k, r\}_{sk}, +\{n\}_{sk}, -n \rangle \rangle$$

$$S_{tB} = \langle C_{\mathcal{R}}, B, \langle -(r, n), +\{n, k, r\}_{sk}, -\{n\}_{sk}, +n \rangle \rangle$$

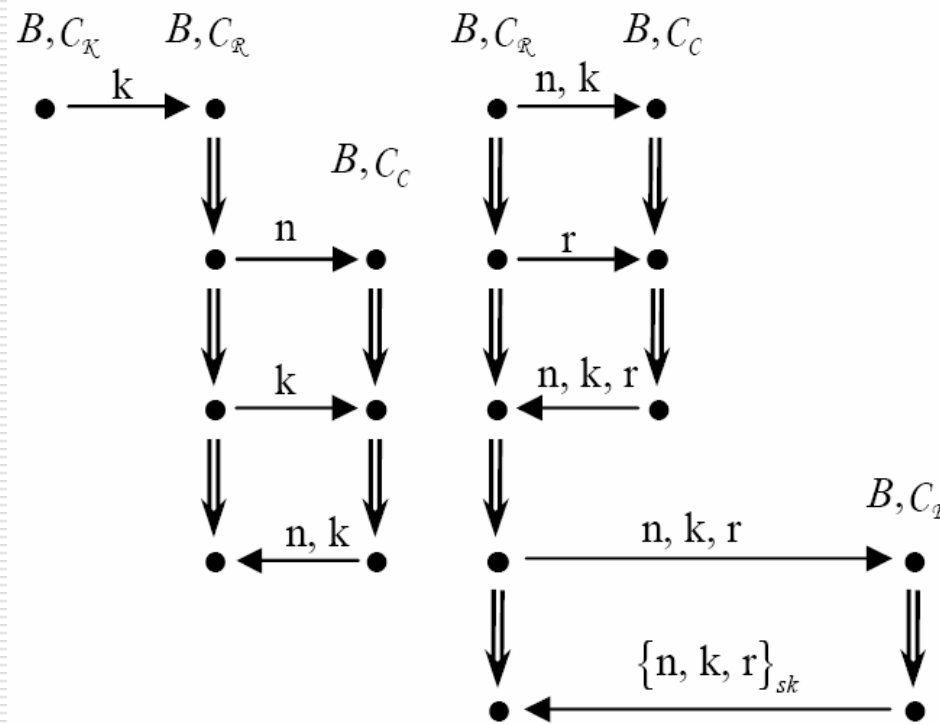
Mathematical representation

Graphical representation

# Protocol model

## - continued -

- Representing protocol operations using strand classifiers



# Performance evaluation functions

---

- Add functions to denote the cost of each operation:

$$f_{sk}, f_{pk}, f_h, f_{kg}, f_{ng}, f_s, f_p : \mathbb{R}^+ \rightarrow \mathbb{R}^+$$

- Add functions to denote the resulting size of each operation:

$$\lambda_S, \lambda_A, \lambda_H : \mathcal{T}_t \rightarrow \mathbb{R}^+$$

- Evaluate the total cost

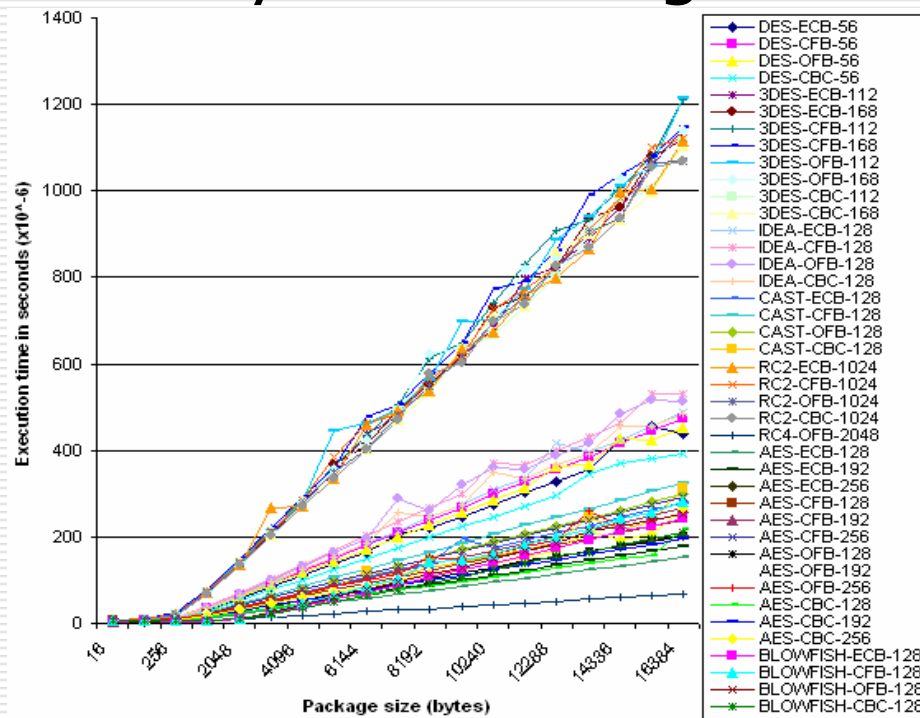
# Function values through exhaustive performance evaluation

---

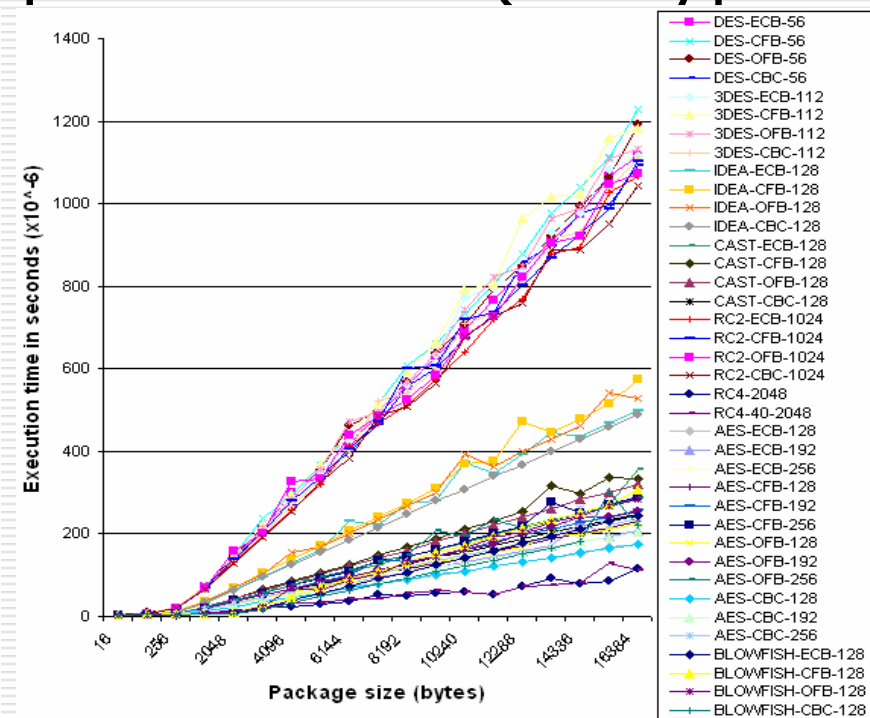
- ❑ Exhaustive performance evaluation
- ❑ Cryptographic libraries: Cryptlib and OpenSSL
- ❑ Analyze the performance of all:
  - Supported cryptographic algorithms
  - Supported cryptographic keys
  - Supported cryptographic modes

# Exhaustive performance analysis

## □ Symmetric algorithm performance (encryption)



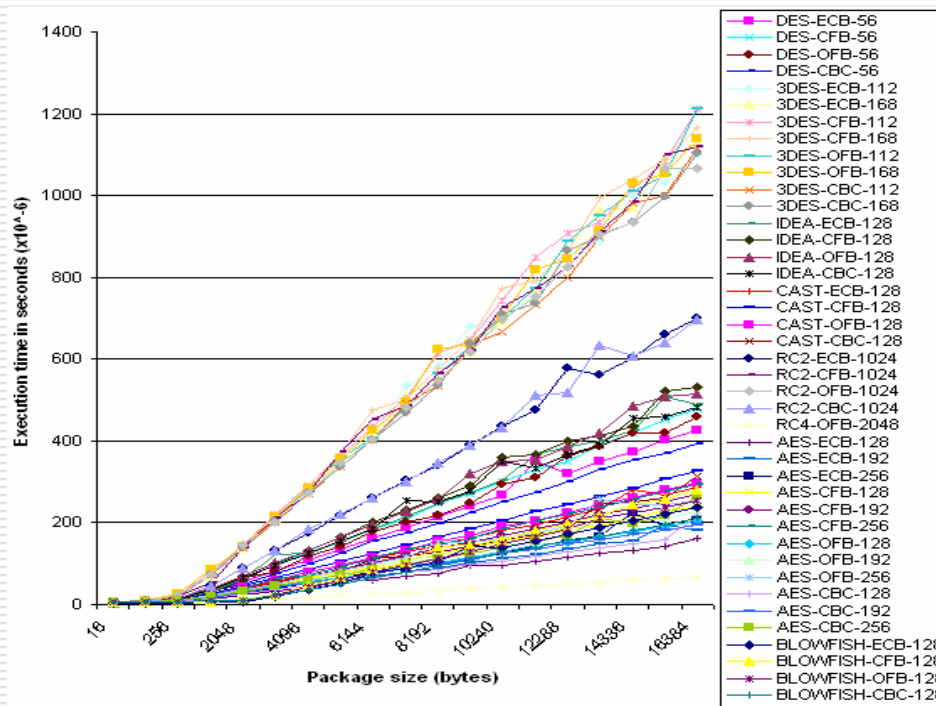
Cryptlib



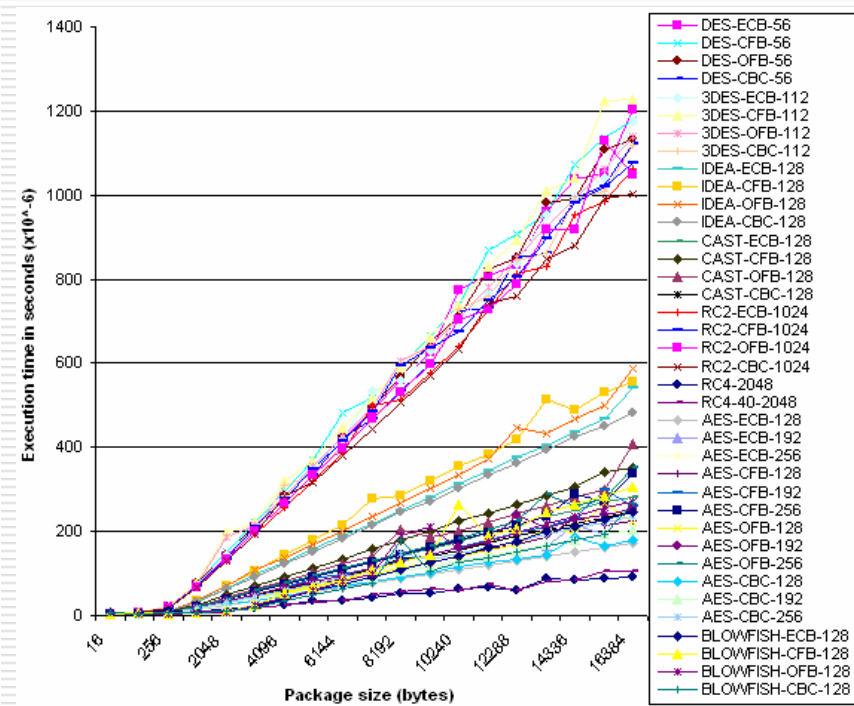
OpenSSL

# Exhaustive performance analysis - continued -

## □ Symmetric algorithm performance (decryption)



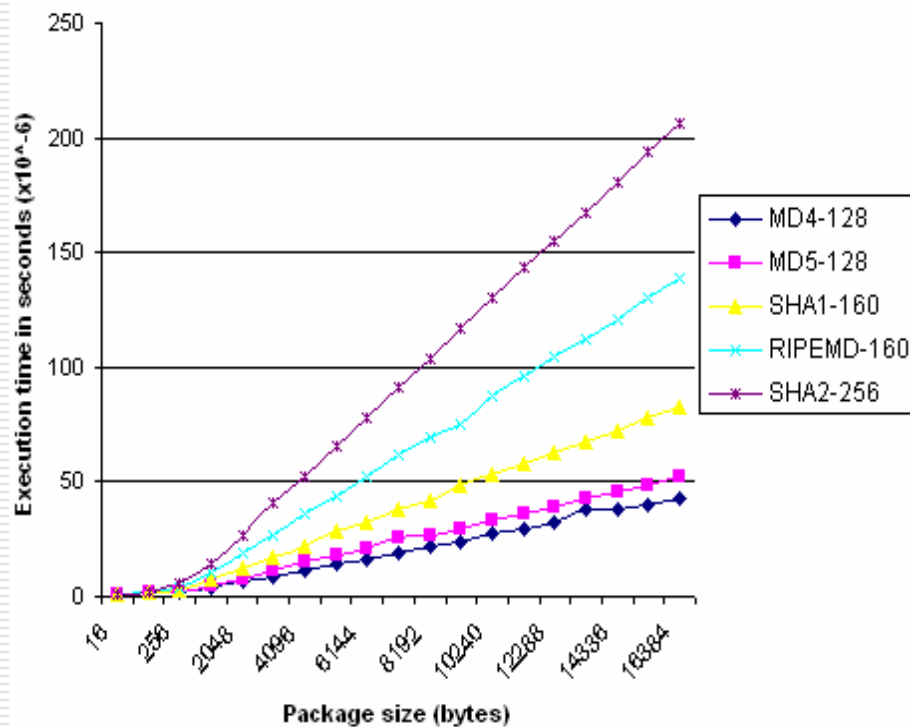
Cryptlib



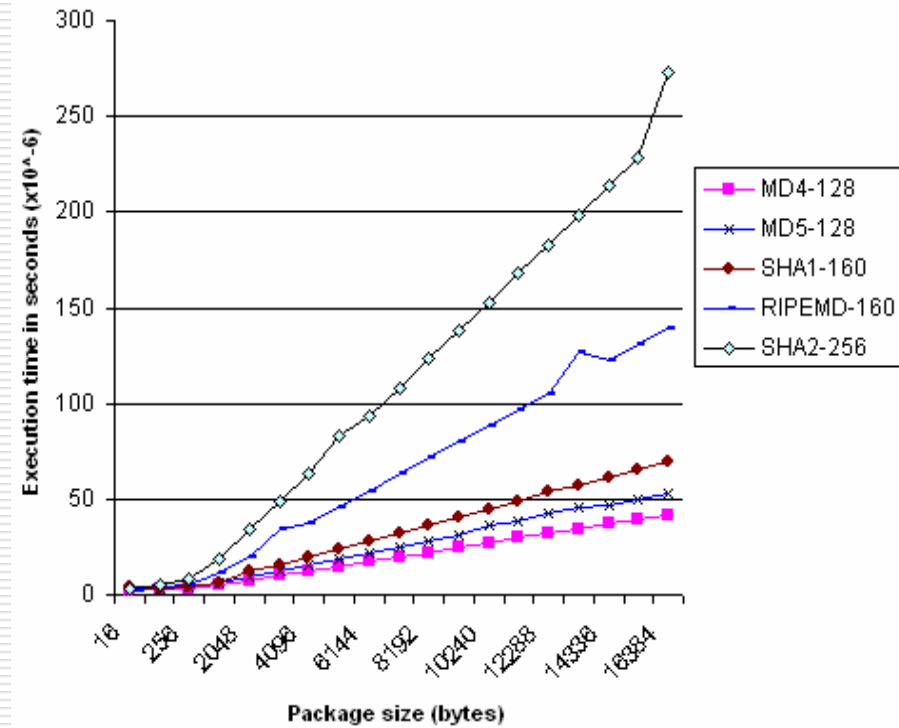
OpenSSL

# Exhaustive performance analysis - continued -

## □ Hash algorithm performance



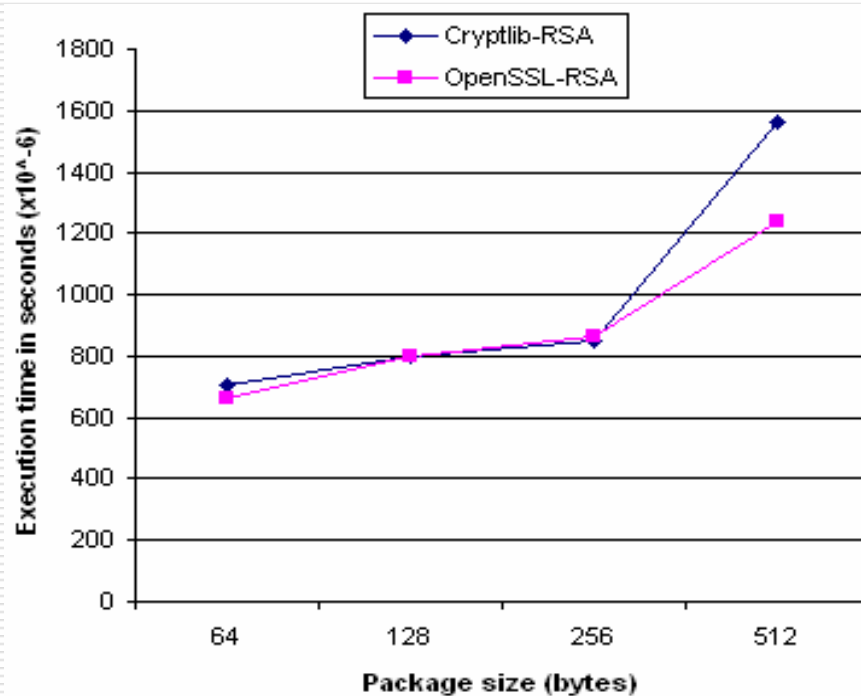
Cryptlib



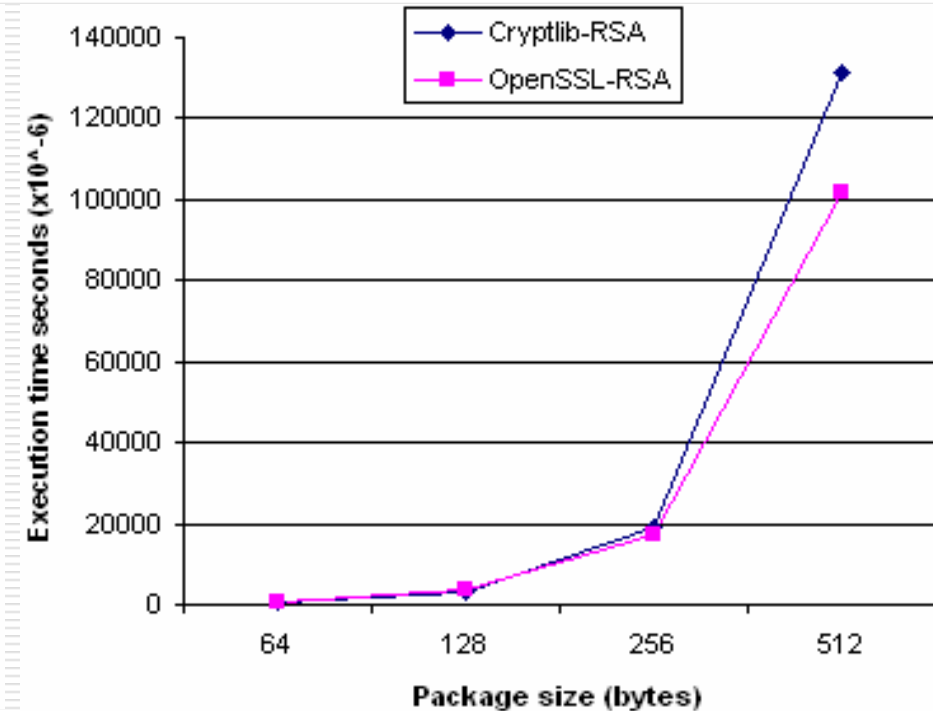
OpenSSL

# Exhaustive performance analysis - continued -

## □ Asymmetric algorithms



Encryption



Decryption

# General performance model

---

- Polynomial function:

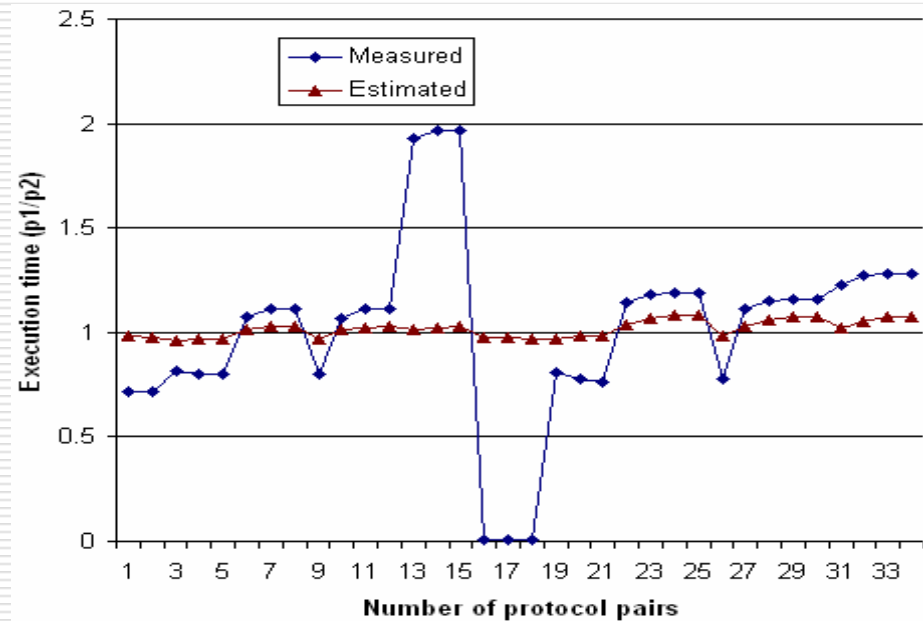
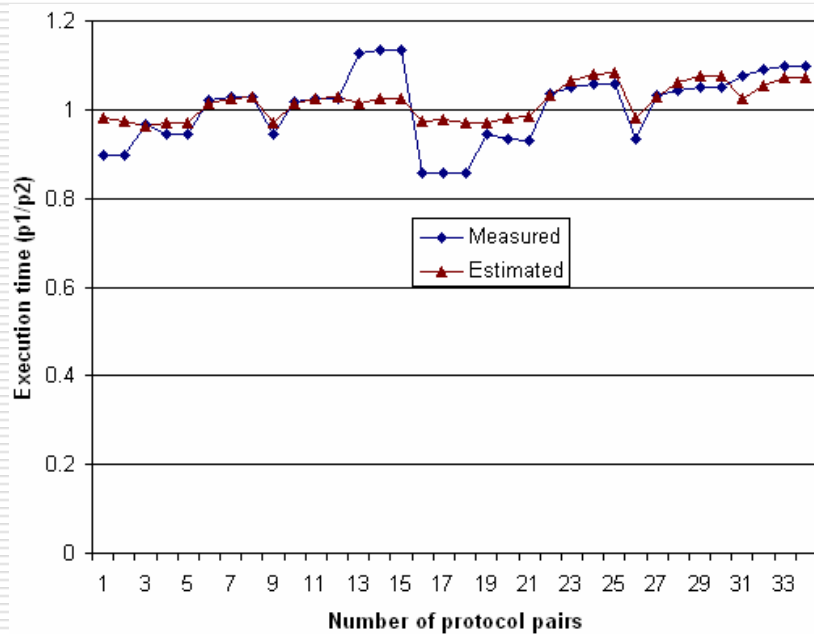
$$f(x) = \alpha_4 x^3 + \alpha_3 x^2 + \alpha_2 x + \alpha_1$$

- For each algorithm type, the following equations must be solved:

$$\left\{ \begin{array}{l} \frac{\partial R}{\partial \alpha_1} = -2 \sum_{i=1}^n (y_i - (\alpha_4 x_i^3 + \alpha_3 x_i^2 + \alpha_2 x_i + \alpha_1)) = 0 \\ \frac{\partial R}{\partial \alpha_2} = -2 x_i \sum_{i=1}^n (y_i - (\alpha_4 x_i^3 + \alpha_3 x_i^2 + \alpha_2 x_i + \alpha_1)) = 0 \\ \frac{\partial R}{\partial \alpha_3} = -2 x_i^2 \sum_{i=1}^n (y_i - (\alpha_4 x_i^3 + \alpha_3 x_i^2 + \alpha_2 x_i + \alpha_1)) = 0 \\ \frac{\partial R}{\partial \alpha_4} = -2 x_i^3 \sum_{i=1}^n (y_i - (\alpha_4 x_i^3 + \alpha_3 x_i^2 + \alpha_2 x_i + \alpha_1)) = 0 \end{array} \right.$$

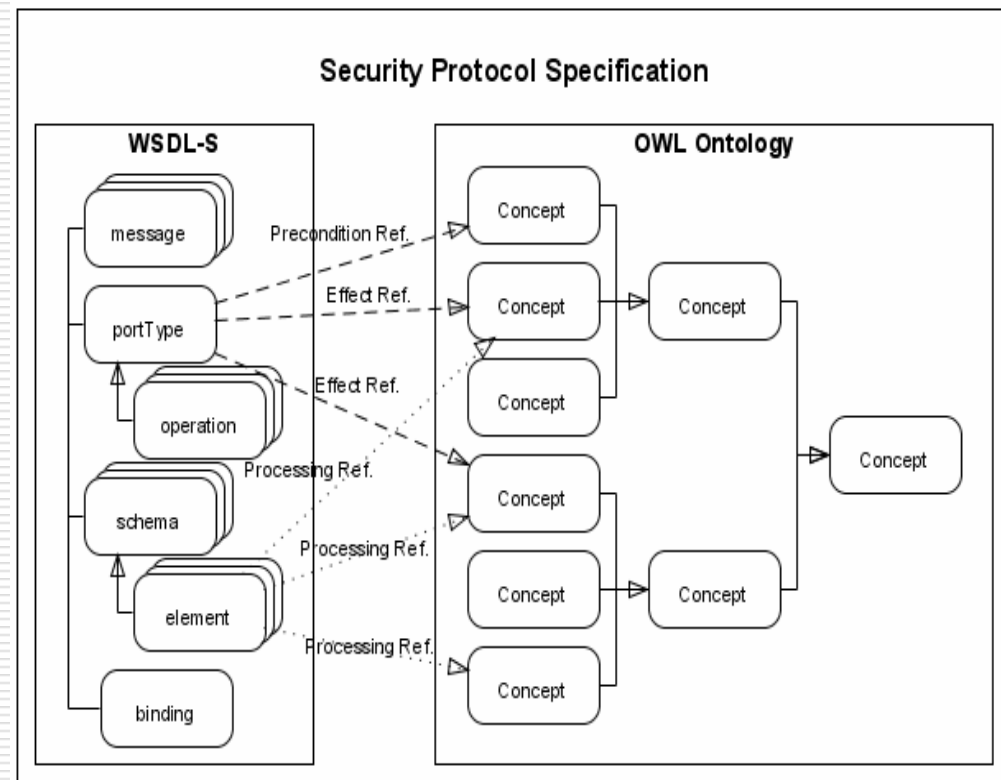
# Model validation

- Generated several thousand security protocols
- Analyzed the measured and predicted comparative performance



# Security protocol specification

- Detailed description of message constructions and processing
- Technologies:
  - WSDL-S
  - OWL



# Sequential specification

---

- Contains information related to:
  - Message transport
  - Message sequence
  - Message component annotations
  - Protocol precondition and effect
- Implemented as: WSDL-S

# Sequential specification

## - continued -

---

### □ Example message specification

```
<xsd:element name="Msg">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="UserName" type="xsd:string"
wssem:modelReference="http://.../SecurityProtocol.owl#A"/>
      <xsd:element name="RandomNumber" type="xsd:base64Binary"
wssem:modelReference="http://.../SecurityProtocol.owl#Na"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

# Semantic specification

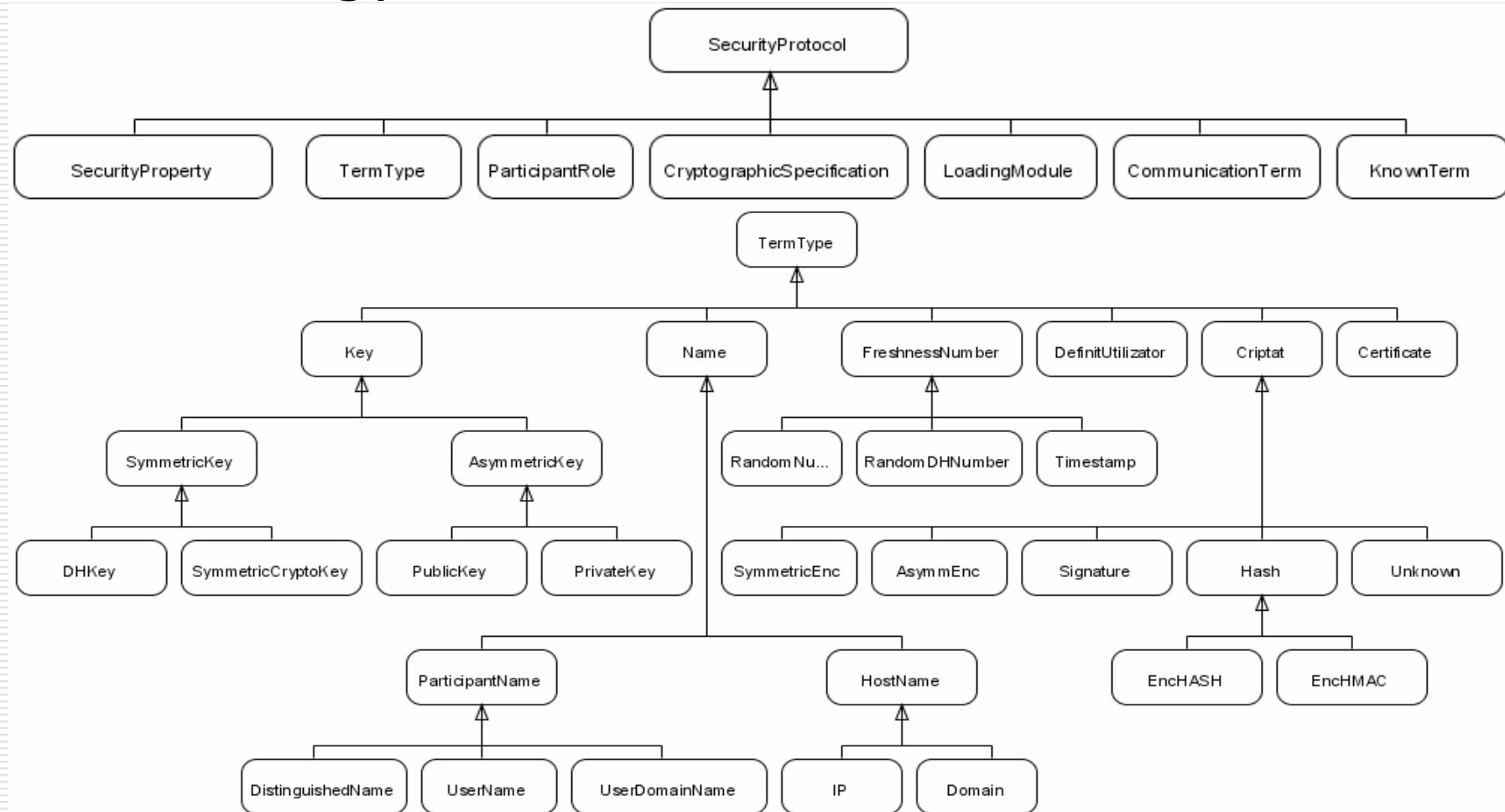
---

- Contains information related to:
  - Message construction:
    - Encryption algorithm, key
    - Message component location
    - Message component type
  - Message processing:
    - Decryption algorithm, key
    - Message component verification

# Semantic specification

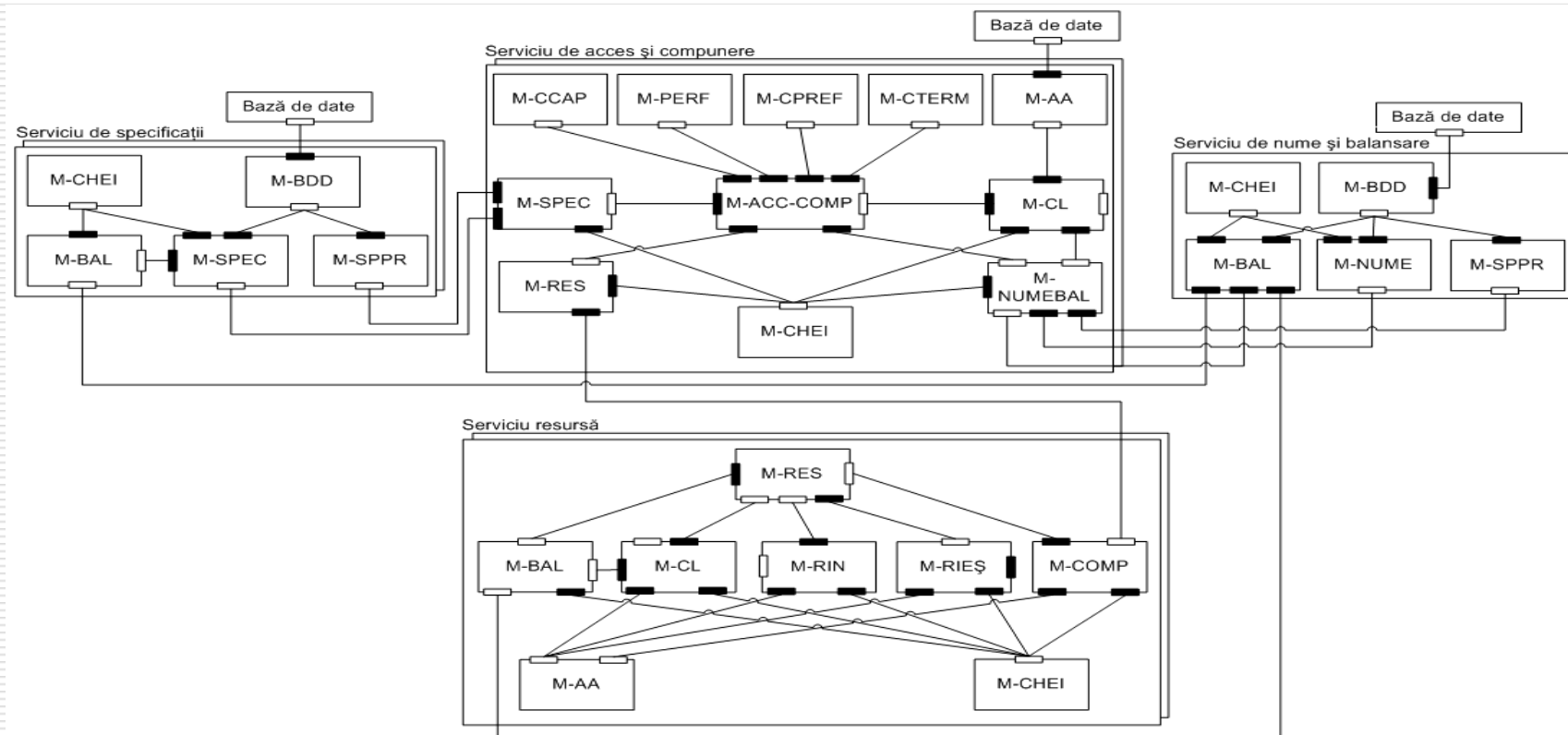
## - continued -

### □ Ontology-view



# Implementation platform

## □ Service-oriented architecture



# Conclusions and future work

---

## □ Conclusions:

- We have developed a comparative performance evaluation method
- We have integrated the proposed method in the composition process

## □ Future work:

- Develop an automated specification development tool



Thanks for your attention!