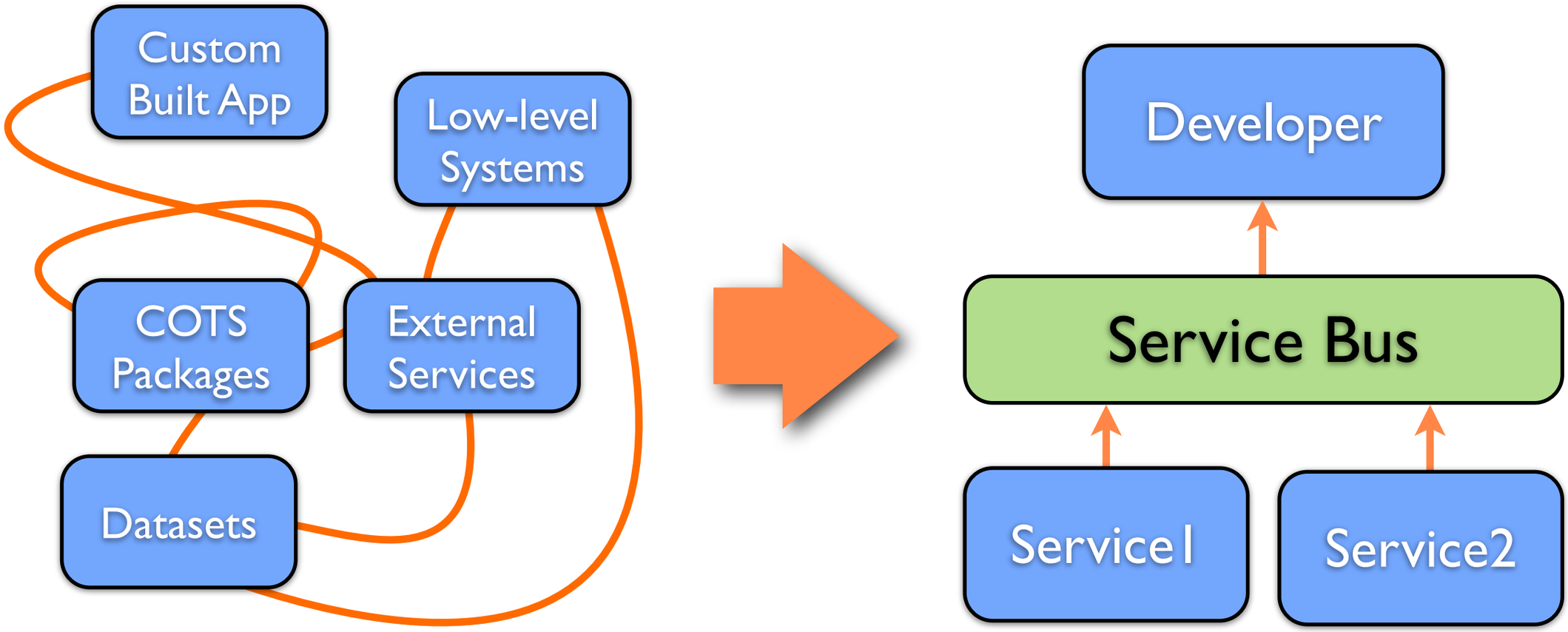
A Risk-Evaluation Assisted System for Service Selection

Ennan Zhai and Liang Gu

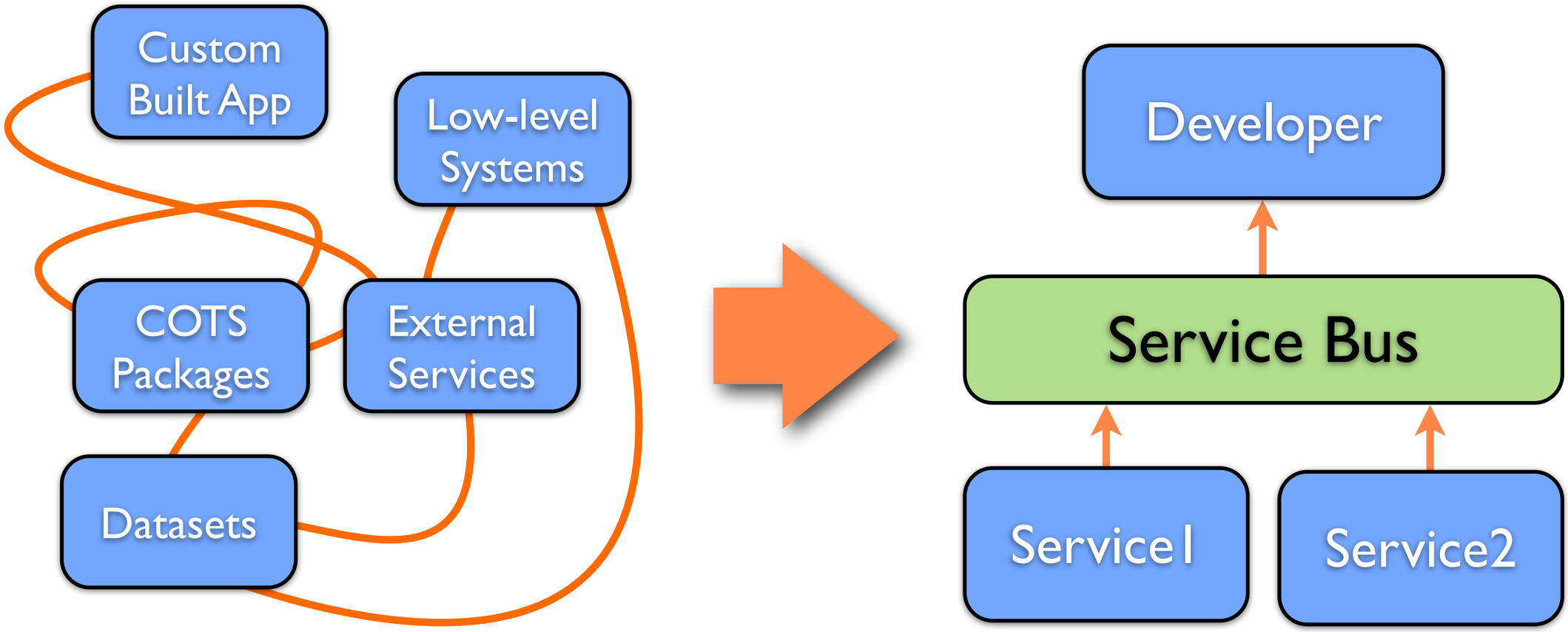
Yale University

{firstname.lastname}@yale.edu

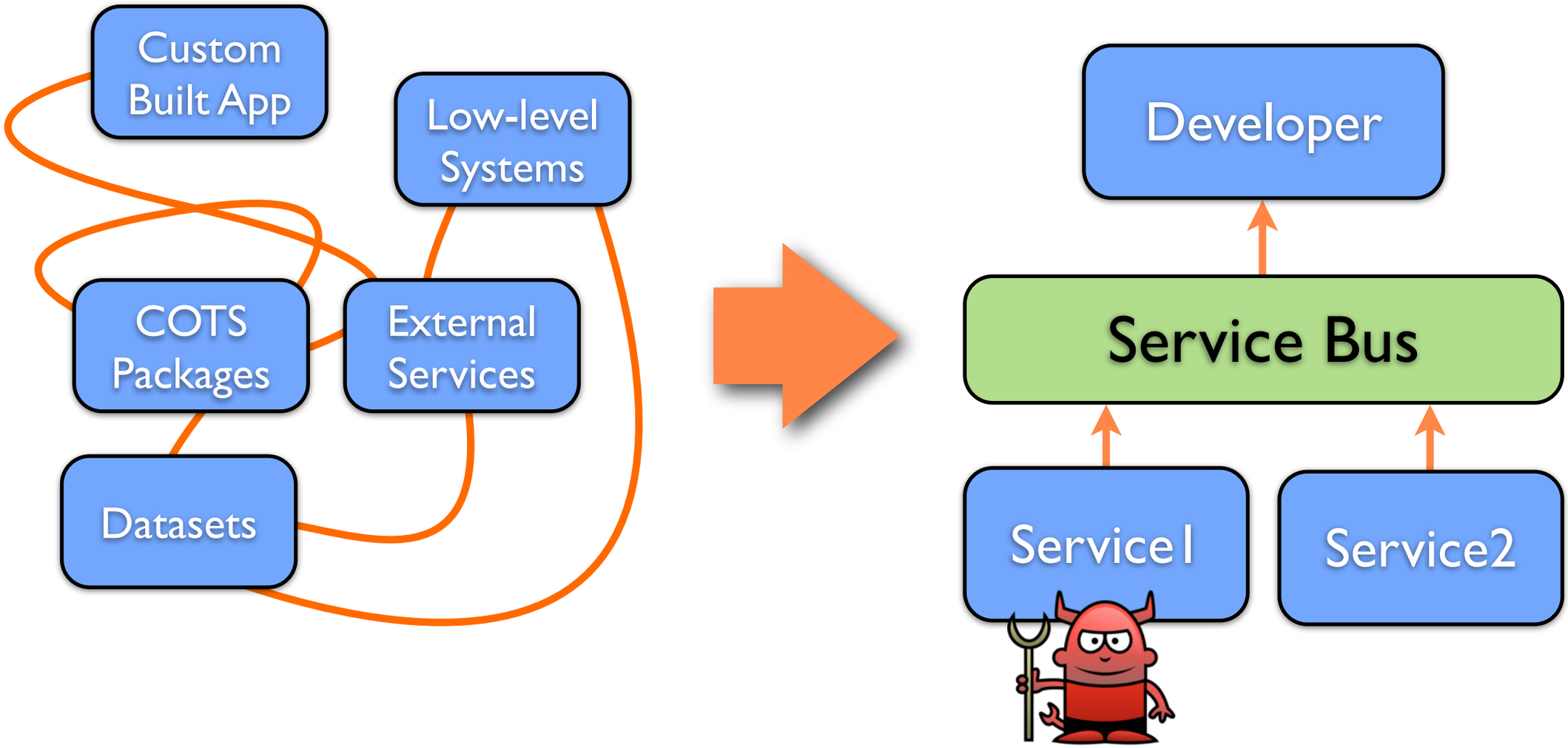
Service Oriented Architecture



Unexpected Risks



Unexpected Risks



Example

Video App

Example

Video App

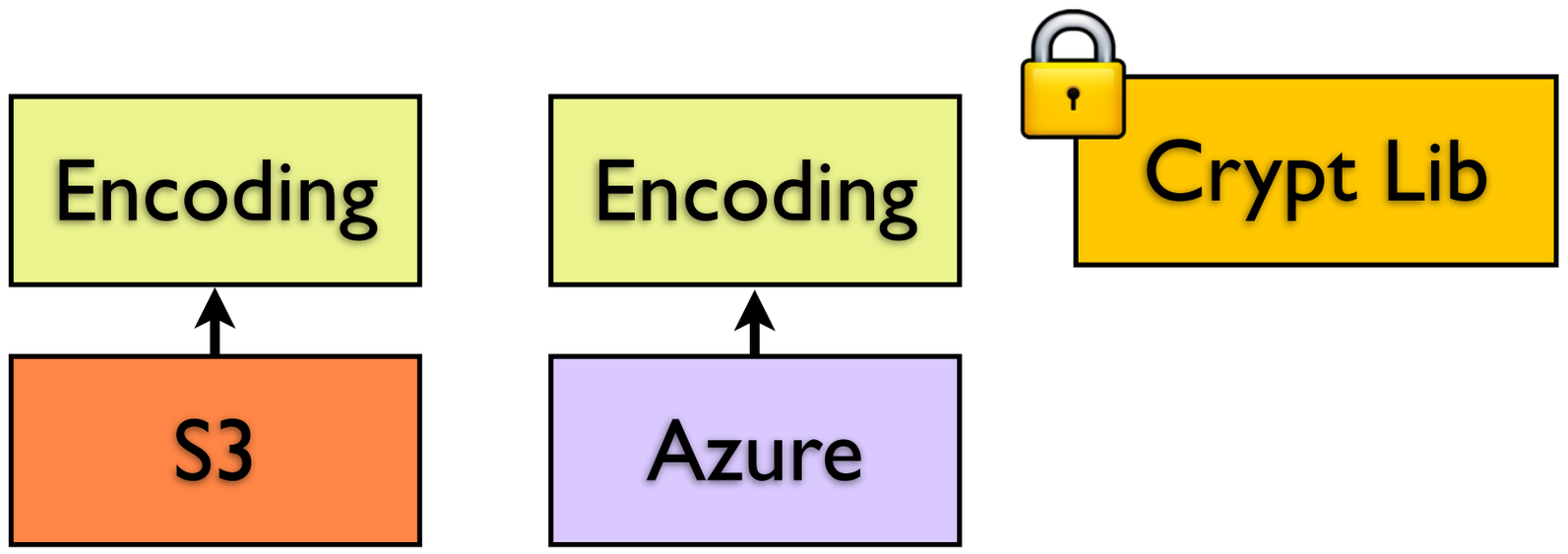
Crypt Lib

S3

Azure

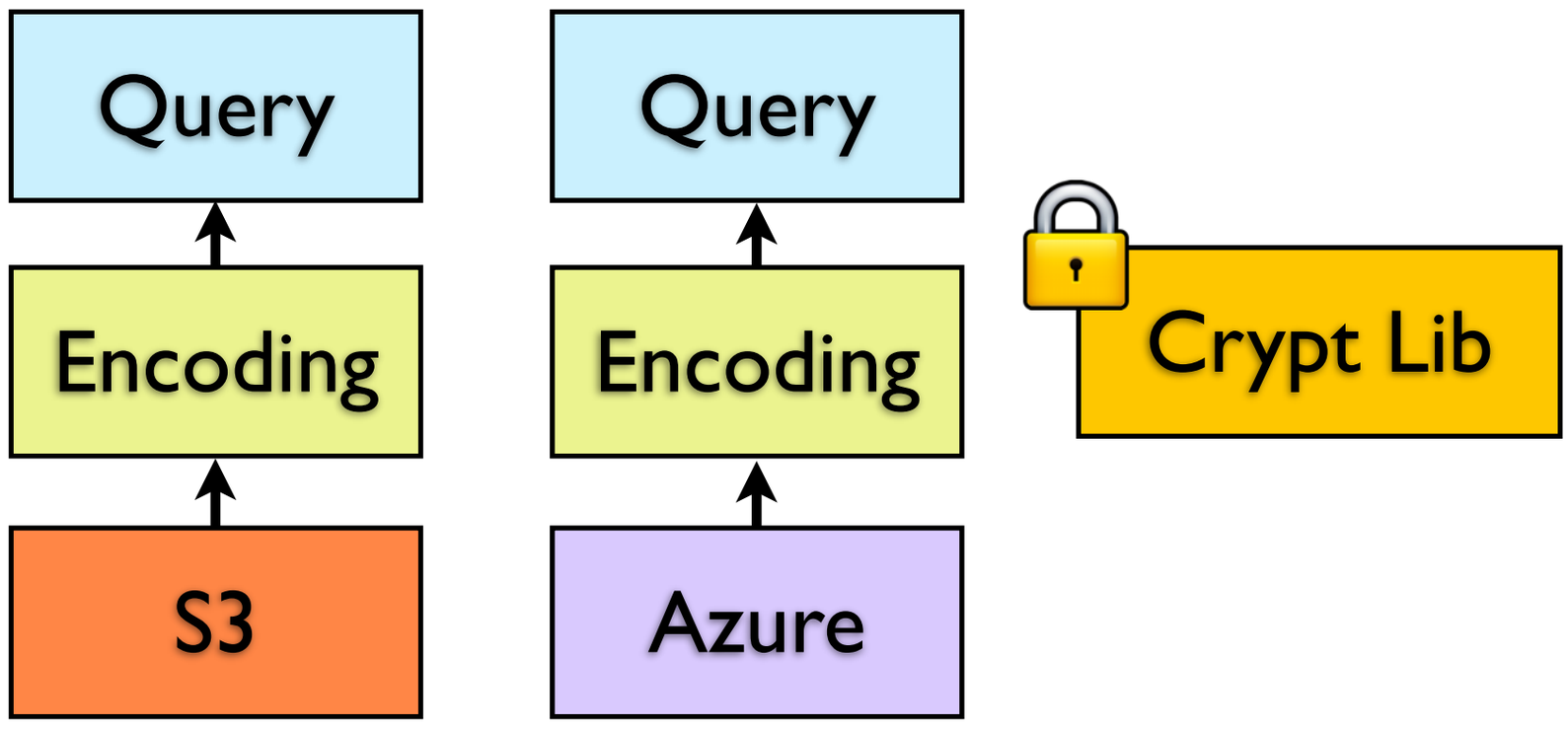
Example

Video App

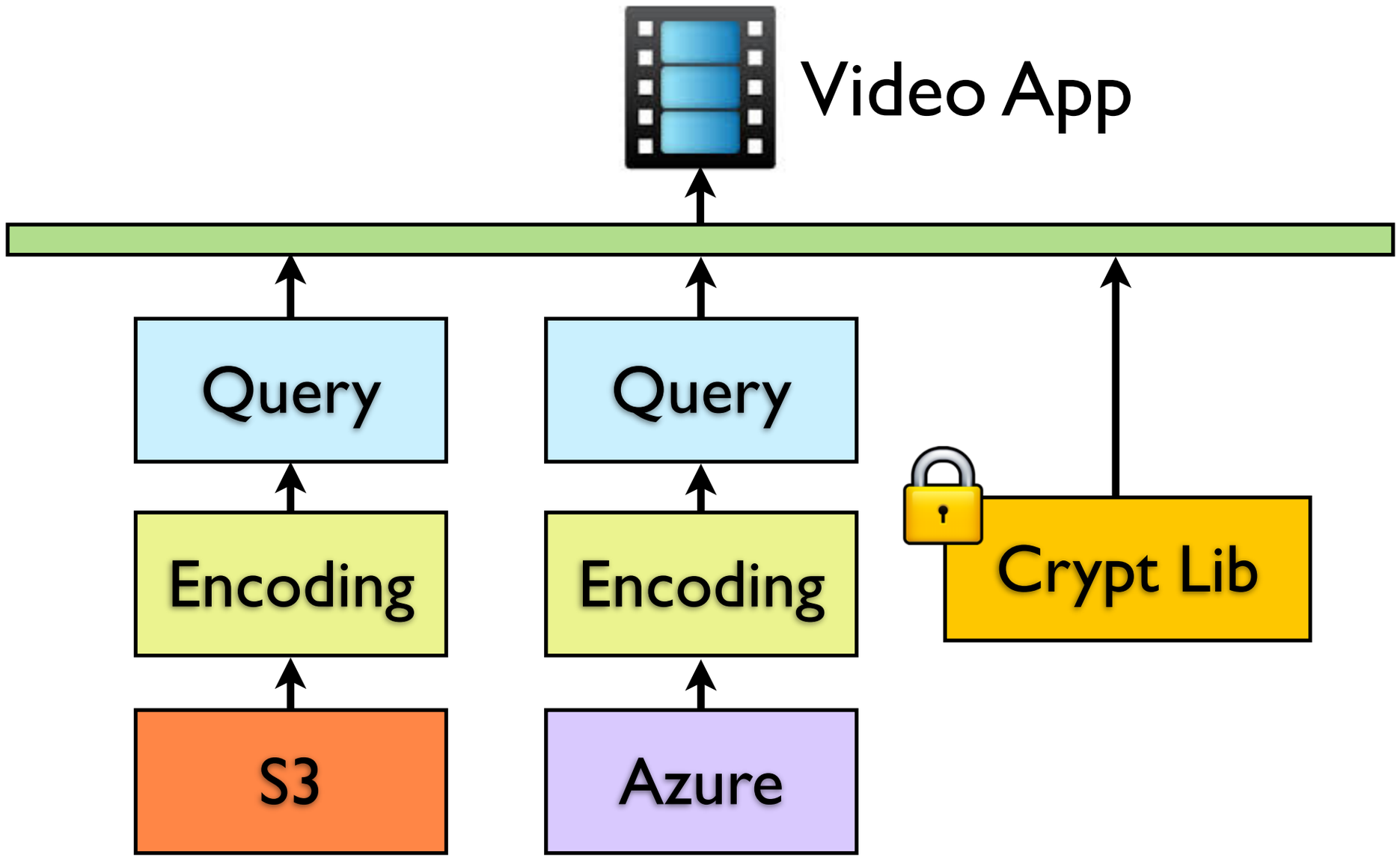


Example

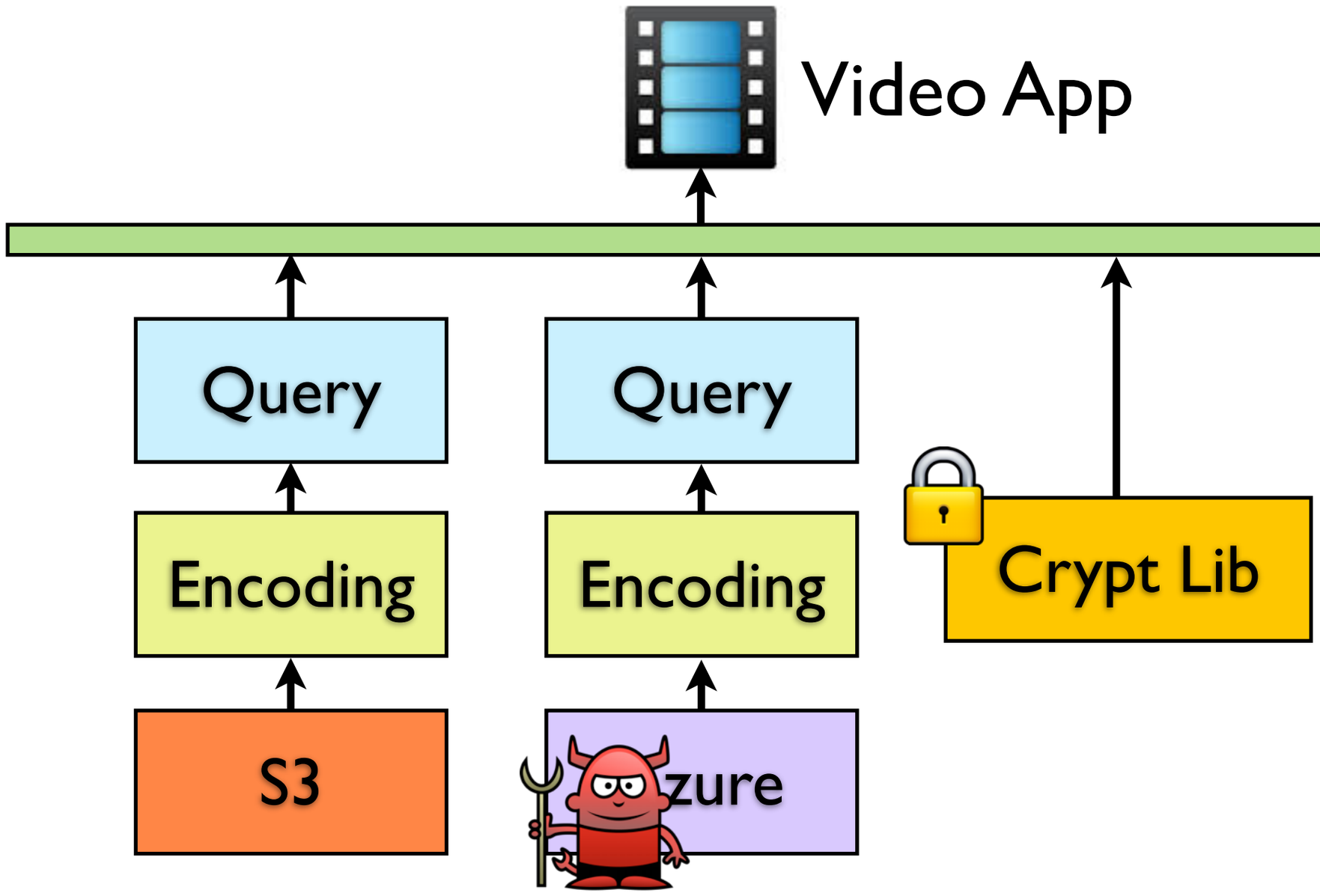
Video App



Example



Example



What leads to the problem?

What leads to the problem?

- Lack of systematic approach to avoid these bugs.

What leads to the problem?

- Lack of systematic approach to avoid these bugs.
- No service provider is wiling to share the information.

Target

- Can we reduce such risk before the service selection of application developers?

Solution:
Risk-Based Service Selection

Solution:
Risk-Based Service Selection

- Select services based on requirements.

Solution:
Risk-Based Service Selection

- Select services based on requirements.
- At-best effort to avoid potential bugs within services.

Solution:
Risk-Based Service Selection

- Select services based on requirements.
- At-best effort to avoid potential bugs within services.
- Do not leak information of service providers.

Road-Map

- Motivating Example
- REaaS Design
- Evaluation

Road-Map

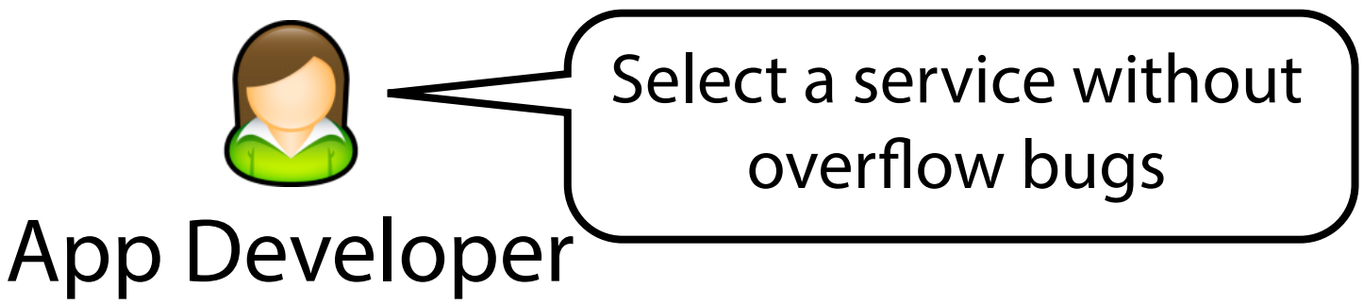
- Motivating Example
- REaaS Design
- Evaluation

Motivating Example

Motivating Example

App Developer

Motivating Example



Motivating Example

Motivating Example

App Developer

ESaaS

Service	Score
Service B	0.1
Service C	0.8
Service A	1.3

Road-Map

- Motivating Example
- REaaS Design
- Evaluation

Road-Map

- Motivating Example
- REaaS Design
- Evaluation

REaaS Workflow

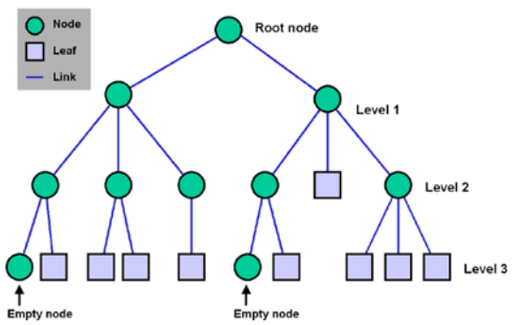
App Developer

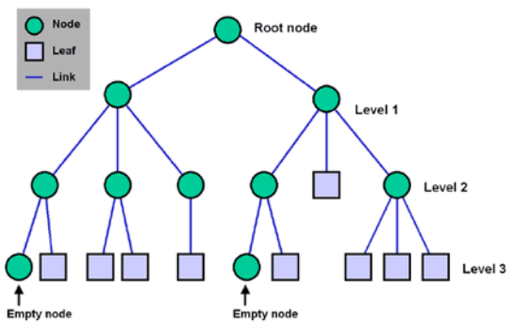
ESaaS

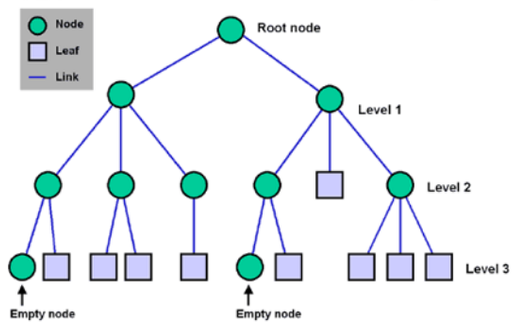
Step1: Service Registration

App Developer

ESaaS



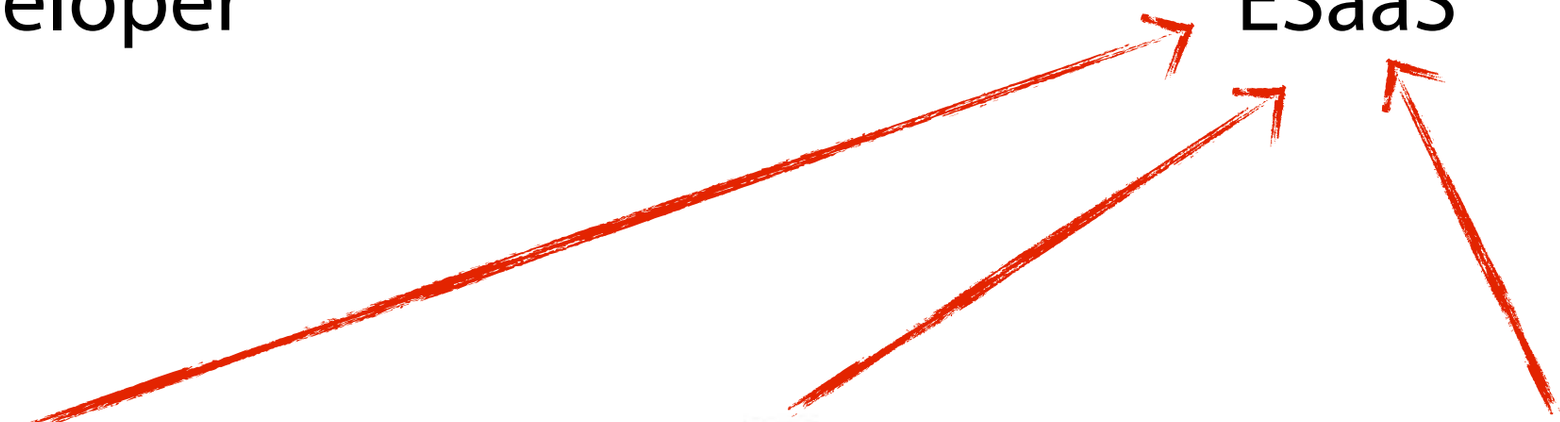


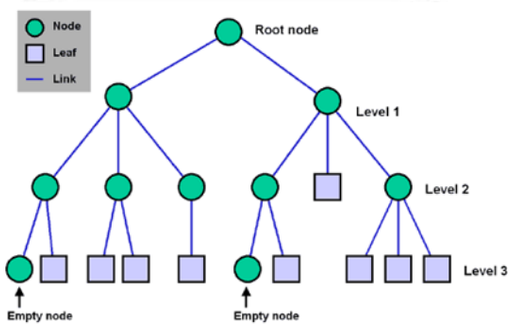


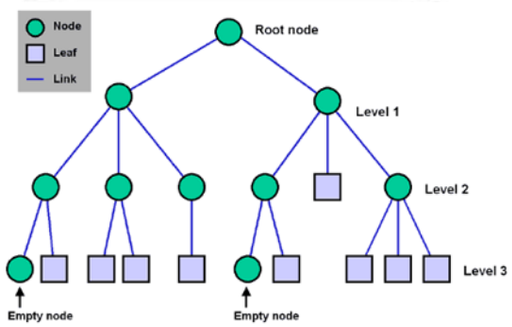
Step1: Service Registration

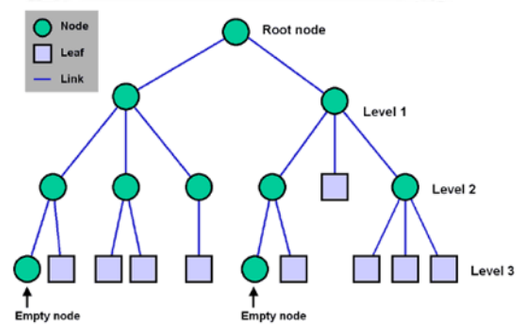
App Developer

ESaaS









Step2: Requirement Submission

Step3: Risk Evaluation

App Developer

ESaaS

Service	Score
Service B	0.1
Service C	0.8
Service A	1.3

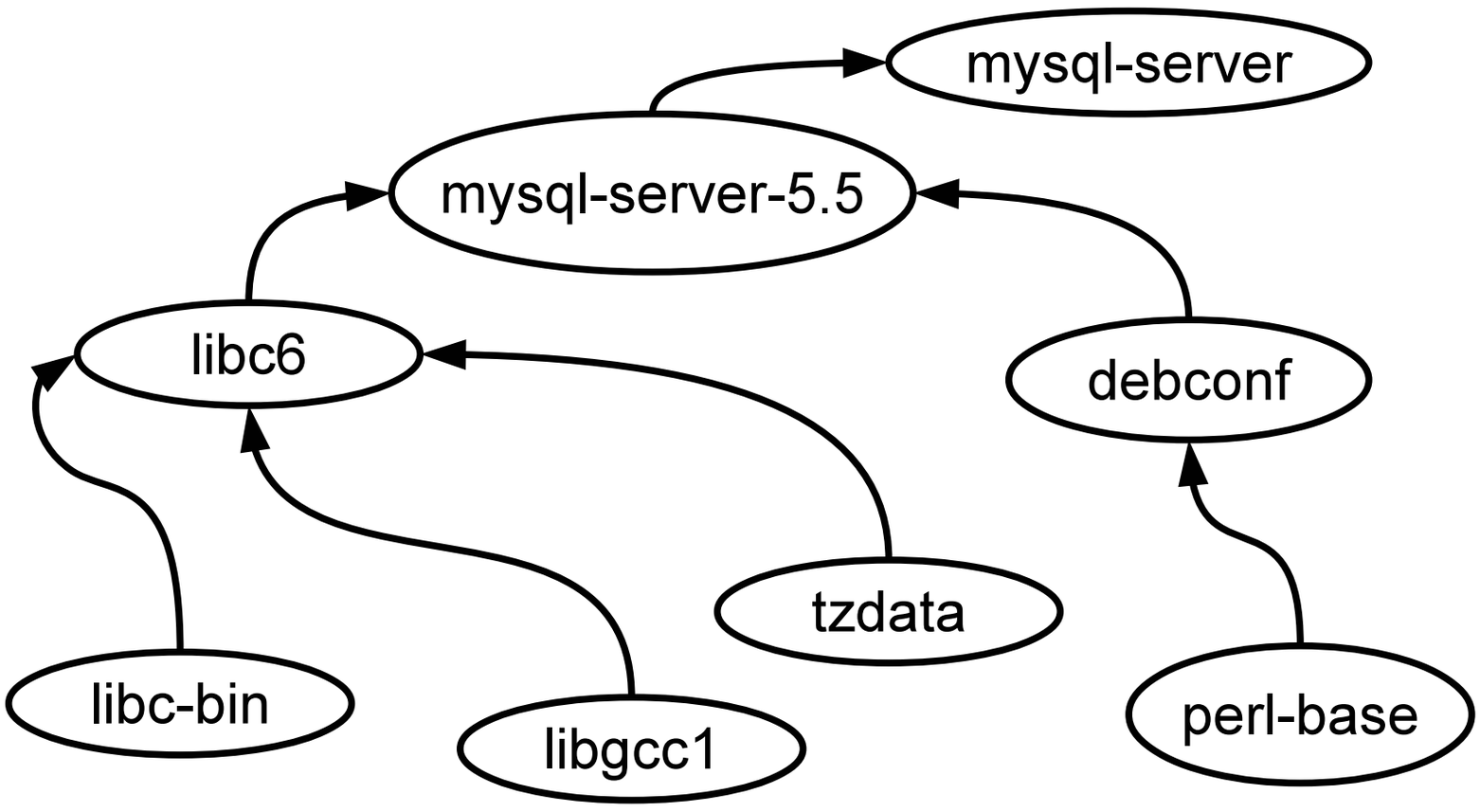
Step1: Service Registration

- We developed a tool automatically getting dependency.
- Running on service side and very fast.

Step1: Service Registration

```
mysql-server
  Depends: mysql-server-5.5
mysql-server-5.5
  Depends: libc6 (>= 2.14)
  Depends: debconf (>= 0.5)
debconf
  Depends: perl-base (>= 5.6.1-4)
libc6
  Depends: libc-bin (= 2.15-0ubuntu10)
  Depends: libgcc1
  Depends: tzdata
```

Step1: Service Registration



Step2: Requirement Submission

Step2: Requirement Submission

- Requirements include:
  - availability
  - integrity
  - confidentiality
- The options follow CVSS (an open bug DB)

Step3: Risk Evaluation

Step3: Risk Evaluation

$$
\begin{cases}
TD_i = \sum_{j=1}^{n} BS_{j(i)}/n \\
BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176 \\
Imp_{j(i)} = 10.41 \cdot ImpactLevel_{j(SecObj)}
\end{cases}
$$

Step3: Risk Evaluation

$$
\begin{cases}
TD_i = \sum_{j=1}^{n} BS_{j(i)}/n \\
BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176 \\
Imp_{j(i)} = 10.41 \cdot ImpactLevel_{j(SecObj)}
\end{cases}
$$

Risk score of service i

Step3: Risk Evaluation

$$
\begin{cases}
TD_i = \sum_{j=1}^{n} BS_{j(i)}/n \\
BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176 \\
Imp_{j(i)} = 10.41 \cdot ImpactLevel_{j(SecObj)}
\end{cases}
$$

Different bugs' impact under different objects

Step3: Risk Evaluation

$$\begin{cases} TD_i = \sum_{j=1}^{n} BS_{j(i)}/n \\ BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176 \\ Imp_{j(i)} = 10.41 \cdot ImpactLevel_{j(SecObj)} \end{cases}$$

Different bugs' impact under different objects

Gotten from CVSS + CVE

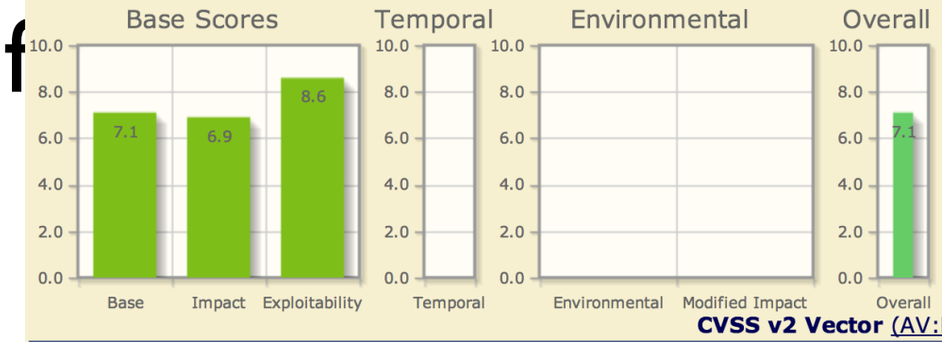
Step3: Risk Evaluation

$$\begin{cases} TD_i = \sum_{j=1}^{n} BS_{j(i)}/n \\ BS_{j(i)} = (0.4 \cdot Exp_{j(i)} + 0.6 \cdot Imp_{j(i)}) \cdot 1.176 \\ Imp_{j(i)} = 10.41 \cdot ImpactLevel_{j(SecObj)} \end{cases}$$

Dif nt objects

Common Vulnerability Scoring System Version 2 Calculator

This page shows the components of the CVSS score for example and allows you to refine the CVSS base score. Please read the CVSS standards guide to fully and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Sco



CVSS Base Score	**7.1**
Impact Subscore	6.9
Exploitability Subscore	8.6
CVSS Temporal Score	**Not Defined**
CVSS Environmental Score	**Not Defined**
Modified Impact Subscore	Not Defined
Overall CVSS Score	**7.1**

Show Equations

CVSS v2 Vector (AV:N/AC:M/Au:N/C:N/I:N/A:C)

Base Score Metrics

Exploitability Metrics

Access Vector (AV)*: Local (AV:L) | Adjacent Network (AV:A) | **Network (AV:N)**

Access Complexity (AC)*: High (AC:H) | **Medium (AC:M)** | Low (AC:L)

Authentication (Au)*: Multiple (Au:M) | Single (Au:S) | **None (Au:N)**

Impact Metrics

Confidentiality Impact (C)*: **None (C:N)** | Partial (C:P) | Complete (C:C)

Integrity Impact (I)*: **None (I:N)** | Partial (I:P) | Complete (I:C)

Availability Impact (A)*: None (A:N) | Partial (A:P) | **Complete (A:C)**

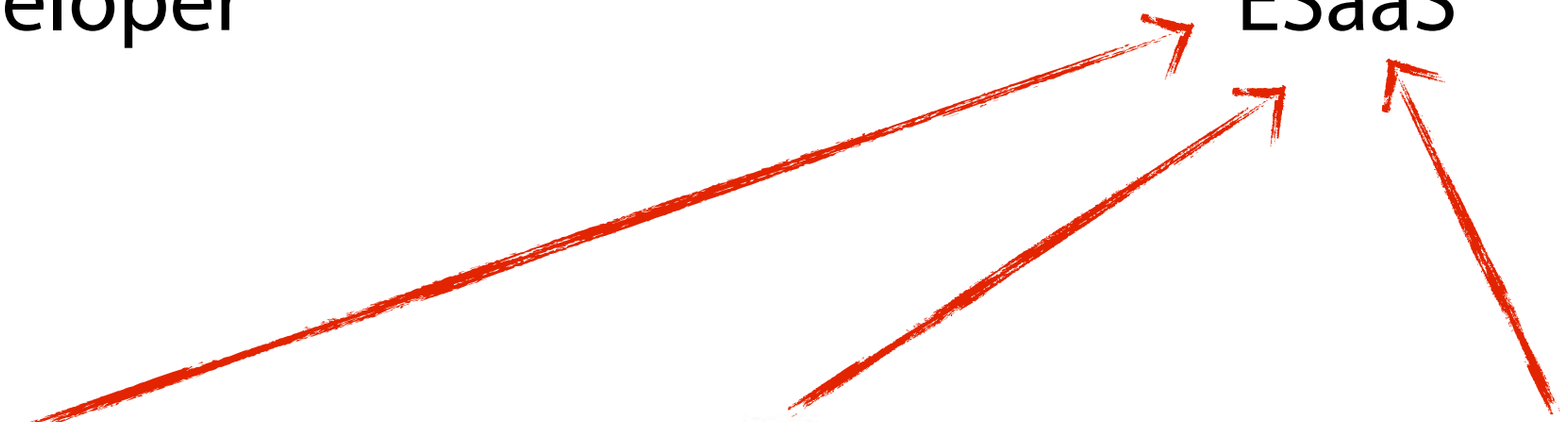
* - All base metrics are required to generate a base score.

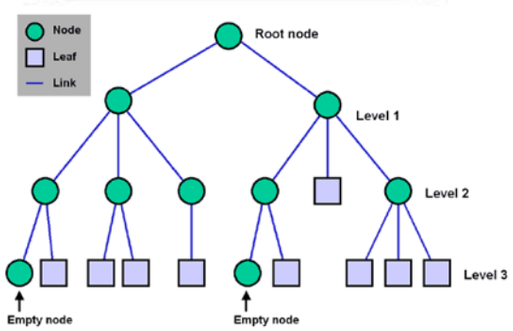
Trust?

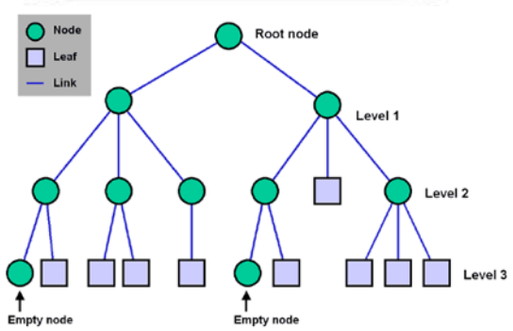
Trust?

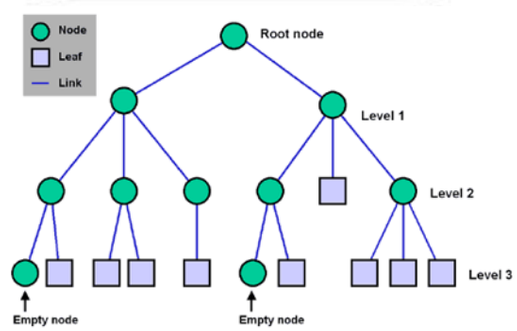
App Developer

ESaaS









Trust?

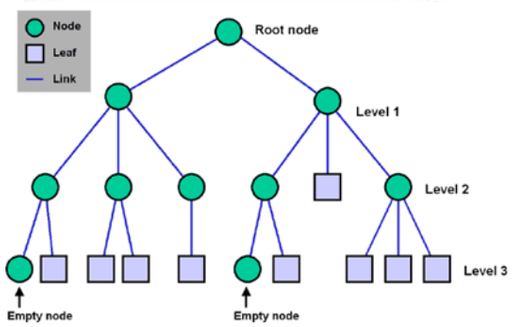
App Developer

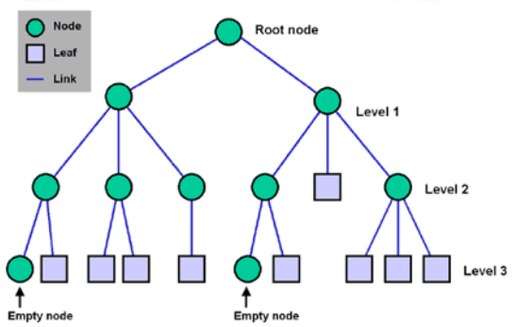
ESaaS

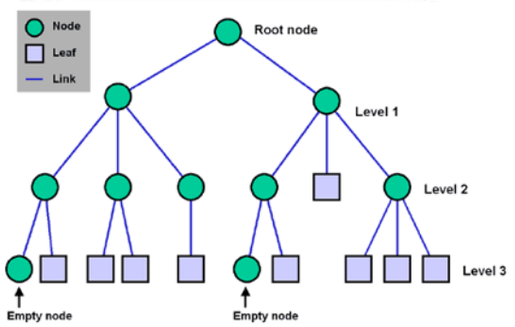
Service A

Service B

Service C





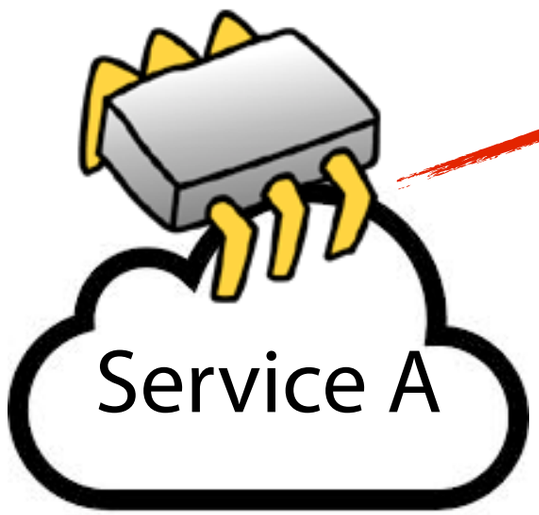


TPM-Based REaaS

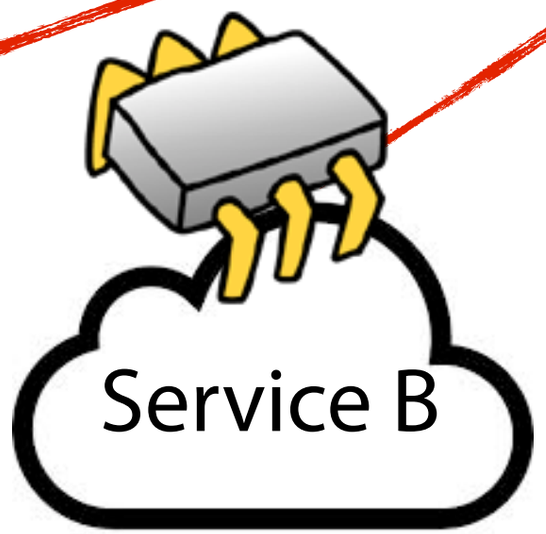
TPM-Based REaaS

App Developer

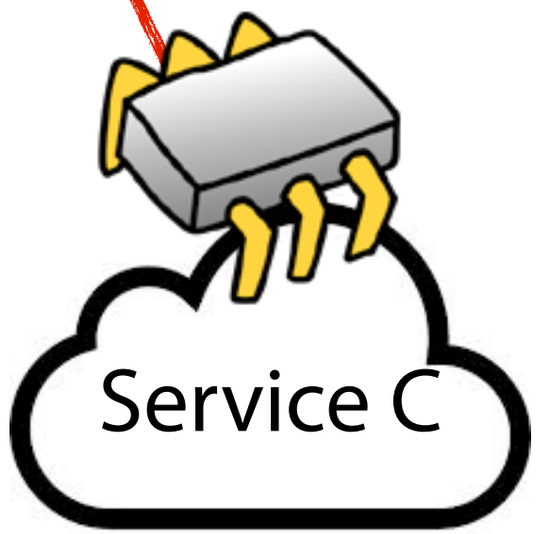
ESaaS



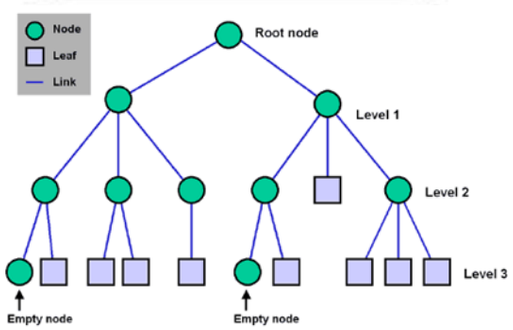
Service A

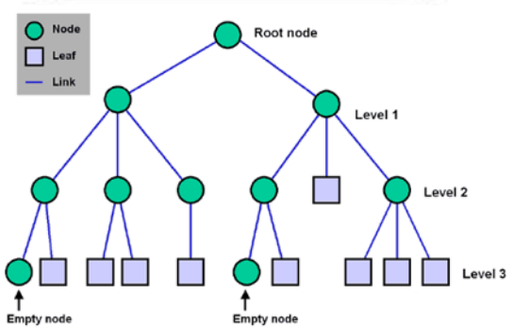


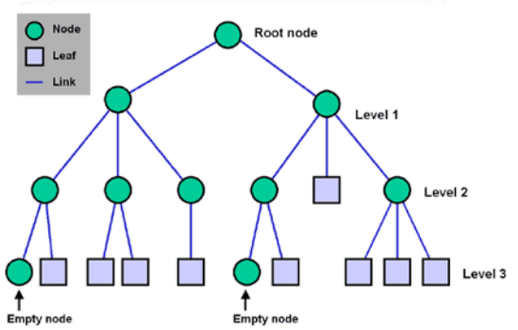
Service B



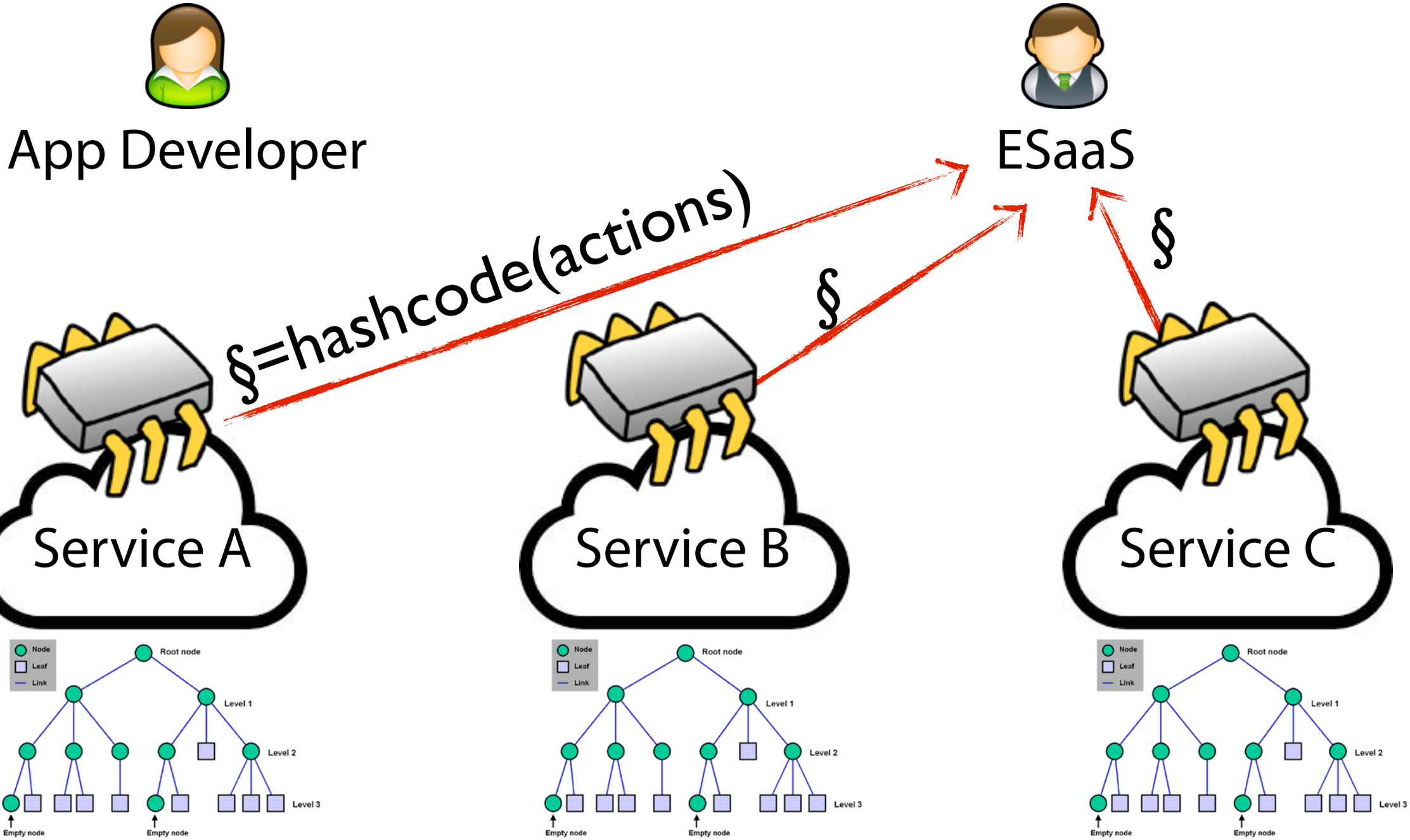
Service C





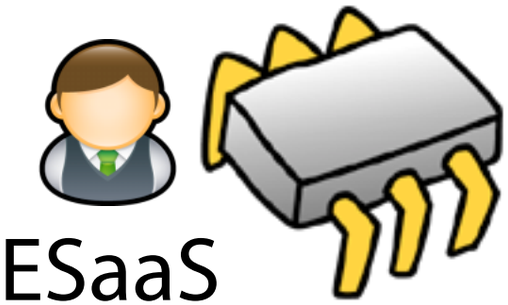


TPM-Based REaaS

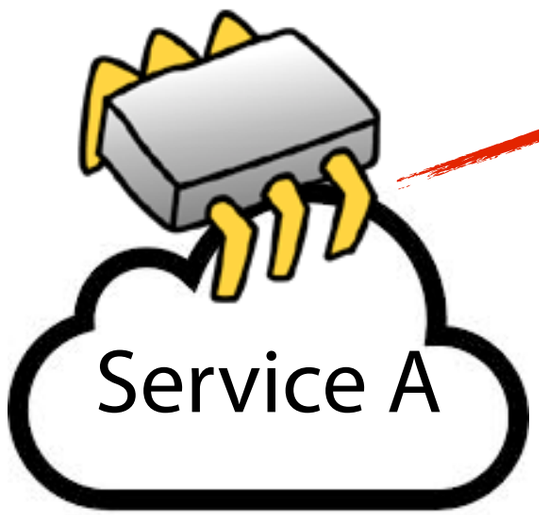


TPM-Based REaaS

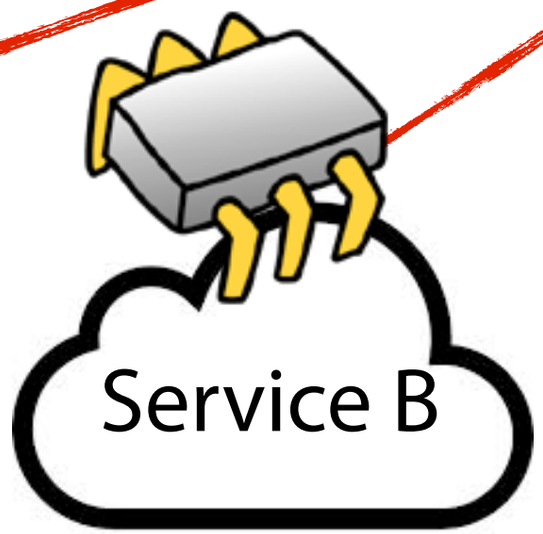
App Developer



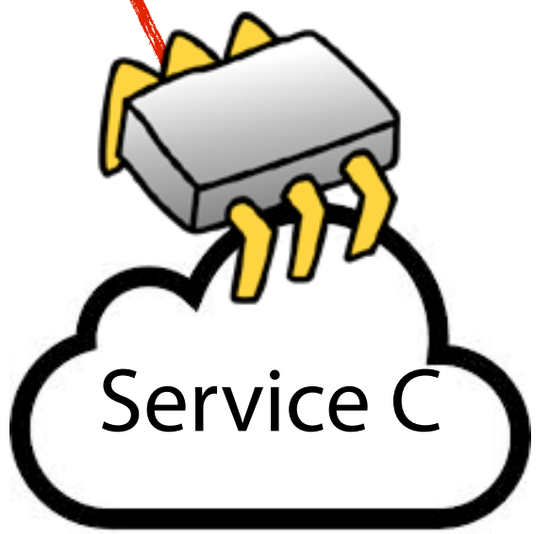
ESaaS



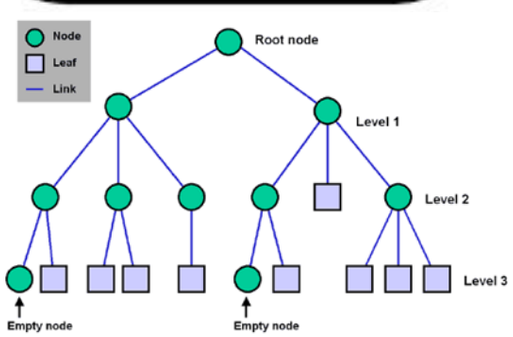
Service A

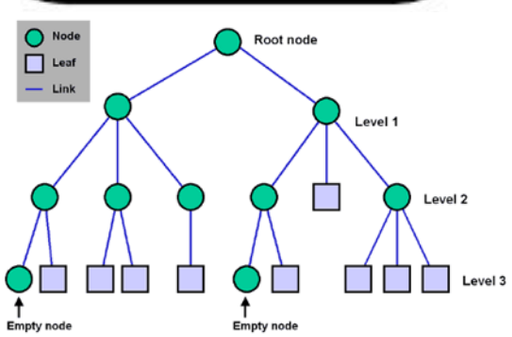


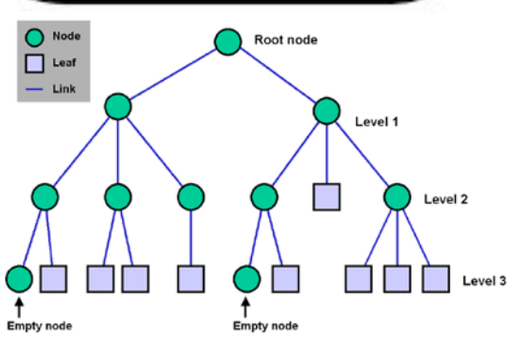
Service B



Service C







TPM-Based REaaS

App Developer

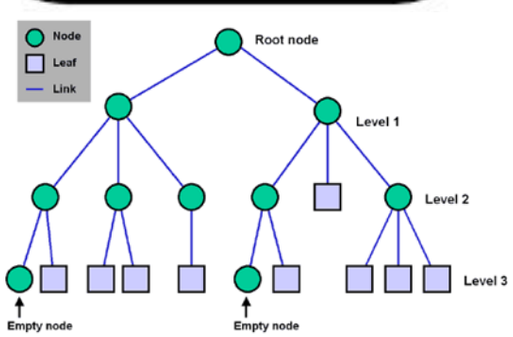
ESaaS

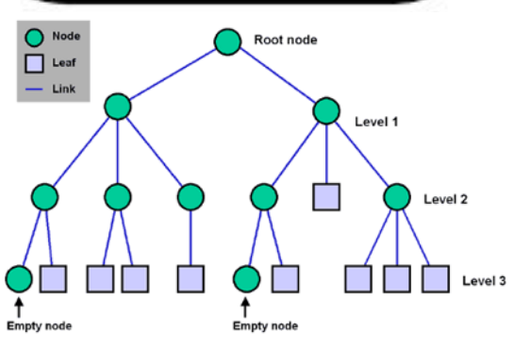
§

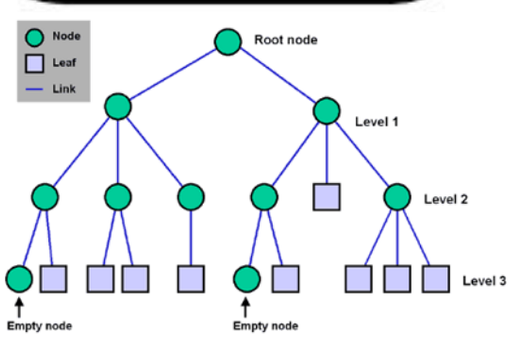
Service A

Service B

Service C







Road-Map

- Motivating Example
- REaaS Design
- Evaluation

Road-Map

- Motivating Example
- REaaS Design
- Evaluation

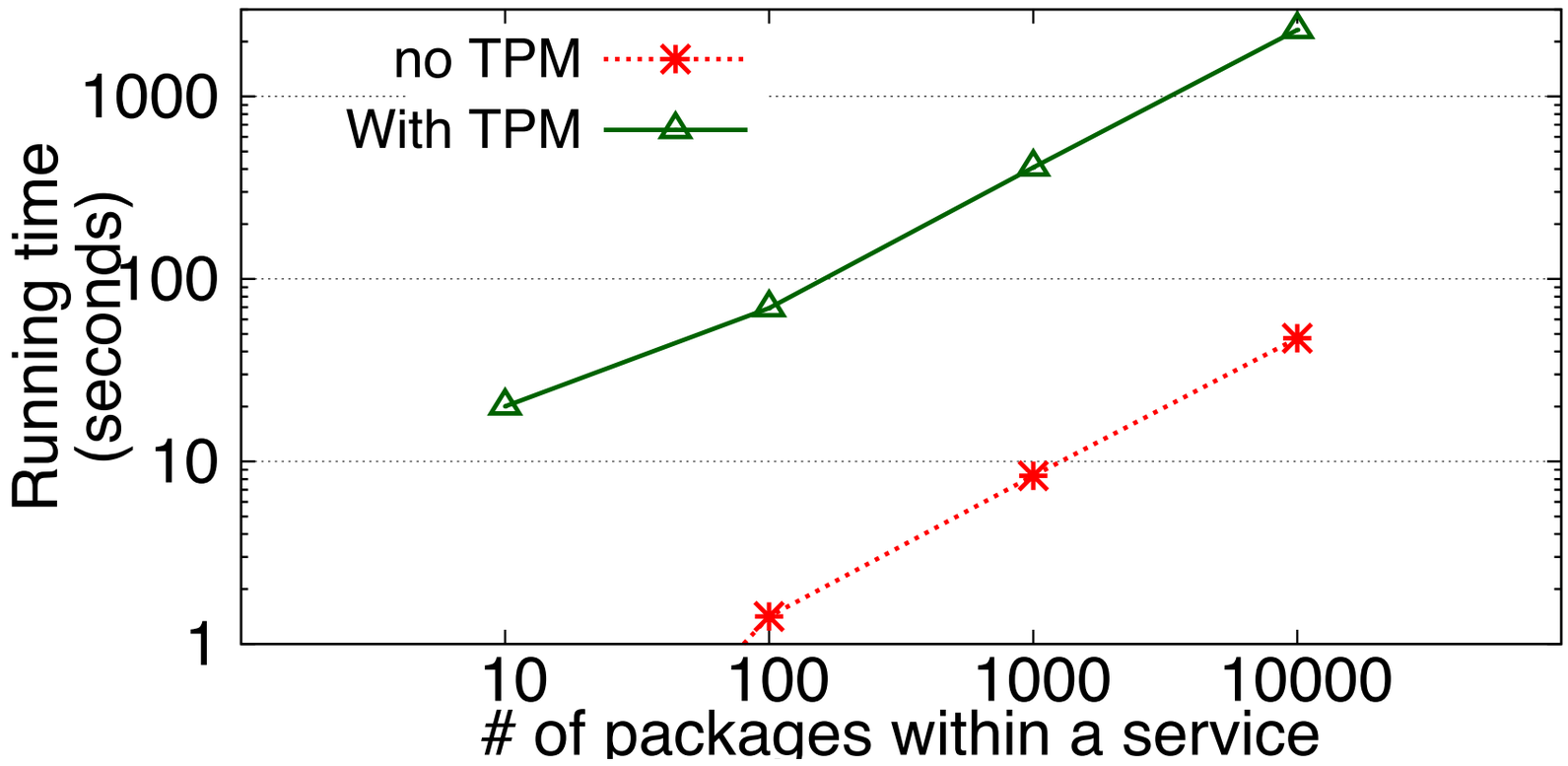
Case Study

	MySQL	PostgreSQL	Riak	MongoDB
# of packages	588	736	103	108
Risk score	8	7	4	2

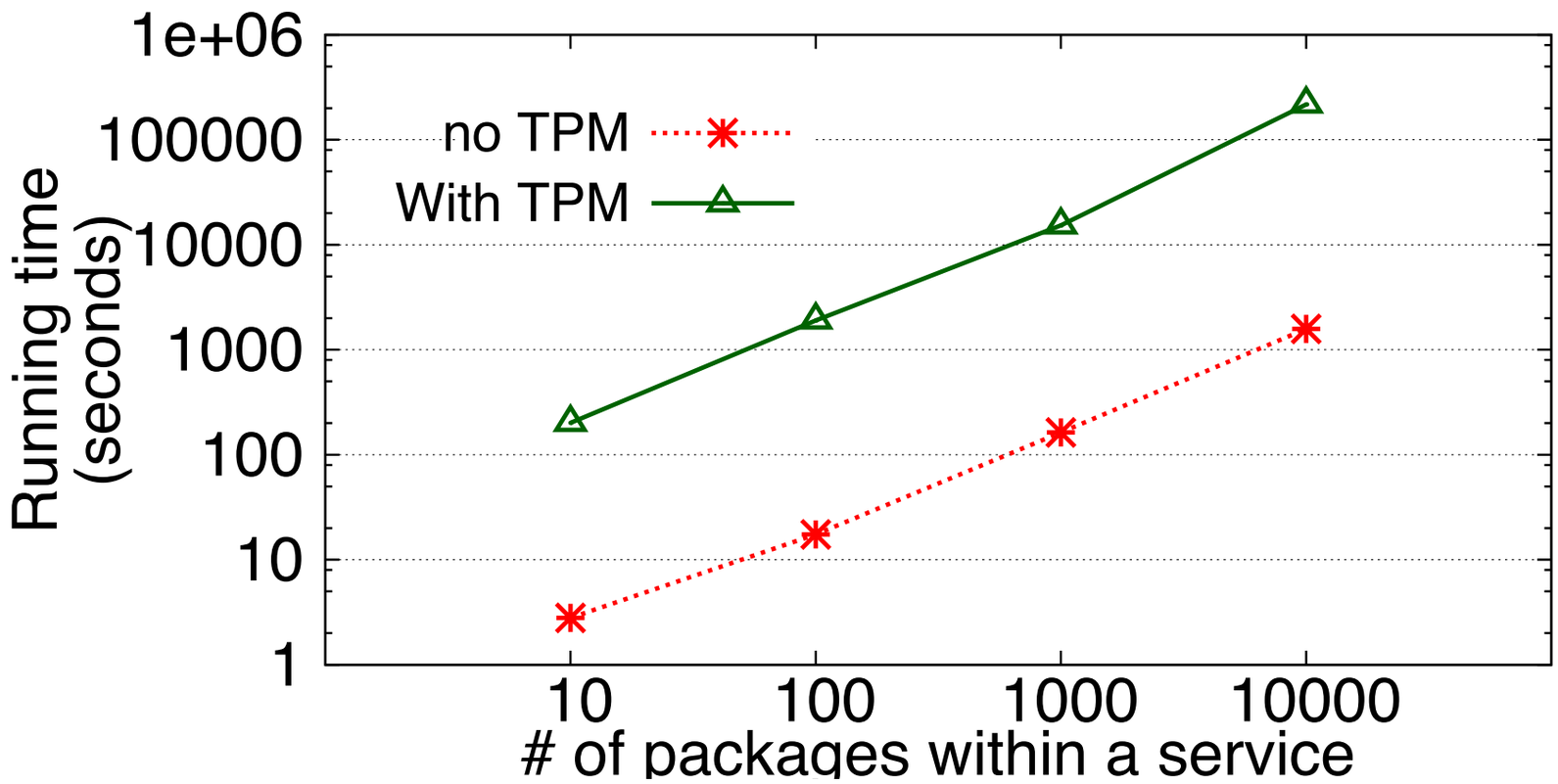
Performance Evaluation

- One Dell XPS14 laptop
  - 2.8GHz 4-Core Intel Xeon CPU
  - 16GB memory
- Public dataset with N packages
  - N = 10, 100, 1000, and 10000

Time for Dependency Collection



Time for Risk Evaluation



Conclusion

- The first-step towards practical risk-based service selection approach.
- TPM-based approach to prevent privacy leakage.
- A realistic case study and performance evaluation.

Thanks!