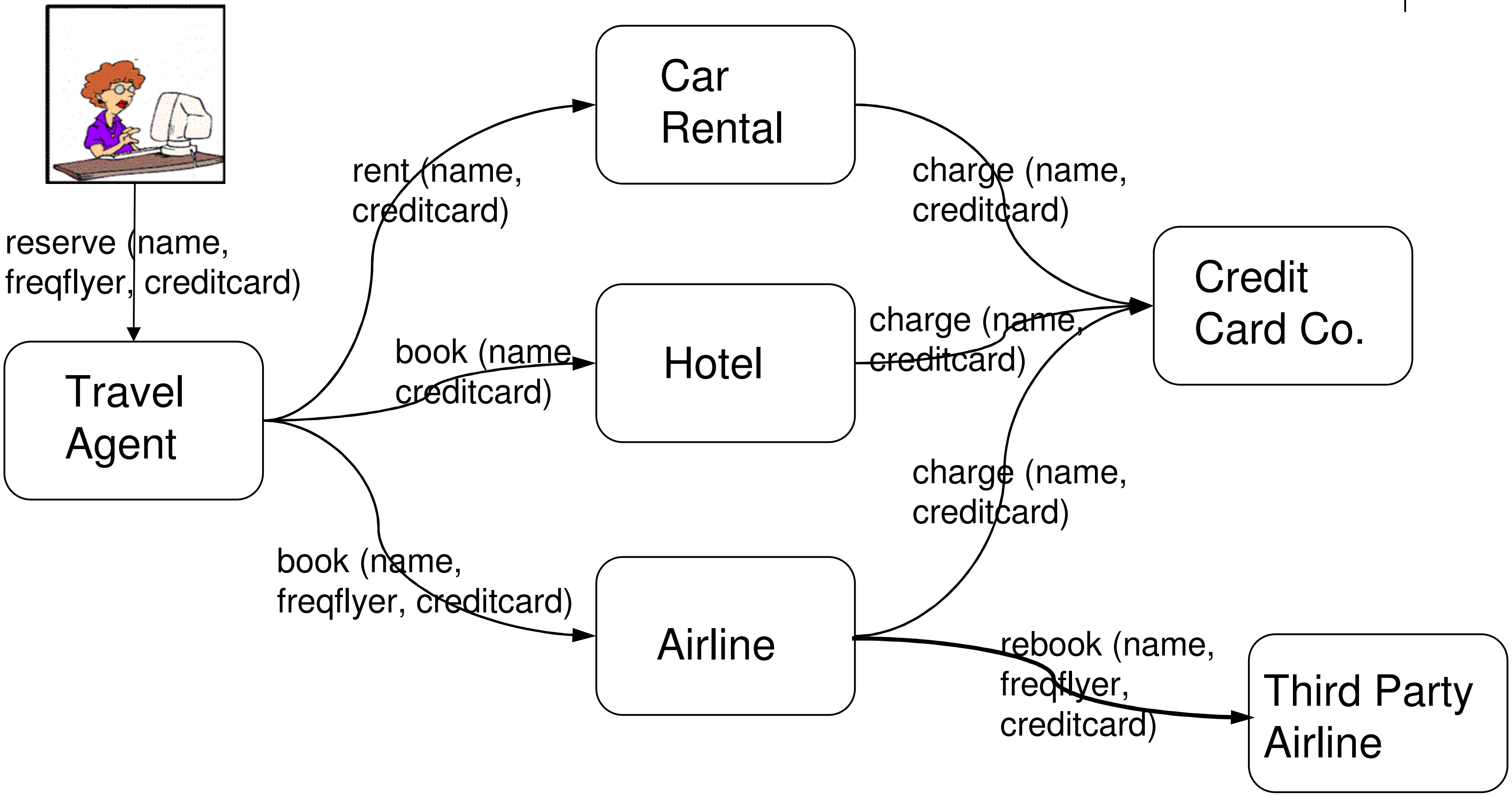
Guarding Sensitive Information Streams through the Jungle of Composite Web Services

Jinpeng Wei, Lenin Singaravelu, Calton Pu
Georgia Institute of Technology
Atlanta, Georgia, USA

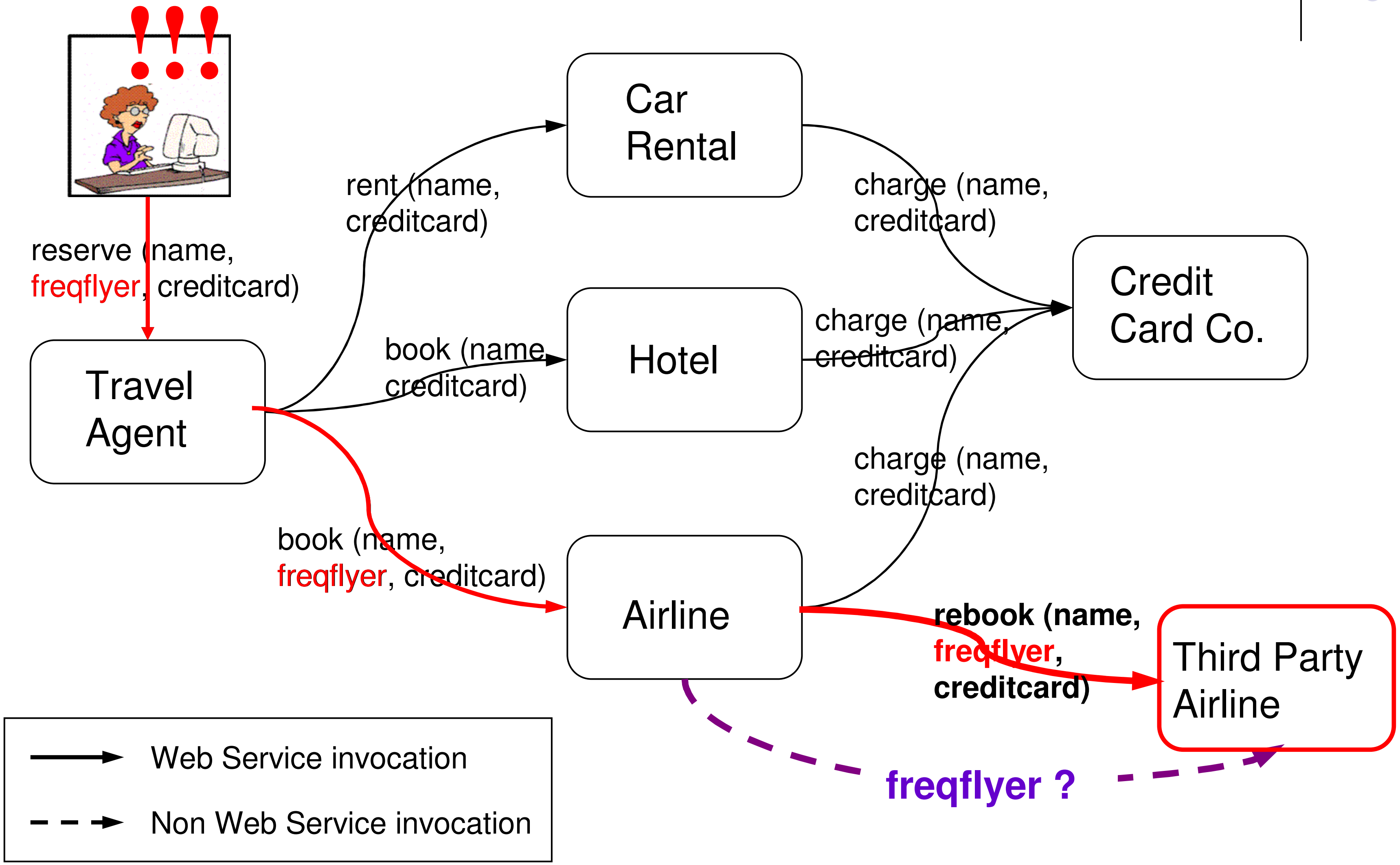
Agenda

- Problem statement
- WS-sensFlow - Security policy specification
- Concrete solution: SF-Guard
  - Security policy enforcement
  - Prototype implementation and its evaluation
- Related work and conclusion

Running Example: Travel Agent


reserve (name, freqflyer, creditcard)
Travel Agent
rent (name, creditcard)
Car Rental
book (name, creditcard)
Hotel
book (name, freqflyer, creditcard)
Airline
charge (name, creditcard)
charge (name, creditcard)
charge (name, creditcard)
Credit Card Co.
rebook (name, freqflyer, creditcard)
Third Party Airline

Running Example: Travel Agent



Agenda

- Problem statement
- WS-sensFlow - Security policy specification
- Concrete solution: SF-Guard
  - Security policy enforcement
  - Prototype implementation and its evaluation
- Related work and conclusion

WS-sensFlow

- Policy-based: specification and attachment of security policies to the web service invocation requests

- Fine-grain
  - Spatially, different data items can have different security policies
  - Temporally, the security policy for the same data item can change from one invocation to another

Security Policy Envelopes

- Formal Definition

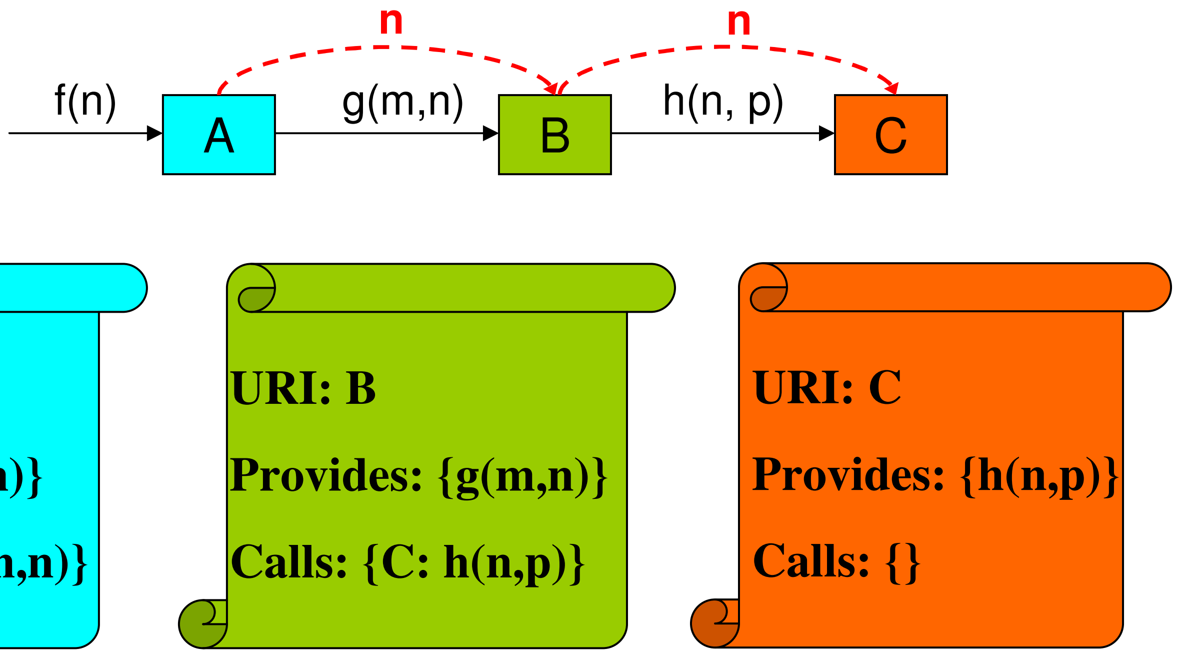
L = <white list>; <black list>
<white list> = **allow** <node list>
<black list> = **deny** <node list>
<node list> = * | <node id> | <node id>, <node list>

- Example

reserve (
name <allow * **>,**
freqflyer <allow Travel Agent, Airline, Hotel, Car Rental**;**
deny Third Party Airline , Credit Card Co**>,**
creditcard <…>)

Secure Policy Specification (1)

- Composite Service Topology Discovery
  - Leverage on meta-information exchanged dynamically among component web services
  - Leverage on ontology to infer information streams



URI: A

Provides: {f(n)}

Calls: {B: g(m,n)}

URI: B

Provides: {g(m,n)}

Calls: {C: h(n,p)}

URI: C

Provides: {h(n,p)}

Calls: {}

Secure Policy Specification (2)

- Generation of SPEs
  - Known nodes: based on the trust on them
  - Unfamiliar nodes: leverage on reputation and trust systems

```
reserve (
  name <allow *>,
  freqflyer <allow Travel Agent, Airline, Hotel, Car Rental;
                deny Third Party Airline , Credit Card Co>,
  creditcard <…>)
```

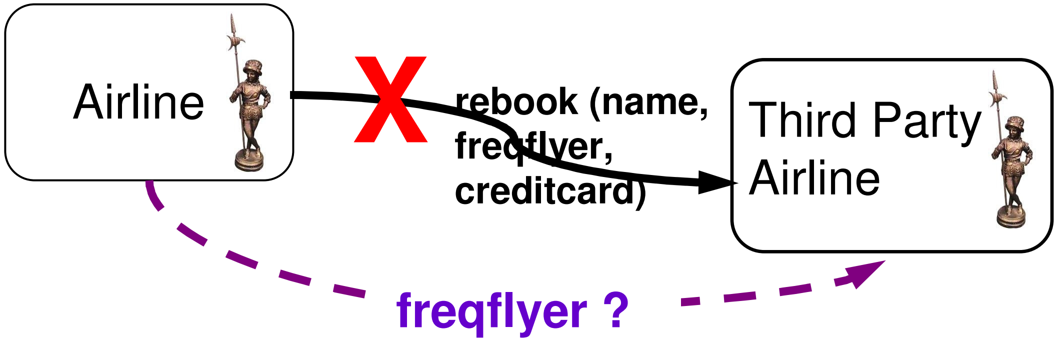
Agenda

- Problem statement
- WS-sensFlow - Security policy specification
- Concrete solution: SF-Guard
    - Security policy enforcement
    - Prototype implementation and its evaluation
- Related work and conclusion

Secure Policy Enforcement and Propagation: SF-Guard (1)

- Threat model: There is a minimal TCB (Trusted Computing Base) on each web service node, but the web service application (business logic) is not trusted
- SF-Guard is added as part of the TCB on each web service node to enforce the SPEs
- SF-Guard checks the security policy envelops before invoking a target web service

freqflyer **<allow** …; **deny** Third Party Airline, …**>**



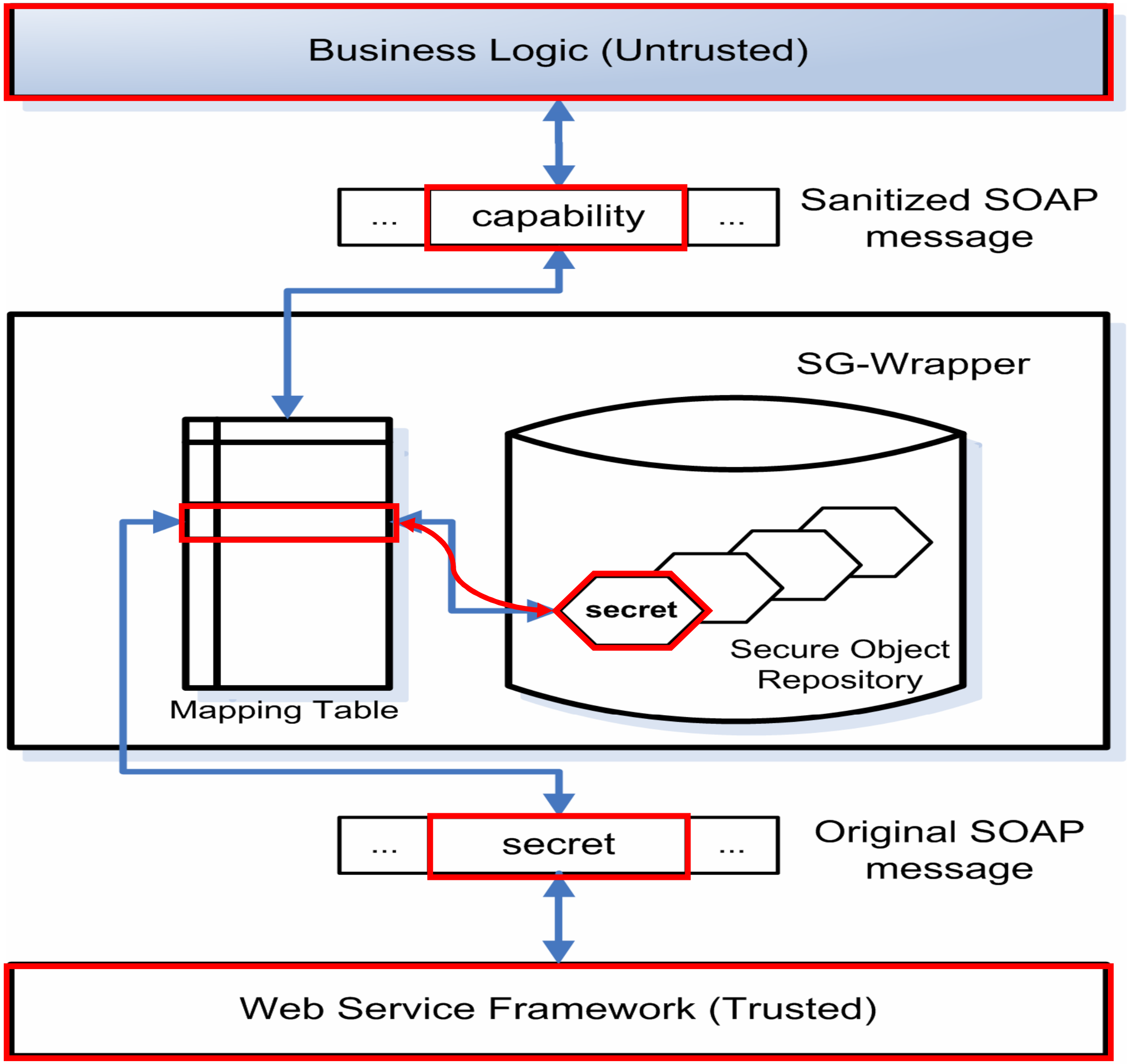
Secure Policy Enforcement and Propagation: SF-Guard (2)

- Using *capabilities* to hide sensitive information from the business logic.
- Operate on the sensitive information on behalf of the business logic

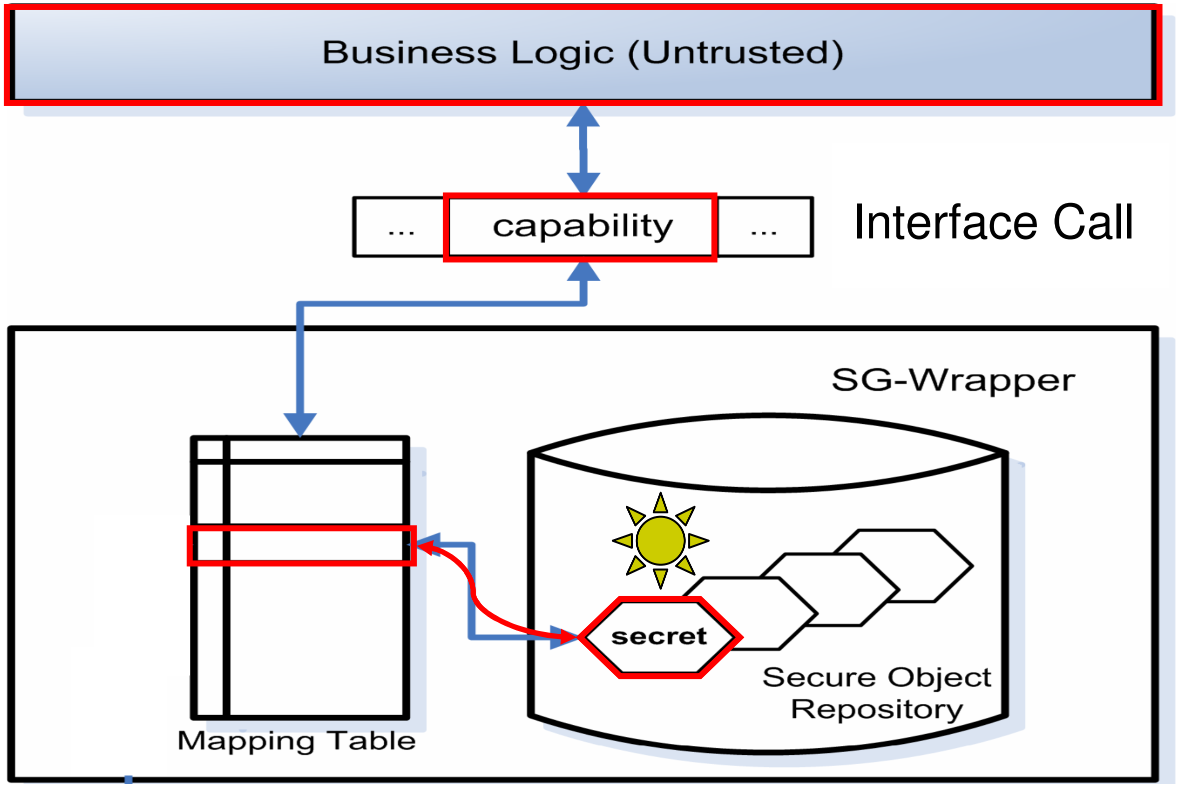
- Feasibility
  - Security-sensitive information is read only. E.g., Social security number
  - Security-sensitive information is atomic. E.g., Credit card number

- Conclusion: Only a few pre-defined simple interfaces are required.

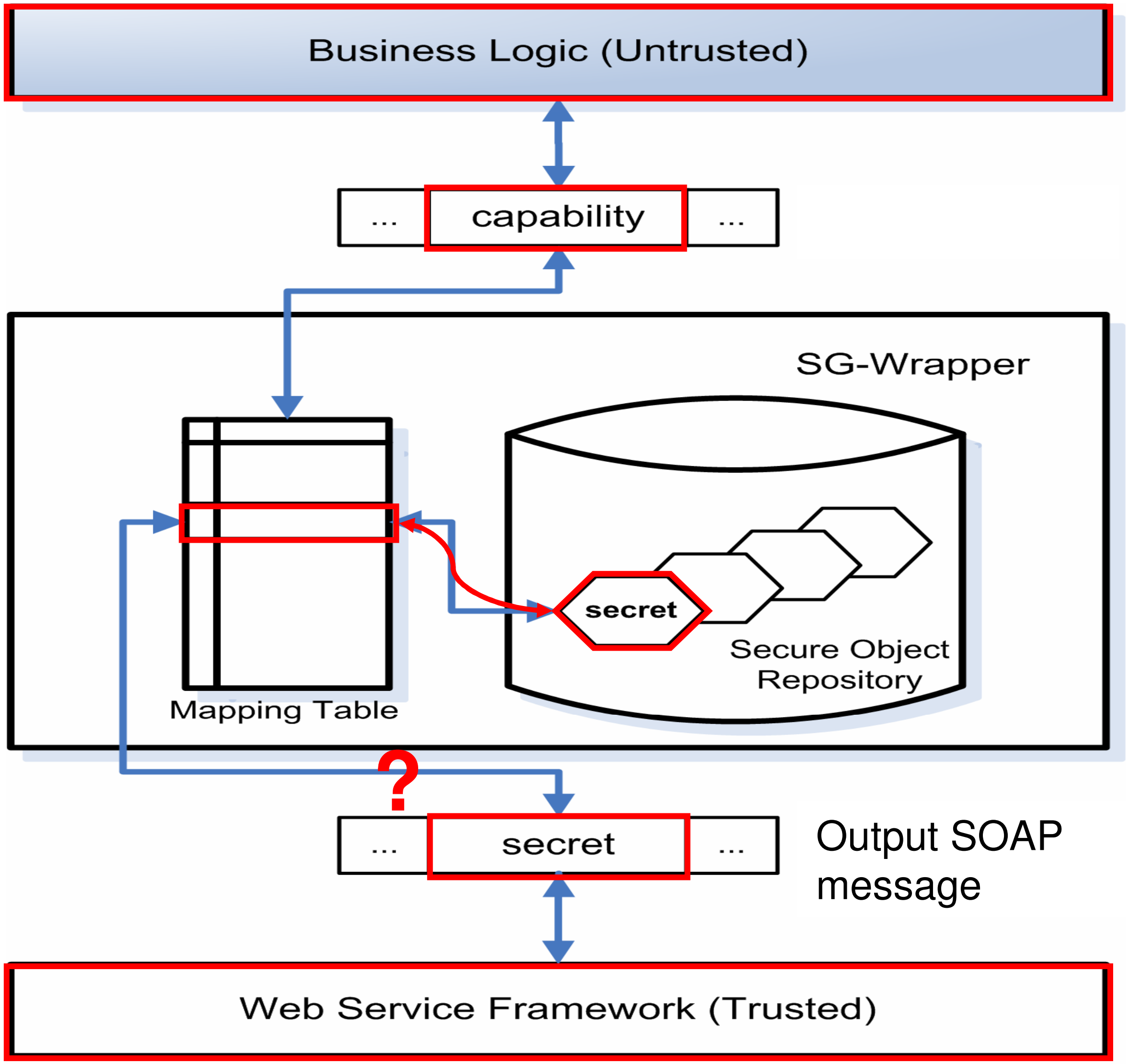
Incoming Message Sanitization



Normal Operation on the Sensitive Information



Outgoing Message Processing



Agenda

- Problem statement
- WS-sensFlow - Security policy specification
- Concrete solution: SF-Guard
  - Security policy enforcement
  - Prototype implementation and its evaluation
- Related work and conclusion

SF-Guard Prototype Implementation

- Based on Apache Axis2
- As a module inserted into the message processing stack, between web service framework and the business logic
- Works by checking and manipulating attributes of the XML elements in a SOAP message. e.g., *whitelist, blacklist, capability*
- A wrapper object for the sensitive information is passed on to the business logic through the Axis2 message context

Evaluation

- Protection of SF-Guard
  - Reducing the size of WSF that has to be trusted
- Reasonable overhead

	Client – T.A.	T.A.– C.Rtl.	C.Rtl.- Cred.	T.A.– Hotel	Hotel– Cred.	T.A.– Air.	Air.– Cred.	Air.– T.P.A.
Original (ms)	793	413	305	123	61	182	60	60
SF-guard (ms)	819	422	310	130	63	192	61	65
Overhead	3.3%	2.2%	1.6%	5.7%	3.3%	5.5%	1.7%	8.3%

Agenda

- Problem statement
- WS-sensFlow - Security policy specification
- Concrete solution: SF-Guard
  - Security policy enforcement
  - Prototype implementation and its evaluation
- Related work and conclusion

Related Work

- Information privacy in web applications
  - P3P (Platform for Privacy Preferences)
- Access control in composite web services [Elisa Bertino, ICWS’06]
- Compliance checking of privacy policies [Xu, ICWS’06]

Conclusion

- WS-senFlow specification to support fine-grain, policy-based access control of security-sensitive data in composite web services
- The SF-Guard architecture to enforce the security policy specifications
- Using a wrapper style design with capability-based protection
- Prototype implementation shows strong protection properties and low overhead

Questions?

Thank you!