Authentication of User's Device and Browser for Data Access in Untrusted Cloud

Denis Ulybyshev[1], Bharat Bhargava[1], Leon Li[2], Jason Kobes[2], Donald Steiner[2] , Harry Halpin[4], ByungChan An[1], Miguel Villarreal[1], Rohit Ranchal[3]

[1]Computer Science and CERIAS, Purdue University; [2]NGC Research Consortium; [3]IBM Watson Health Cloud; [4]MIT

MOTIVATION

- Data (D) = $\{d_1,..,d_n\}$
- Access Control Policies (P) = $\{p_1,..,p_k\}$

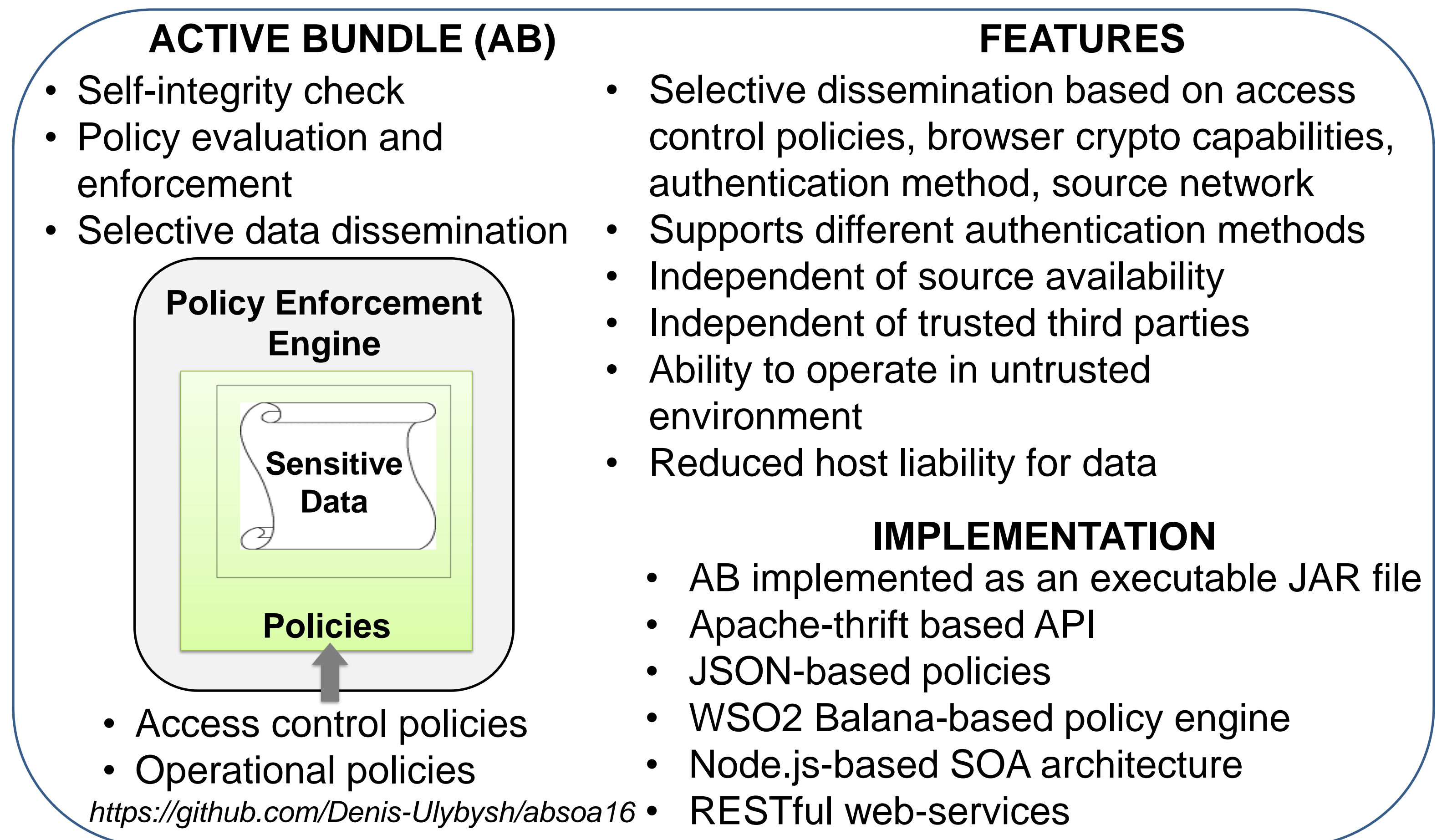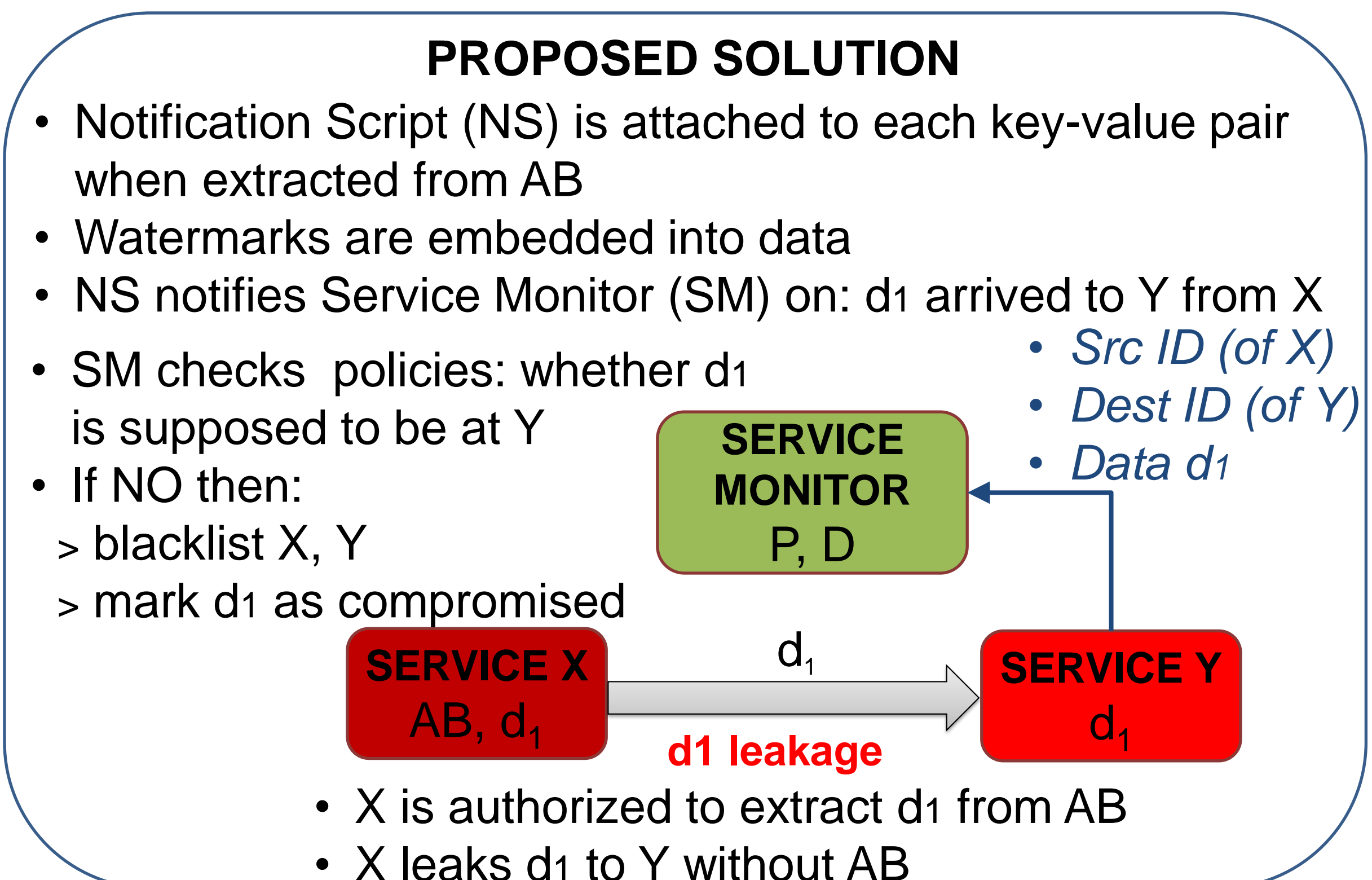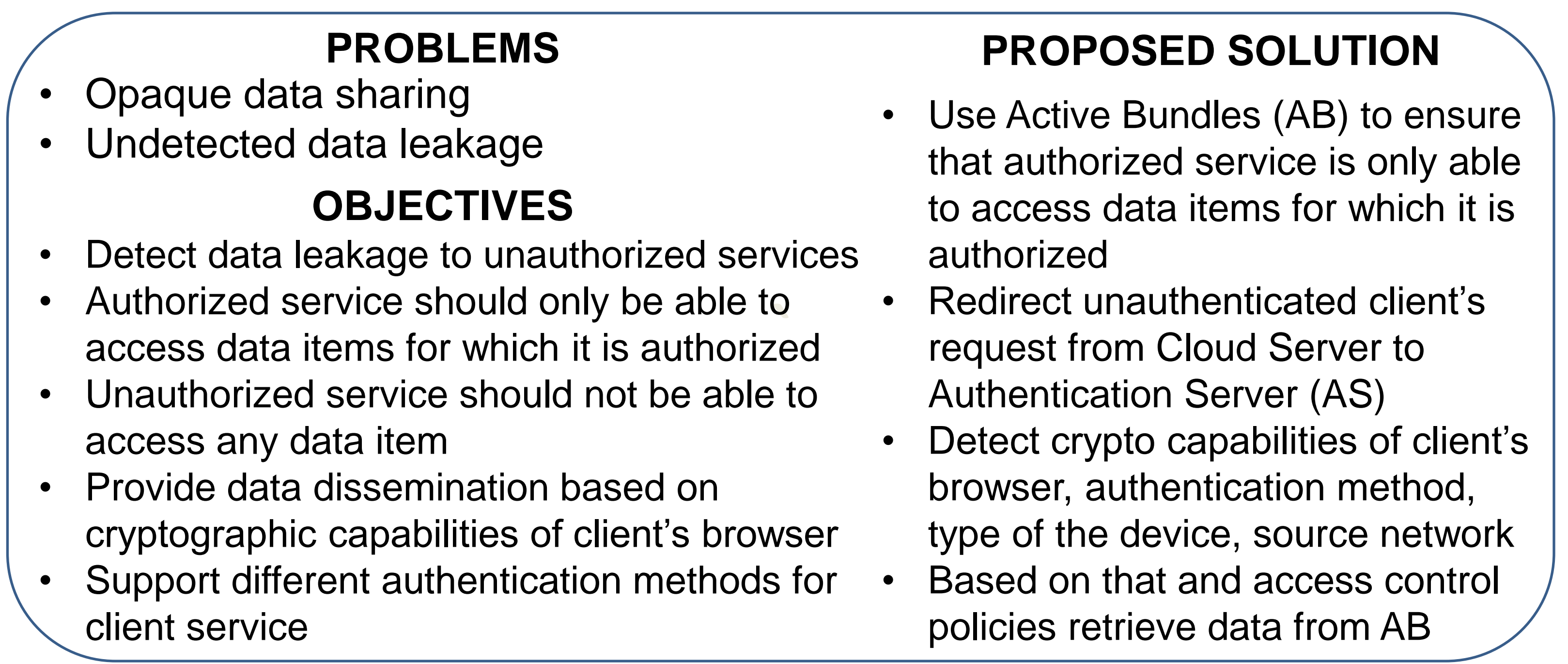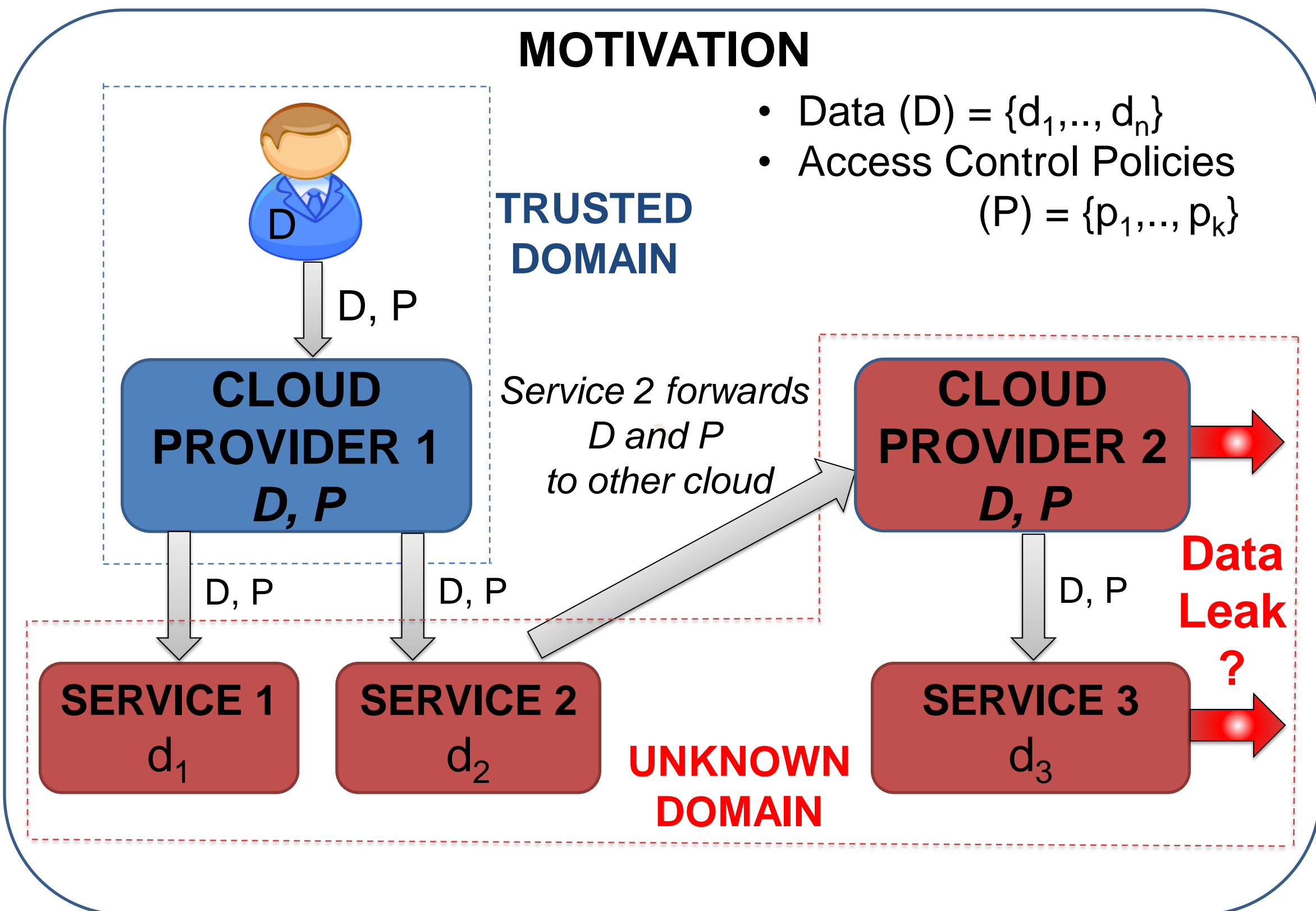## PROBLEMS

- Opaque data sharing
- Undetected data leakage

OBJECTIVES

- Detect data leakage to unauthorized services
- Authorized service should only be able to access data items for which it is authorized
- Unauthorized service should not be able to access any data item
- Provide data dissemination based on cryptographic capabilities of client's browser
- Support different authentication methods for client service

PROPOSED SOLUTION

- Use Active Bundles (AB) to ensure that authorized service is only able to access data items for which it is authorized
- Redirect unauthenticated client's request from Cloud Server to Authentication Server (AS)
- Detect crypto capabilities of client's browser, authentication method, type of the device, source network
- Based on that and access control policies retrieve data from AB

PROPOSED SOLUTION

- Notification Script (NS) is attached to each key-value pair when extracted from AB
- Watermarks are embedded into data
- NS notifies Service Monitor (SM) on: d_1 arrived to Y from X
- SM checks policies: whether d_1 is supposed to be at Y
- If NO then:
  - > blacklist X, Y
  - > mark d_1 as compromised

- Src ID (of X)
- Dest ID (of Y)
- Data d_1

- X is authorized to extract d_1 from AB
- X leaks d_1 to Y without AB

ACTIVE BUNDLE (AB)

- Self-integrity check
- Policy evaluation and enforcement
- Selective data dissemination

- Access control policies
- Operational policies

https://github.com/Denis-Ulybysh/absoa16

FEATURES

- Selective dissemination based on access control policies, browser crypto capabilities, authentication method, source network
- Supports different authentication methods
- Independent of source availability
- Independent of trusted third parties
- Ability to operate in untrusted environment
- Reduced host liability for data

IMPLEMENTATION

- AB implemented as an executable JAR file
- Apache-thrift based API
- JSON-based policies
- WSO2 Balana-based policy engine
- Node.js-based SOA architecture
- RESTful web-services

ELECTRONIC HEALTHCARE RECORDS DISSEMINATION IN CLOUD

Data request flow:
- Clients are directed to the services according to their role (researcher, insurance or doctor)
- Authorization level (included in ticket) and role define the data accessible by requester
- For role "Doctor", depending on Authorization Level: doctor will be given either "only his/her own patient info" or "his/her own and other doctors' patients info"

AUTHENTICATION TICKET CREATION AND VALIDATION

Authentication Server:
- Knows shared secret K and Private Key PrivKey
- Ticket_Info = (Auth_Level, Expiration_Time, Client_ID, Client_Role, Request_Field)
- Enc_Ticket_Info = $EncAES256_K$(Ticket_Info)
- Ticket_Signature = $Enc_{PrivKey}$(SHA512(Enc_Ticket_Info))
- Ticket = <Enc_Ticket_Info, Ticket_Signature>

Doctor, Insurance or Researcher Service:
- Knows shared secret K and Public Key PubKey
- Receives Ticket = <Enc_Ticket_Info, Ticket_Signature>
- Checks: Dec_{PubKey}(Ticket_Signature) = SHA512(Enc_Ticket_Info)
- Gets data: $DecAES256_K$(Enc_Ticket_Info)

FUTURE WORK
- Implement data leakage detection based on watermarks
- Support database of ABs in Hospital Information System
- Comprehensive performance and scalability evaluation
- Support isolated AB execution AB (Linux Docker Container)

PURDUE UNIVERSITY