

Efficient Enforcement of Dynamic Cryptographic Access Control Policies for Outsourced Data

Anne V.D.M. Kayem¹, Patrick Martin², and Selim G. Akl²

10th Annual Information Security South Africa (ISSA) Conference

Rosebank, Johannesburg, South Africa
(August 15 – 17, 2011)

1 -- University of Cape Town, South Africa

2 -- Queen's University, Kingston, ON, CANADA

Outline

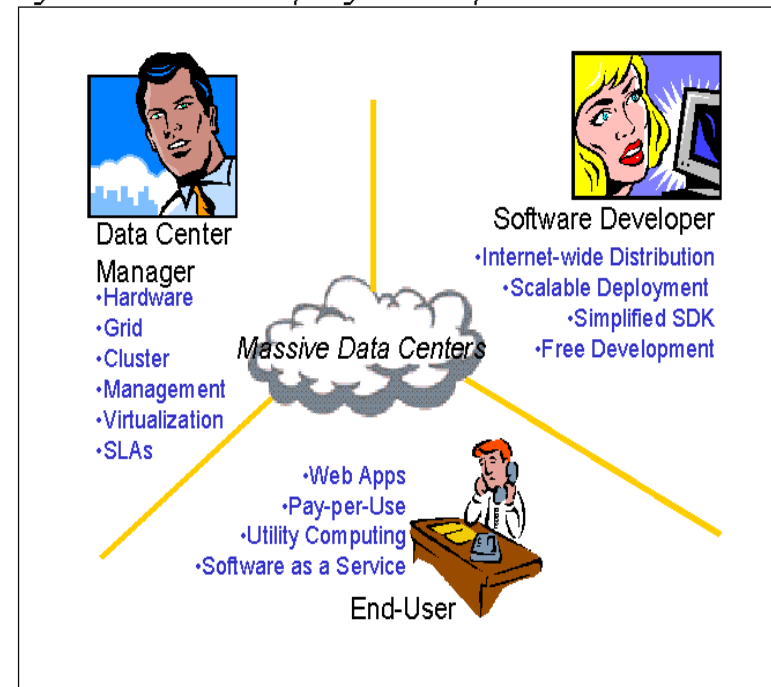
- Introduction
 - Context and Motivation
- Background
- Problem Statement
- An Access Control Approach
 - Key Protection
 - Data Access Mechanism
 - Key Update Procedure
- Performance Analysis
- Conclusion: Summary and Future Work

Introduction

Context and Motivation

- Large sources of data on the internet
- Need to take advantage of distributed data sources
 - Data outsourcing
 - Service oriented architectures
 - Sensor network data
 - Robotics – Accessing Historical Data...
- Cost effective information sharing

Figure 1 – Three Cloud Computing User Participants

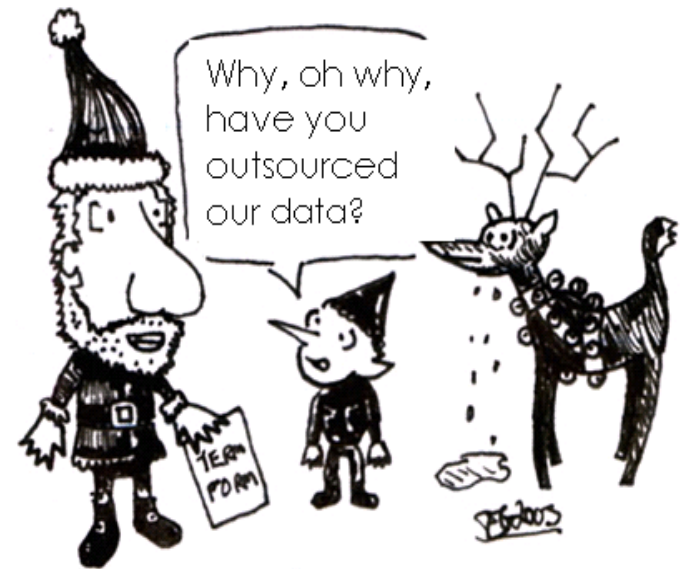


Source: Martin Hingley, June 2009

Introduction

Context and Motivation

- Example: Data Outsourcing
- Organizations transfer data management to third party service providers
- Minimizes costs (data storage, customer service management,...)



Copyright © 2004 Stephen E. Gideon

Question: How to formulate and enforce rules that guarantee data security and privacy beyond trusted boundaries?

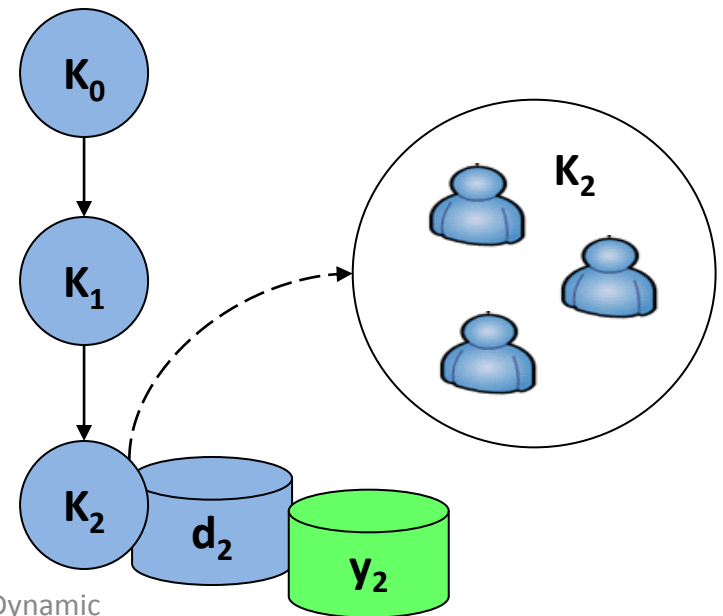
Background

- Environment and Assumptions:
 - Data owners, service provider, users
 - Access control mechanism on both the data owner's and service provider's end
 - Updates are applied by sending the service provider a new encrypted copy



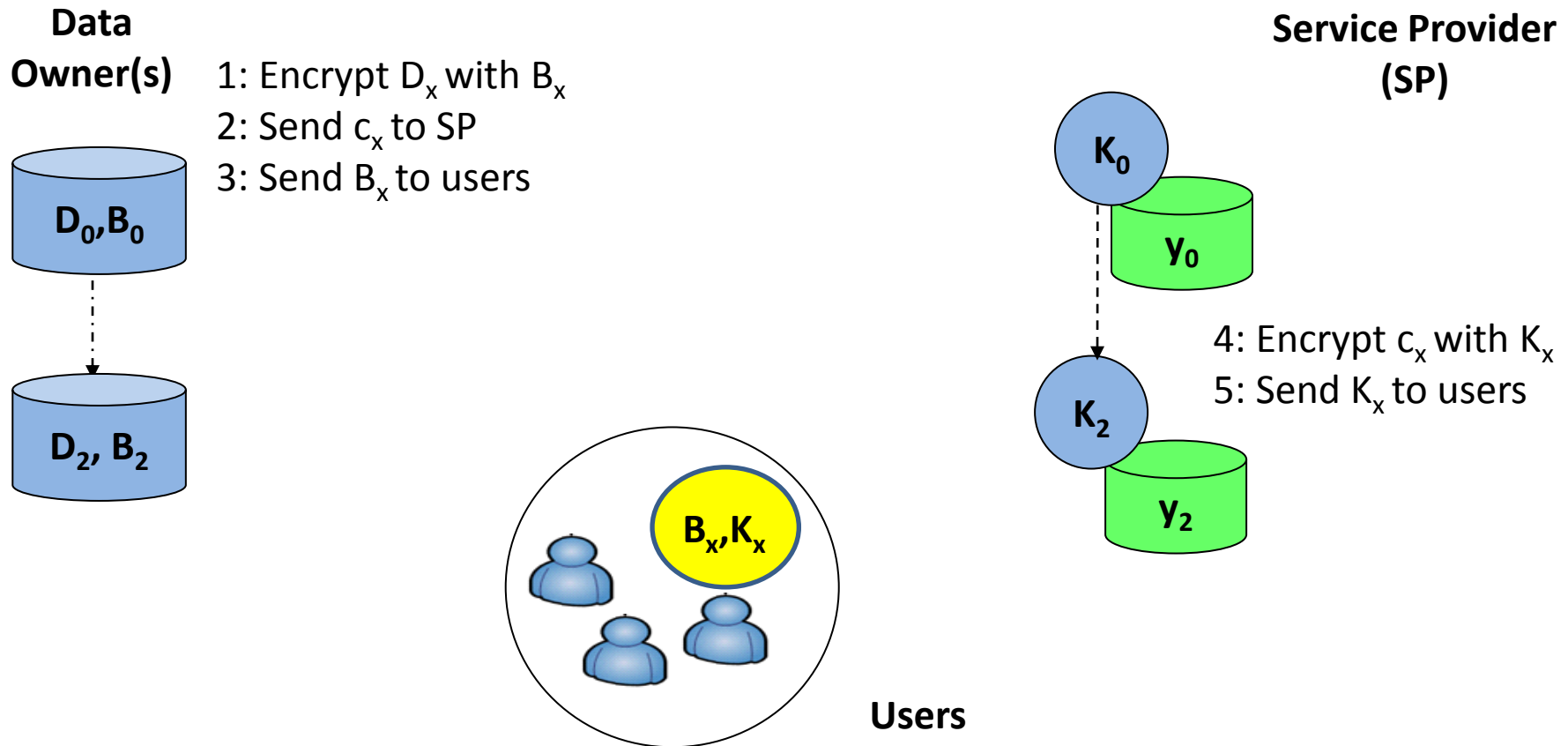
Background

- An Access Control Method:
 - Cryptographic access control
 - Access hierarchy
 - High level users access information at lower levels but not the reverse



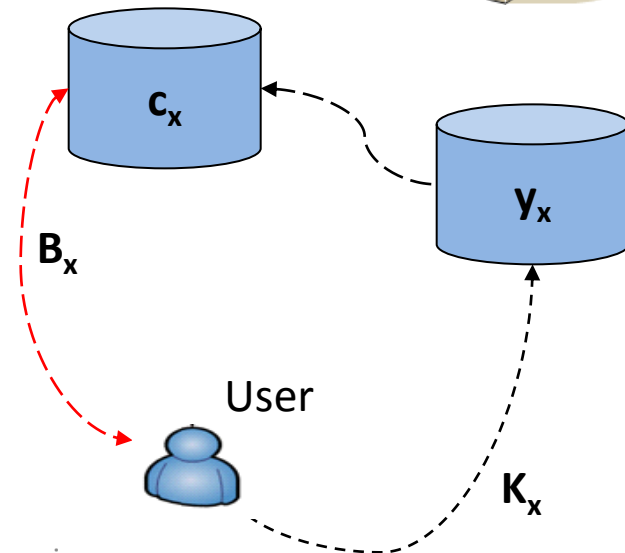
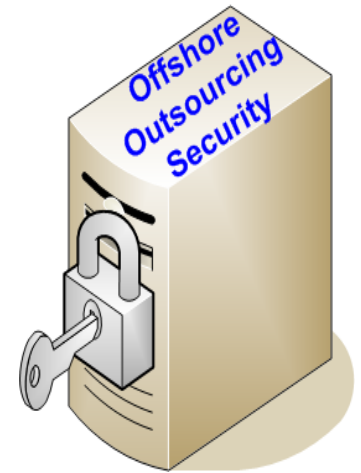
Background

De Capitani Di Vimercati et al. - 2007

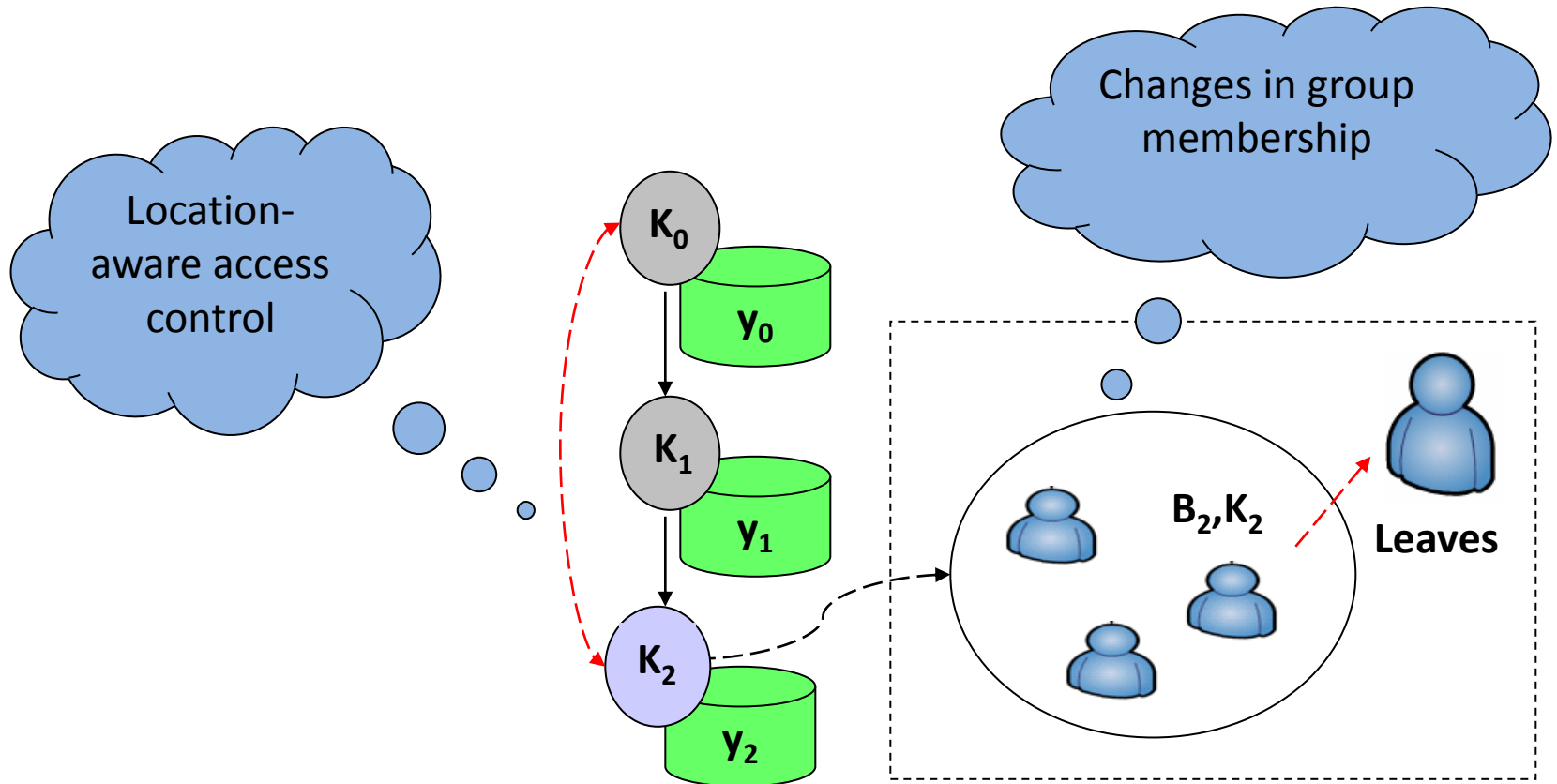


Background

- Data Access mechanism
- User:
 - Use K_x to decrypt y_x (from service provider)
 - Obtain c_x
 - Use B_x to decrypt c_x (into a readable format)



Problem Statement

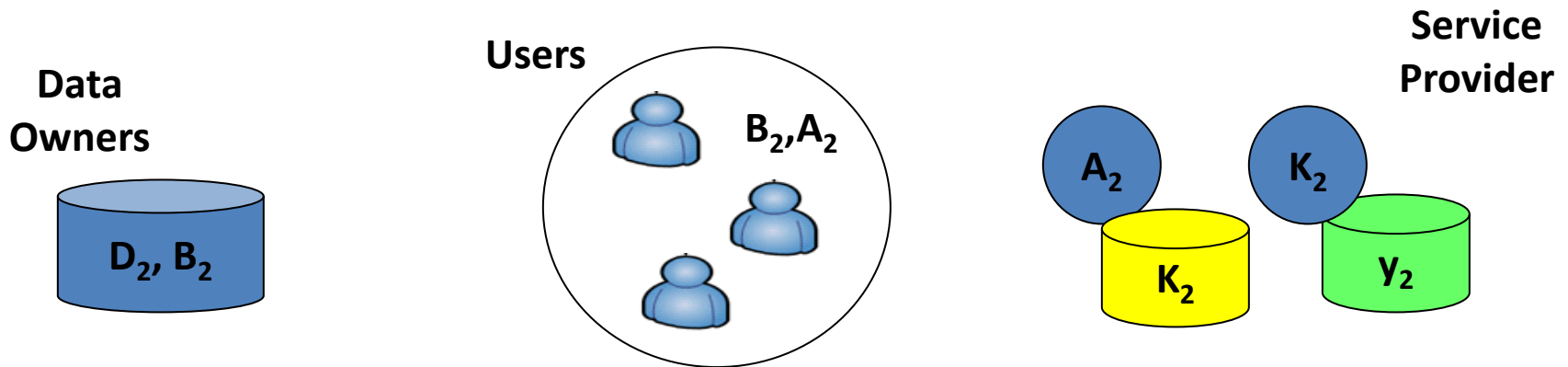


Two Problems:

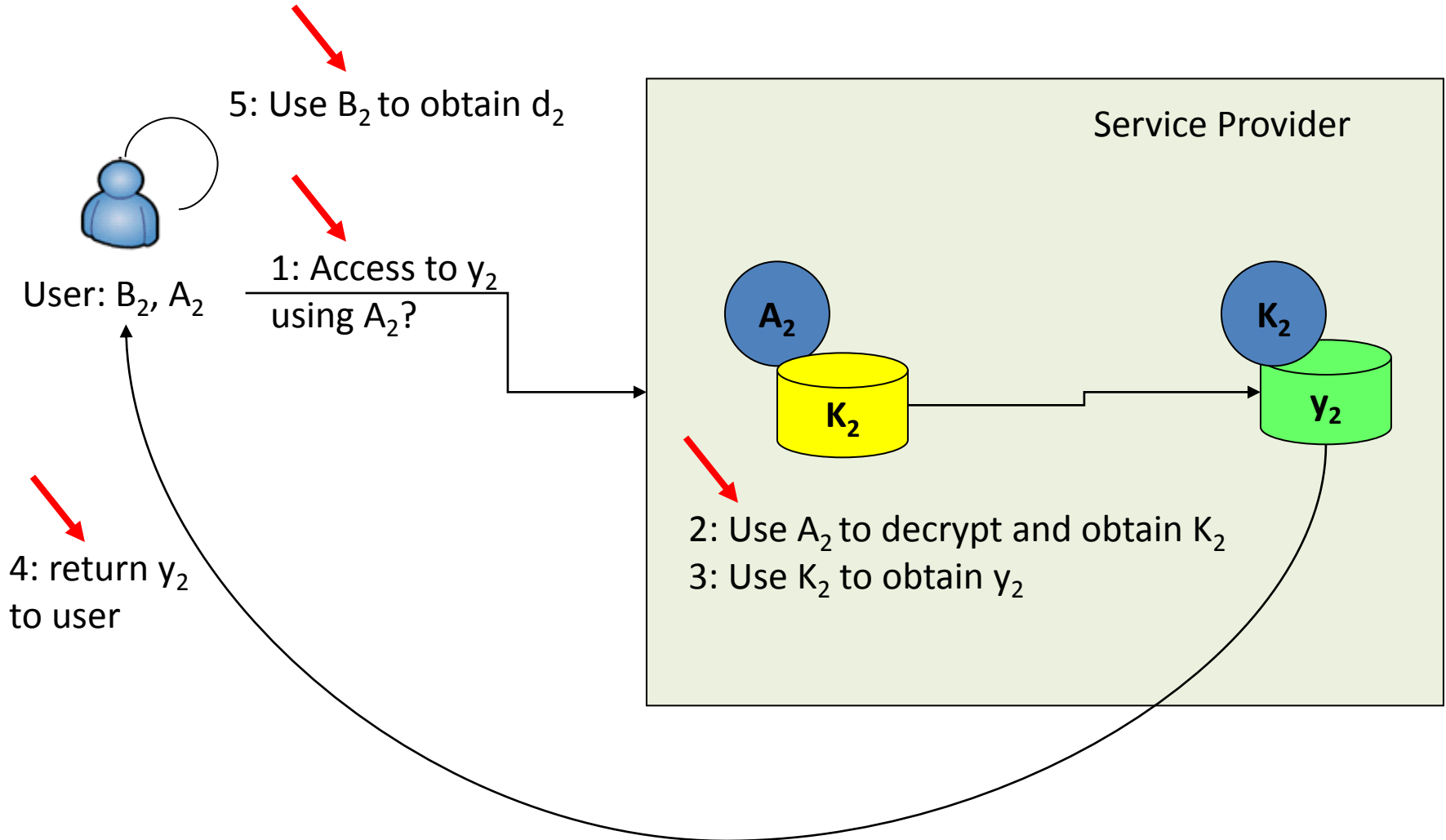
- Performance in handling key Updates
- Guaranteeing data privacy

Key Protection

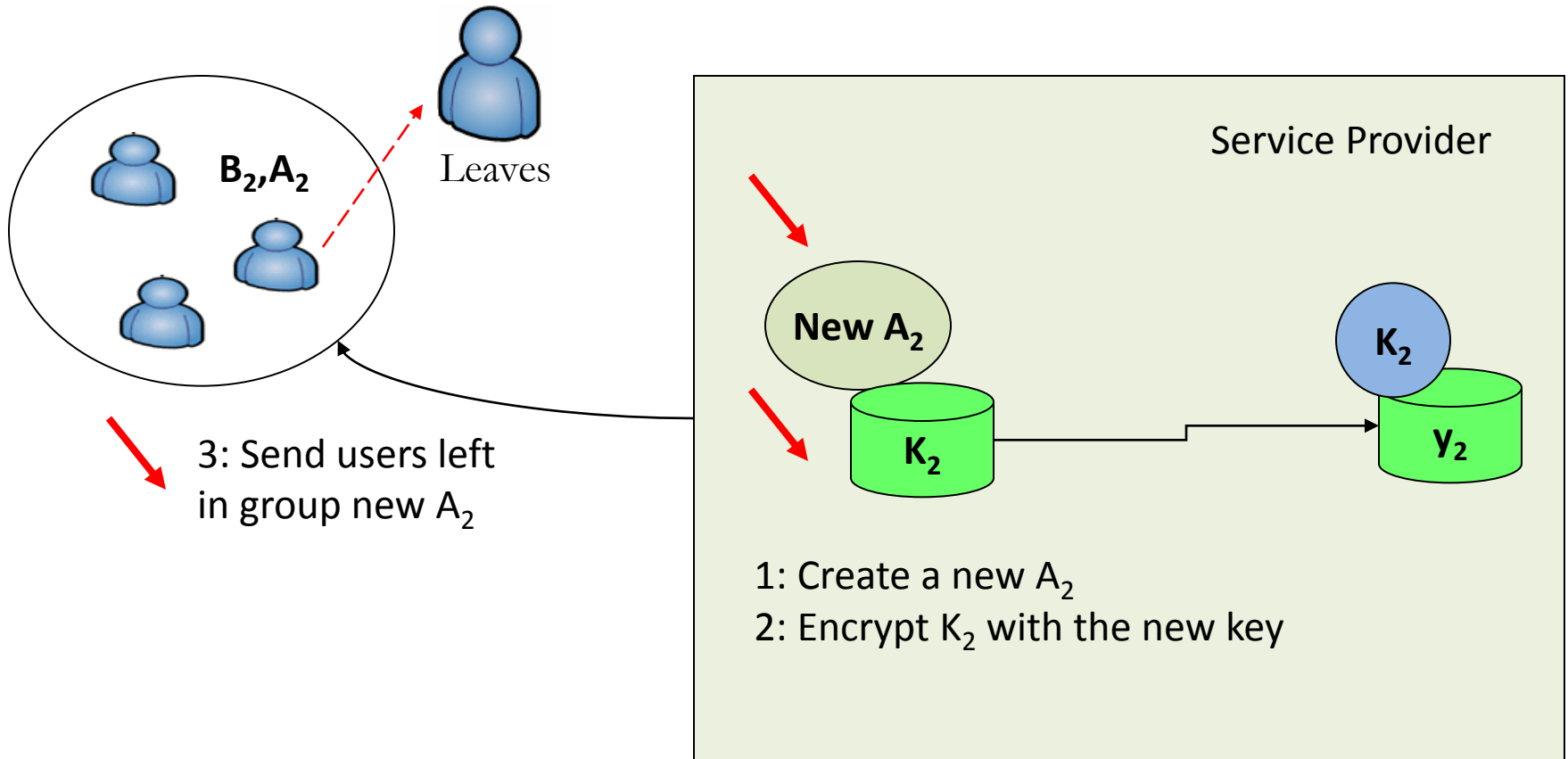
- **Idea:** Use a third layer of encryption on the service provider's end.
- **Goals:**
 - No necessity for data re-encryption when security policies change
 - Alleviates cost of data re-encryptions and vulnerability due to repeated transfers.
- **Method:** Service provider creates A_x , encrypts K_x and transmits A_x to clients



Data Access Mechanism



Key Update Procedure

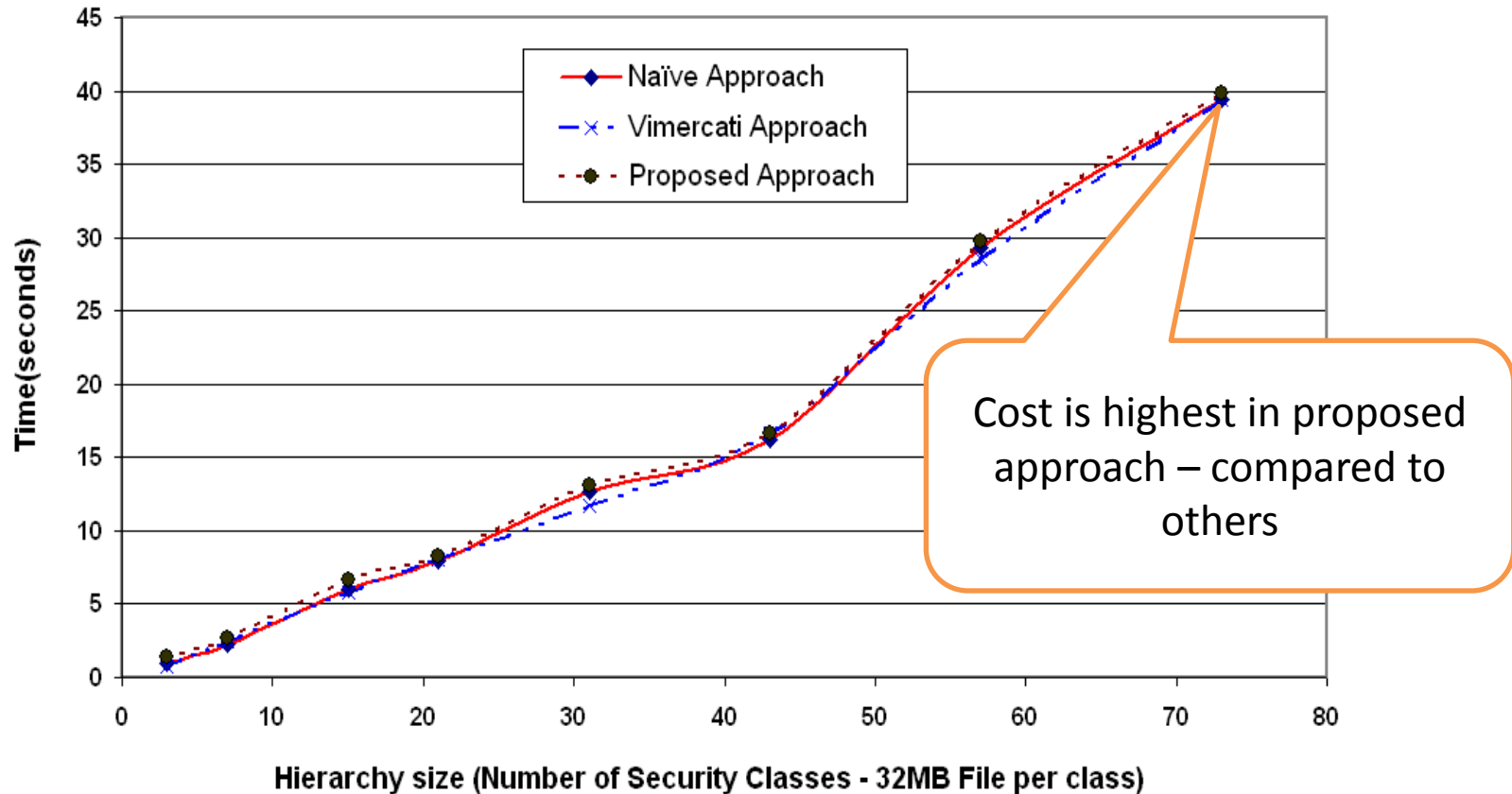


Implementation Setup

- Experiments on IBM Pentium IV computer
- Intel 3.00Ghz processor and 1GB of RAM.
- Implemented using Java 2 Standard Development Kit and Eclipse on Microsoft Windows XP Platform
- Number of executions: 10
- Size of data \approx 32MB
- Key Management and Updates with:
 - One node with 100 users
 - Client-Server model
 - Hierarchy sizes of 3 – 73 security classes

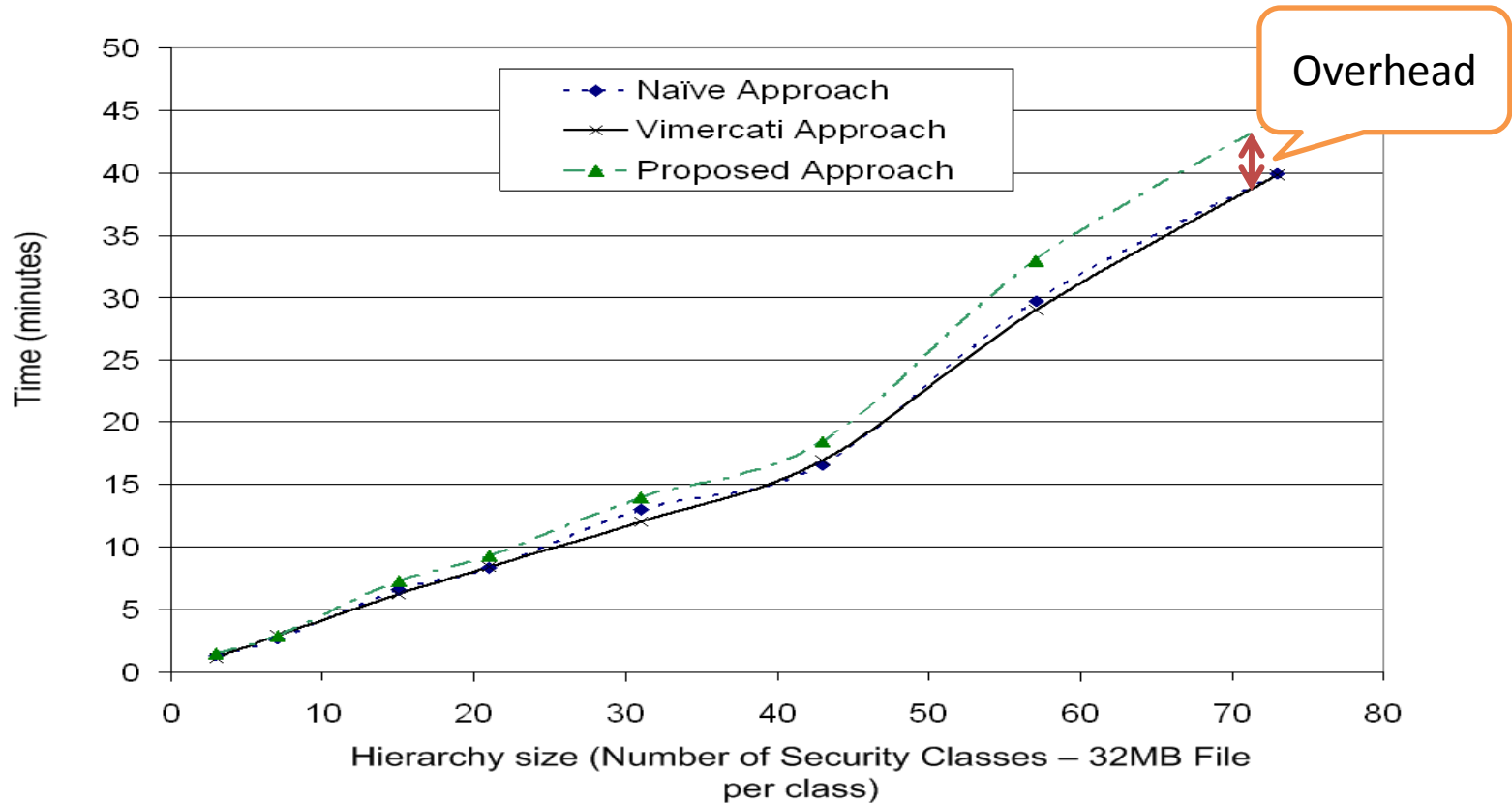
Performance Evaluation

- System Setup : Cost of Key Generation only per hierarchy



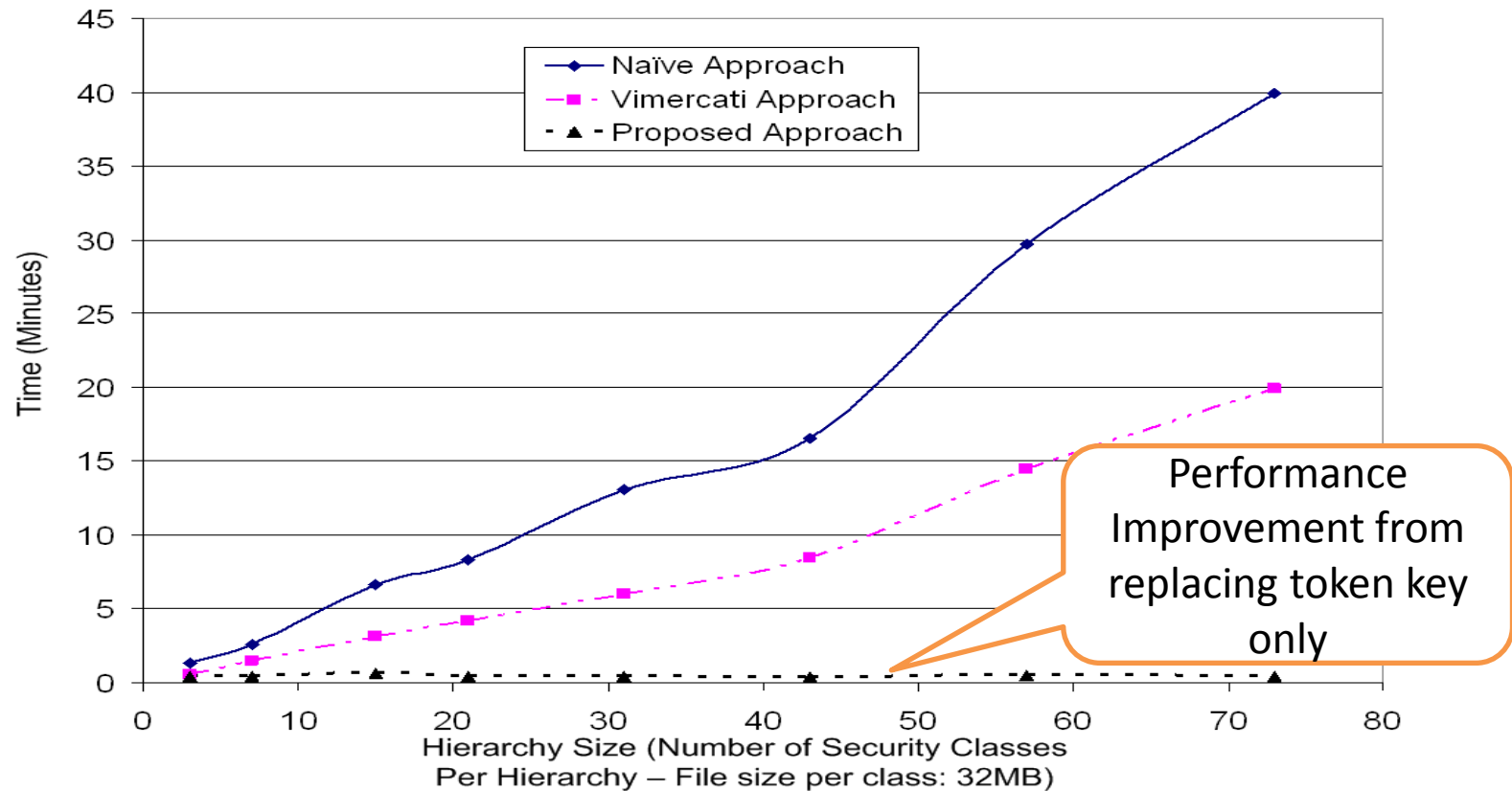
Performance Evaluation

- System Setup : Cost of Key Generation and Data Encryption per hierarchy



Performance Evaluation

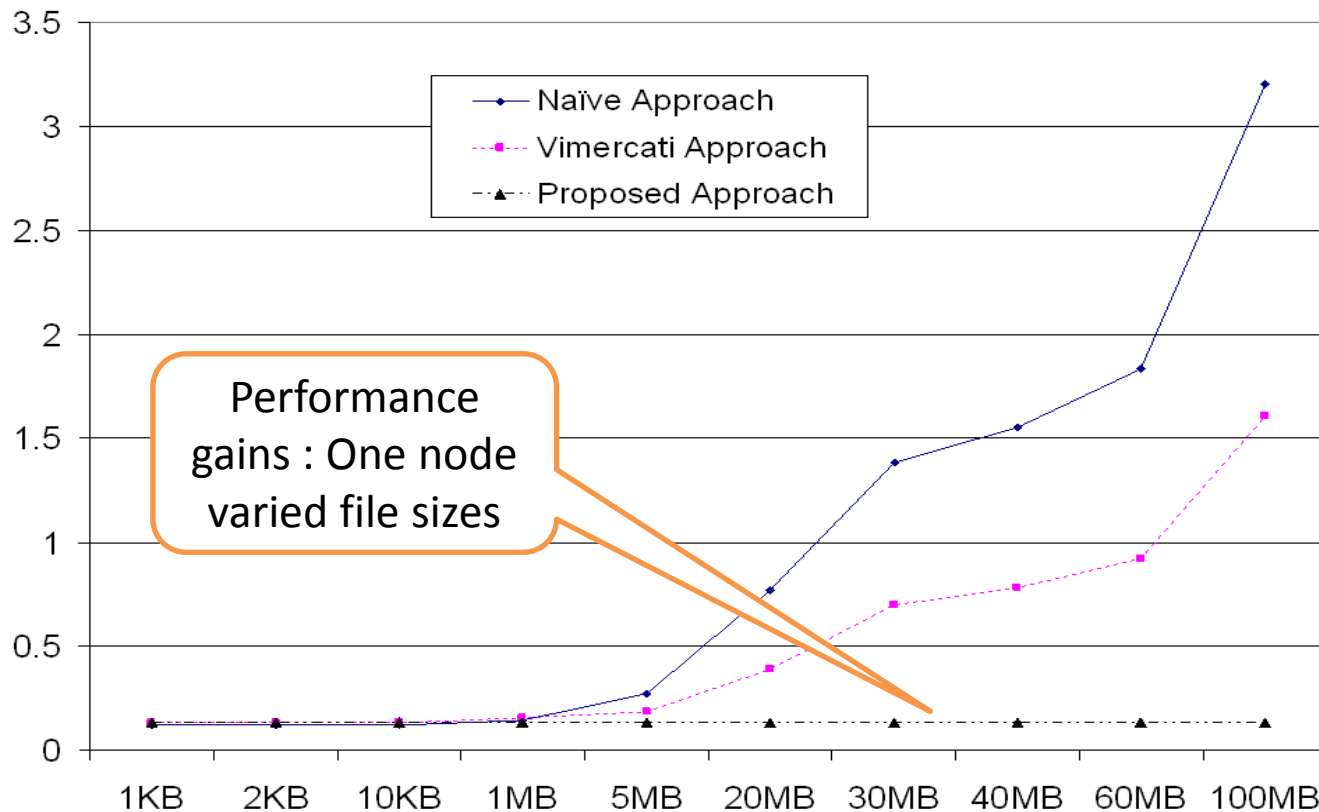
- Key Updates: Cost of Replacements and Re-encryptions



Performance Improvement from replacing token key only

Performance Evaluation

- Key Updates: Cost of Replacements and Re-encryptions per file size

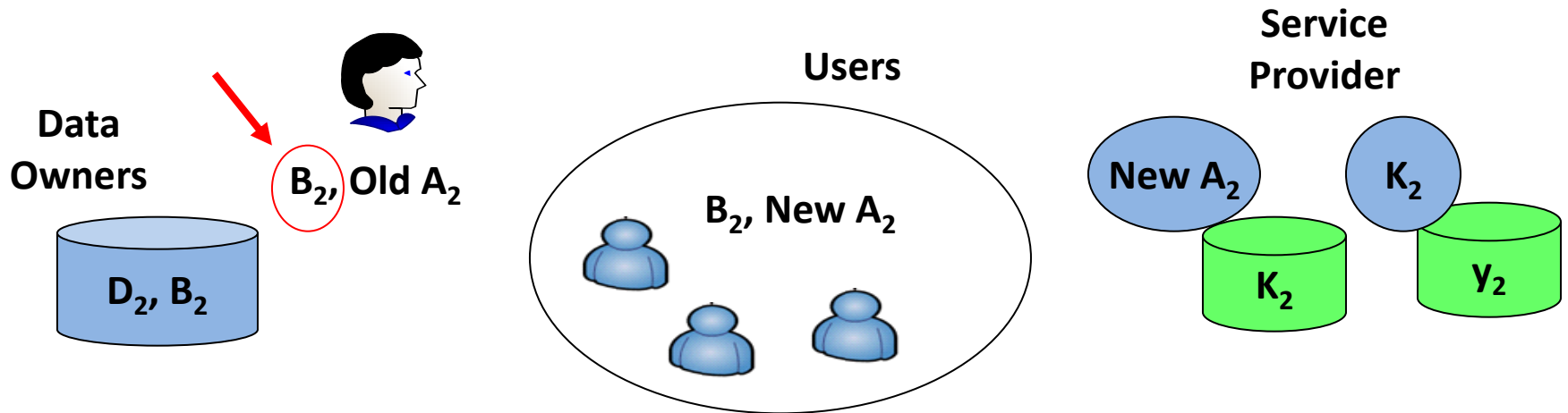


Conclusion: Summary

- Outsourced Data protection
 - security and performance efficiency of cryptographic access control (CAC) mechanisms
 - Existing CAC schemes solve the security issue
 - But!!! – Can be slow when faced with large volumes of data
- Our solution
 - Circumvents the performance concern
 - How?? → avoiding time intensive data re-encryptions to handle security policy updates
 - Uses a third key so updates imply only updating this key instead of the data.
- Performance Analysis
 - Shows approach to be cost effective for security policy updates
 - Case of CAC scheme in the outsourced scenario

Conclusion : Future Work

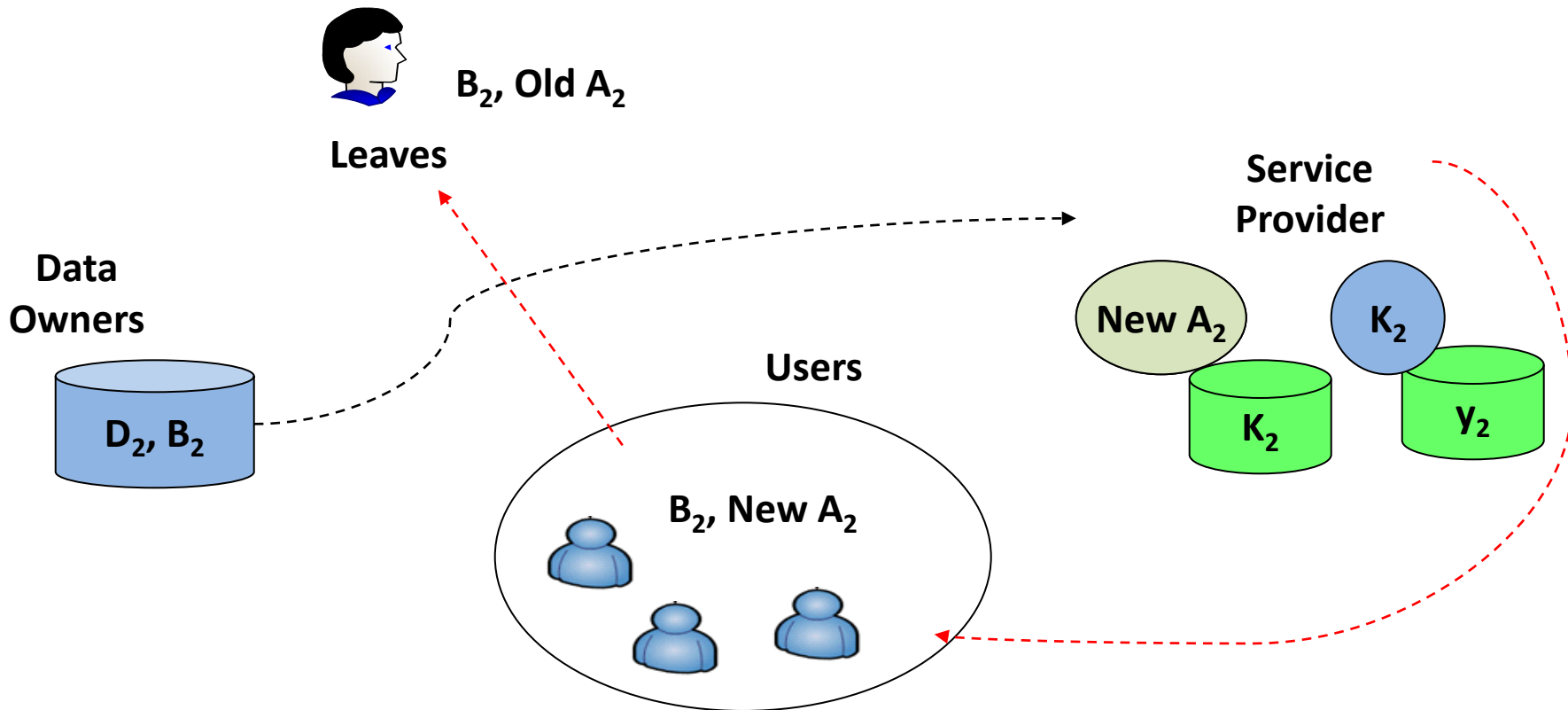
- Key protection solves the performance problem but can fail to prevent malicious updates
 - Preventing updates by excluded users except by complete change is difficult



Efficient Enforcement of Dynamic Cryptographic Access Control Policies for Outsourced Data

Conclusion : Future Work

- Problem of “lost” updates or data consistency



Thank You!

Email: akayem@cs.uct.ac.za