WEB SECURITY
UNIT - III

Outline:

Database Security:

- Recent Advances in Access Control
- Access Control Models for XML
- Database Issues in Trust Management and Trust Negotiation
- Security in Data Warehouses and OLAP Systems

3. Database Issues in Trust Management and Trust Negotiation

3.1 Introduction to Trust Management

- **Authorization** is a central issue in:
  - Computer security
  - Privacy protection
  - Achieving confidentiality, integrity, and availability
- **Traditional authorization**:
  - Based on **pre-established trust relationships**
  - Used within a single organization
  - Relies on identities (login names, passwords, PKI identities)
  - Example: enterprise authorizes employees after authentication
- **Cross-organizational and open environments**:
  - No prior trust relationships
  - Manual approaches (local accounts, X.509 lists) are:
    - Administratively expensive
    - Unscalable
    - Unmanageable for dynamic relationships
- **Ad hoc authorization scenarios**:
  - Senior citizen discounts
  - Family/friends accessing photo albums
  - Requirements unrelated to organizational affiliation
  - Identity-based approaches fail
- **Emergence of Trust Management (TM)**:
  - Enables **on-the-fly trust establishment**
  - Authorization based on attributes instead of identity
  - Key early contributions:
    - Bina et al.: attribute-based authorization using certificates
    - Blaze et al.: delegation-based authorization; coined *trust management*
    - SPKI (Ellison et al.)
    - SDSI (Rivest et al.): naming and key binding
- **Digital credentials in TM**:
  - Cryptographically signed
  - Convey authorization-relevant information

    - Used to determine compliance with access control policies
- **Two additional challenges**:
    1. **Credential discovery**:
        - Credentials issued and stored in a decentralized manner
    2. **Credential and policy sensitivity**:
        - Credentials may contain confidential information
        - Policies may leak sensitive facts
        - Policies themselves require access control
- **Trust negotiation**:
    - Automated, bilateral trust establishment
    - Parties exchange credentials and policies incrementally
    - Supports main authorization decision plus secondary ones
    - Based on credentials
- **Relation to databases**:
    - Many TM systems based on **Datalog**
    - Authorization = query evaluation over distributed data
    - Credential and policy storage resembles distributed databases
- **Scope of the chapter**:
    - Focuses on **authorization-based trust management**
    - Not reputation or trustworthiness estimation systems
- **Chapter structure**:
    - Section 2: What is Trust Management
    - Section 3: History and systems
    - Section 4: Evaluation problems
    - Section 5: Trust negotiation
    - Section 6: Open issues and trends

3.2 What is Trust Management?

- **Traditional access control**:
    - Identity-based (ACLs, PKI)
    - Closed environments
    - All authorized identities known in advance
- **Decentralized environments**:
    - Unknown clients and peers
    - ACLs exclude legitimate users
    - Peer-to-peer systems require mutual trust
- **Trust Management approach**:
    - Delegation of authority
    - Resource owners rely on external authorities
    - Supports decentralized authorization
- **Digital credentials**:
    - Signed using public key cryptography
    - Verifiable by anyone with issuer's public key
    - X.509v3 is common, but alternatives exist

- Can include attributes (employee ID, license, age)
- **Monotonic authorization semantics**:
  - Adding credentials cannot revoke authorization
  - No negative evidence
  - Fail-safe behavior
  - Important in decentralized systems with incomplete information
- **Certificates vs credentials**:
  - PKI certificates bind keys to identities
  - TM credentials bind keys to **authorization-relevant attributes**
- **Capability-based roots**:
  - PolicyMaker and KeyNote resemble capabilities
  - Credentials grant specific access rights
  - Delegation chains prove authorization
- **Credential chains**:
  - Series of signed delegations
  - Each signed using previous key
  - Requester proves possession of final key
- **Attribute-based credentials**:
  - Used in later systems (SPKI/SDSI, RT, Cassandra)
  - Describe properties (student, citizen, age)
  - Enable scalable policies
- **Delegation idioms**:
  - **Delegation**: passing access rights
  - **Appointment**: granting attributes not possessed by appointer
  - **Threshold delegation (k-out-of-n)**:
    - Authority effective only with cooperation of k entities
- **Compliance checking (authorization query evaluation)**:
  - Determines if credentials satisfy policy
  - Requires credential chain discovery
  - Can be represented as paths in a credential graph
- **Trust negotiation**:
  - Runtime bilateral trust establishment
  - Credentials + explicit policies
  - Automated protocols

Fig. 1. Example trust negotiation

- Scenario: Alice purchases prescription medication from Bob's pharmacy
- Steps illustrated:
  - Bob discloses sales policy
  - Alice requests proof Bob is licensed pharmacist
  - Bob presents pharmacist credential
  - Alice verifies and sends prescription
  - Alice provides doctor license credential
  - Bob verifies prescription and patient identity
  - Access granted

- Demonstrates:
  - Mutual disclosure
  - Credential discovery
  - Privacy-preserving trust establishment
  - Automation via software agents

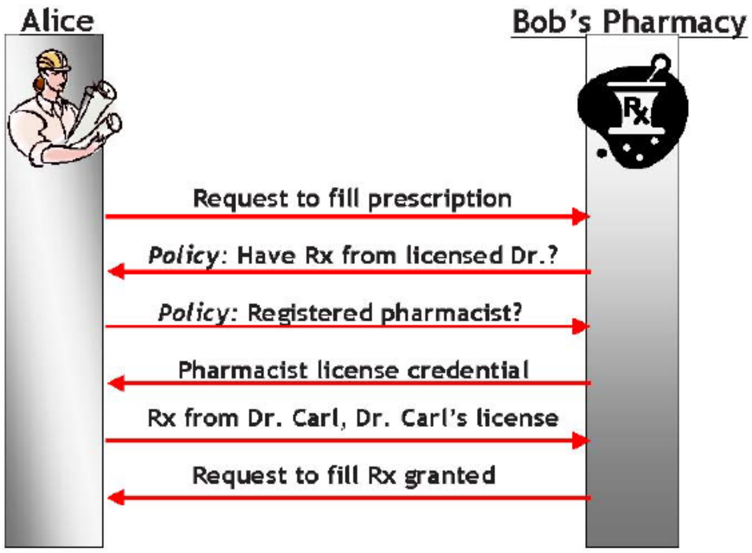

Fig. 1. Example trust negotiation

3.3 History

- Surveys major trust management systems
- Focuses on:
  - Features
  - Contributions
  - Intended applications

3.3.1 PolicyMaker and KeyNote

PolicyMaker

- First trust management system
- Proof-of-concept
- Assertions are programmable:
  - <Source> ASSERTS <AuthorityStruct> WHERE <Filter>
- Application-dependent semantics
- Does not perform cryptographic verification

KeyNote

- Successor to PolicyMaker
- Uses a fixed, human-readable assertion language
- Enforces cryptographic signature verification
- Binds public keys to authorizations
- Uses **action environment** (name–value bindings)

Fig. 2. An example KeyNote assertion

- Shows:
  - Authorizer public key
  - Licensees
  - Conditions on file and access type
  - Signature
- Demonstrates delegation and read access control

```
KeyNote-Version: 1
Authorizer: rsa-pkcs1-hex: “1234abcd”
Licensee: dsa-hex: “9876dcba” || rsa-pkcs1-hex: “6789defg”
Comment: Authorizer delegates read access to either of the licensees
Condition: ($file == “/etc/passwd” && $access == “read”) → {return “OK”}
Signature: rsa-md5-pkcs1-hex: “f00f5673”
```

Fig. 2. An example KeyNote assertion.

3.3.2 SPKI/SDSI

- Combines:
  - SDSI: naming infrastructure
  - SPKI: authorization certificates

SDSI Contributions

- **Local and extended names**
- Names bound to keys
- Avoids global naming conflicts
- Supports groups via name resolution

SPKI Contributions

- Authorization certificates:
  - Issuer
  - Subject (key or SDSI name)
  - Delegation bit
  - Authorization tag
- Enables dynamic group-based authorization

3.3 QCM and SD3

QCM (Query Certificate Manager)

- Designed for distributed data security
- Introduced **policy-directed certificate retrieval**
- Uses query decomposition and optimization

SD3

- Successor to QCM
- Verifies cryptographic signatures
- Supports distributed credential retrieval
- Uses extended Datalog with SDSI names
- Implements certified evaluations

3.3.4 RT

- Family of **Role-based Trust Management languages**
- Combines RBAC and TM
- Based on Datalog
- Supports:
  - Efficient query evaluation
  - Role hierarchies
  - Delegation
  - Constraints
- Credential types:
  - Simple Member
  - Simple Inclusion
  - Linking Inclusion
  - Intersection Inclusion
- Formal Datalog translation defines semantics
- Supports extensions: RT1, RT2, RTC, RTT, RTD
- **Application Domain Specification Documents (ADSDs)**:
  - Define role vocabulary
  - Ensure consistent semantics
  - Identified by URIs

3.3.5 OASIS and Cassandra

- Both are role-based TM systems

OASIS

- Introduced **appointment**
- Supports sessions
- Uses first-order logic

Cassandra

- Designed for electronic health records
- Uses Datalog with constraints
- Supports sessions and trust negotiation
- Uses predefined predicates for permissions and role activation

Fig. 3. The PCA system

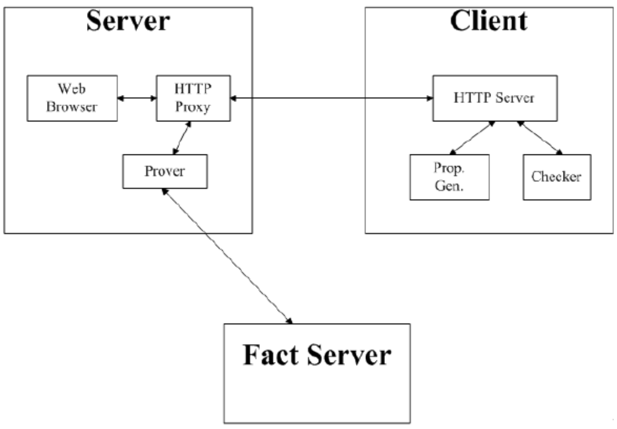

Fig. 3. The PCA system

- Shows:
    - Client proxy
    - Server
    - Proof generation on client side
    - Proof checking on server side

3.3.6 PCA

- **Proof Carrying Authorization**
- Designed for web services
- Client constructs proof
- Server checks proof
- Uses higher-order logic
- Avoids undecidable computation on server

Fig. 4. An example TPL rule

- Shows:
    - XML-based rule
    - Prolog translation
- Demonstrates rule portability and logic-based interpretation

XML:

$< \text{GROUP NAME} = \text{“}\texttt{Hospitals}\text{”} >$

$\quad < \text{RULE} >$

$\quad\quad < \text{INCLUSION ID} = \text{“}\texttt{reco}\text{”} \quad TYPE = \text{“}\texttt{Recommendation}\text{”}$

$\quad\quad\quad \text{FROM} = \text{“}\texttt{self}'' >< \backslash\text{INCLUSION} >$

$\quad < \backslash\text{RULE} >$

$< \backslash\text{GROUP} >$

Prolog:

$group(X, \texttt{Hospitals}) \quad : \quad - \quad cert(Y, X, \text{“}\texttt{Recommendation}\text{”}, \texttt{RecFields}),$
$group(Y, \texttt{self}).$

Fig. 4. An example TPL rule shown in its concrete XML syntax and its internal Prolog representation.

3.3.7 TPL

- Trust Policy Language
- Designed for strangers
- Maps users to roles
- Uses XML syntax
- Certificate-format independent
- Supports non-monotonic policies
- Uses credential collectors for negative credentials

3.4 Evaluation Problems and Strategies

- Key issues:
    1. Separation of authorization from applications
    2. Special-purpose policy languages

3. Credential discovery and retrieval

- Credential storage is decentralized
- Evaluation can be:
  - Distributed
  - Centralized with remote retrieval

3.4.1 General-Purpose Query Evaluation Engine

- Introduced by PolicyMaker
- Advantages:
  - Reusability
  - High assurance
  - Formal correctness
- Separates application logic from authorization logic

3.4.2 Efficiency and Expressivity

- PolicyMaker:
  - Highly expressive
  - Undecidable compliance checking
- LBPOC (Locally Bounded Proof of Compliance):
  - Polynomial restrictions
  - Bounded execution resources
- Declarative approaches preferred:
  - SPKI/SDSI: polynomial-time closure
  - KeyNote: monotonic but undecidable authorization set
  - SD3 and RT: Datalog-based
  - Cassandra: constraint-based with groundness analysis
  - PCA: requester constructs proof
  - LolliMon: higher-order linear logic

3.4.3 Credential Retrieval Mechanisms

- Early systems assumed all credentials provided
- Modern systems support **automatic credential discovery**
- Avoids duplication of effort
- TM engine assists in:
  - Identifying missing credentials
  - Retrieving them
- Two approaches:
  1. Remote evaluation via queries (QCM, SD3)
  2. Remote credential fetching (RT)

Fig. 5. QCM system

- Illustrates:
  - Distributed credential repositories
  - Policy-directed retrieval
  - Cooperative TM engines

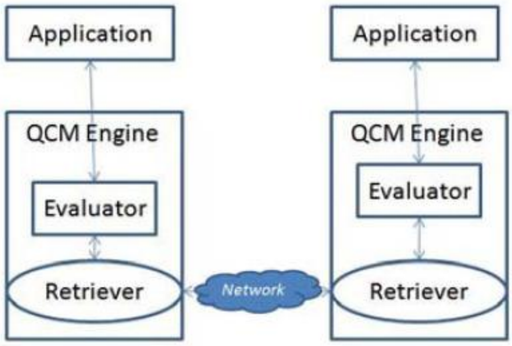

Fig. 5. QCM system

3.4.4 Distributed Evaluation

- **QCM** was the **first trust management (TM) system** to integrate **credential retrieval** directly into the evaluation engine.
- When local credentials are insufficient:
  - Queries are forwarded to **remote TM engines**
  - These engines belong to principals whose assertions may satisfy the query
- **Figure 5** (referenced):
  - Illustrates QCM's distributed architecture
  - Shows query propagation among multiple credential repositories

QCM Evaluation Modes

- **Verify-only mode**:
  - Credential retrieval disabled
  - Used to verify that returned credentials solve the query
- **Verify-retrieval mode**:
  - Enables remote credential discovery
- Credential retriever:
  - Shares code with the evaluator
  - Does not significantly increase code size

Types of Remote Replies

- When a remote engine is queried, it may return:
  - **Extensional answers**:
    - Direct reply
    - Table of tuples satisfying the query
    - Often packaged as a newly signed credential
  - **Intensional answers**:
    - Proof
    - Partial proof
    - Set of credentials from which an answer can be deduced
- In **SD3 terminology**:
  - Direct replies → extensional
  - Proof-based replies → intensional
- TM engines returning answers to applications should return **extensional answers** only

Quality of Service in SD3

- SD3 allows servers to choose response types:
    - **High level**: full evaluation + direct reply
    - **Medium level**: partial proof + hints to other servers
    - **Low level**: locally available credentials only
- Helps prevent:
    - Bottlenecks
    - Denial-of-service attacks

Integrity and Authenticity of Replies

- Replies are **digitally signed**
- Two signing approaches:
    - **On-line signing**:
        - Server signs answers dynamically
    - **Off-line signing**:
        - Server returns pre-signed credentials
        - Protects private keys
        - Requires synchronization with off-line signer
- Intensional answers:
    - Typically require verification of more signatures
- **QCM optimization**:
    - Uses **hash trees**
    - Reduces signing and verification overhead
- QCM allows servers to choose either signing method

Cyclic Dependencies in Distributed Evaluation

- Cycles among predicate definitions may cause:
    - Repeated subqueries
    - Nontermination
- Solutions:
    - **QCM**:
        - Uses time-outs to detect cycles
        - Risk: legitimate queries may be denied
    - **SD3**:
        - Tags queries with waiting sites
        - Detects cycles explicitly
        - Costly in bandwidth and processing

3.4.5 Local Evaluation with Distributed Credentials

- **QCM and SD3**:
    - Credentials issued, revoked, and stored in a **distributed manner**
    - Credentials stored with **issuers**
    - Ensures discoverability under basic availability assumptions

Limitations of Issuer-only Credential Storage

- In practice, some credentials should be stored with **subjects**
- Example:

- Student discounts:
  - University issuing credentials may not want to serve every verification request
  - Privacy concerns: student controls disclosure

RT Approach to Distributed Credentials

- **RT** allows credentials to be stored:
  - With issuers
  - With subjects
- Evaluation is performed **locally**
- Remote servers only provide credentials
- Authorization modeled as a **graph**:
  - Nodes: role expressions
  - Edges: credentials and derived relationships
  - **Chains** = proofs of authorization

Graph-based Evaluation

- Query: Is *D* a member of *A.r*?
- Process:
  - Add nodes for *D* and *A.r*
  - Extend graph using relevant credentials
- Problem:
  - If credentials are not stored by principals named in nodes, they may be undiscoverable

Example (from [49])

- **Table 1** (referenced):
  - RT_0 credentials describing:
    - EPub discounts
    - University accreditation
    - Student enrollment

(1) FAB.university ⟵ StateU
(2) RegistrarB.student ⟵ Alice
(3) EPub.discount ⟵ EOrg.preferred
(4) EOrg.university ⟵ FAB.accredited
(5) StateU.student ⟵ RegistrarB.student
(6) EOrg.preferred ⟵ EOrg.university.student

Table 1.

- Credential chain:
  - Part (a): EPub.discount ← EOrg.preferred ← EOrg.university.student
  - Part (b): EOrg.university ← FAB.accredited ← StateU
  - Part (c): StateU.student ← RegistrarB.student ← Alice
- Demonstrates necessity of:
  - Storing some credentials with subjects (Alice)
  - Storing others with issuers (EOrg, FAB, RegistrarB)

Credential Discovery Problem

- If credentials are stored arbitrarily:
  - Evaluation may fail even if authorization is valid
- RT solution:
  - **Type system for credentials**
  - Role names assigned types
  - Types indicate where credentials should be stored
- **Well-typing rules**:
  - Ensure all credentials in a chain are discoverable
- Extended from **RT_0 to full RT**

PeerAccess Solution

- Uses **brokers, issuers, and subjects**
- Supports **proof hints**
- Brokers act like:
  - Search engines for credentials
- Can encode retrieval strategies of:
  - QCM
  - SD3
  - RT

3.5 Automated Trust Negotiation

- Many algorithms exist for runtime trust establishment
- Examples include:
  - Unipro, Cassandra, Trust-χ, PeerAccess, idemix
- Common advantages over identity-based access control:
  - Dynamic bilateral trust
  - Attribute-based authorization
  - No reliance on centralized trusted third parties
  - Gradual disclosure of credentials
  - Possible authorization without revealing exact attributes

Policy Language Requirements

- Well-defined semantics
- **Monotonicity**:
  - Additional credentials should not reduce trust
- Support for:
  - Conjunction, disjunction
  - Transitive closure
  - Attribute constraints
  - Multi-credential constraints (joins)

3.5.1 Supporting Autonomy during Trust Negotiation

- Parties are autonomous:
  - Free to choose actions and strategies
- Negotiation defined by:
  - **Protocol**: message types and ordering rules

    - **Strategy**: local decision-making algorithm

Disclosure Strategies

- Disclose all credentials
- Disclose all relevant credentials
- Disclose minimal credentials:
    - Based on set inclusion
    - Based on weighted sensitivity
- Cryptographic strategies:
    - Prove policy satisfaction without revealing credentials
- Hybrid strategies:
    - Direct disclosure for low sensitivity
    - Cryptographic methods for high sensitivity

Strategy Interoperability

- Parties need only agree on:
    - A **set of interoperable strategies**
- Guarantees:
    - Negotiation succeeds if authorization is possible
    - Preserves autonomy
    - Supports sensitive policies

3.5.2 Avoiding Information Leakage during Trust Negotiation

Leakage Risks

- Credentials reveal unnecessary attributes
- Behavior reveals possession or absence of credentials
- Attribute values inferred via repeated queries
- **Need-to-know attacks**:
    - Harvest credentials through policy rewriting

Protection Techniques

- Selective disclosure credentials
- Cryptographic proof systems:
    - Prove properties without revealing values
- Non-response to sensitive queries
- **Acknowledgement policies (ack-policies)**
- Attribute-based requests instead of credential-based
- Ontologies to choose least revealing credentials

Limitations

- Direct disclosure offers no guarantee against redistribution
- PeerAccess enforces disclosure policy propagation
- P3P policies can be checked, but not enforced
- Most secure solution:
    - Cryptographic TN without direct disclosure

3.5.3 Trust Negotiation Implementations

- Most TN systems are:
    - Proofs of concept

    - Not production-ready
- Publicly available systems:
    - TrustBuilder
    - TrustBuilder2
    - Trust-χ

TrustBuilder2 Framework

- Java-based, modular TN framework
- Components:
    - Strategy modules
    - Compliance checkers
    - Query interfaces
    - Audit modules
- **Figure 6**:
    - Shows internal structure of TrustBuilder2 agent
- Supports:
    - Plug-ins
    - Multiple policy languages
    - Multiple credential formats
- Enables controlled experiments and component comparison

3.6 Open Issues and Trends

3.6.1 Policy Engineering and User Interfaces

- Policy specification is error-prone
- Empirical studies show:
    - Low correctness of user-written policies
- Research needs:
    - Policy debugging tools
    - User-friendly languages
    - Explanation of authorization decisions
    - Policy compilation from high-level abstractions
- Use of **default policies** from credential issuers

3.6.2 Real-world Trust Negotiation Deployments

- TN adoption is limited
- Major challenges:
    - Scalability
    - Security hardening
    - Real-world deployment experience
- Need for small-scale production deployments

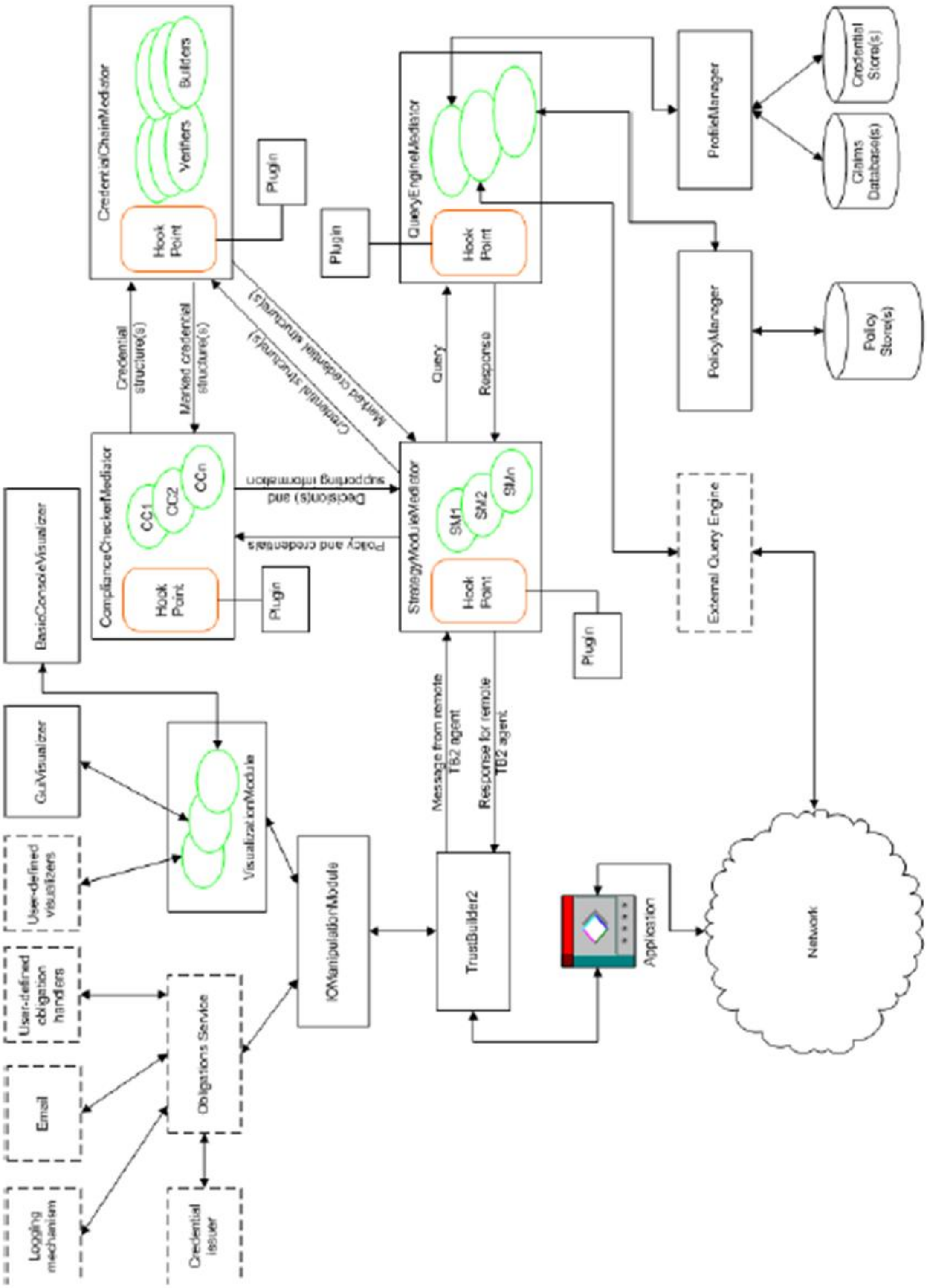

Fig. 6. Internal structure of a TrustBuilder2 trust negotiation agent

3.6.3 Distributed Proof Construction

Key Challenges

- **Autonomy**:
  - Parties use different proof strategies
- **Sensitive information**:

    - Proof fragments may be confidential
    - Need controlled disclosure
- **Non-monotonicity**:
    - Negative conditions and environmental changes
    - Time- and context-dependent authorizations
- Distributed proof construction remains a **major open research problem**

4. Security in Data Warehouses and OLAP Systems

4.1 Introduction

- Rapid growth in **computer and network technologies** has enabled organizations to collect, store, and analyze **large volumes of data**.
- **Data warehouses** and **OLAP (On-Line Analytical Processing) systems** are central to organizational decision-making.
- **Security importance**:
    - Leakage of organizational secrets → **financial and competitive damage**
    - Indiscriminate data collection → **privacy violations**
    - Government data warehouse breaches → **high-impact losses (e.g., national security)**

Unique Security Challenges in OLAP

- OLAP systems primarily support **decision support**, not transactional access.
- They rely heavily on **data aggregation** to:
    - Hide insignificant details
    - Highlight global patterns and trends
- **Data cube model [15]** organizes multidimensional aggregates via **dimension hierarchies**.
- **Primary threat**:
    - **Insiders** with legitimate access using OLAP queries to infer sensitive data.
- Traditional methods are insufficient:
    - **Data sanitization** → vulnerable to **linking attacks**
    - **Access control** → not directly applicable due to **different data models**
    - **Inference control** → computationally expensive and impractical for online OLAP

Motivation and Chapter Overview

- Aggregation does not fully destroy sensitive information.
- Remaining information + external knowledge → **indirect disclosure**.
- Online inference control is considered impractical due to:
    - High computational cost
    - Continuous tracking of query history
- **Offline inference control** (e.g., census tables) proves inference threats are real.

Chapter goals:

- Demonstrate inference threats via OLAP queries
- Identify security requirements
- Review:

    - Two inference-control extensions for OLAP
    - A **preventing-then-removing** approach
- Show applicability to a broad range of aggregation types

Chapter organization:

- Section 2: Background
- Section 3: Inference threats and security requirements
- Section 4: Three-tier security architecture
- Section 5: Inference control methods
- Section 6: Conclusion

4.2 Background

4.2.1 Data Warehouses and OLAP Systems

Data Warehouses

- Centralized repositories storing enterprise data
- Organized using:
    - **Star schema**: fact table + dimension tables
    - **Snowflake schema**: normalized dimension tables
- Data characteristics:
    - Integrated from multiple sources
    - Cleaned and transformed
    - Periodically refreshed (not continuously updated)
- **Data marts**: subsets tailored to specific organizational needs

OLAP Systems

- Coined by **Codd et al., 1993 [9]**
- Used for **business data analysis** (sales, healthcare, etc.)
- Key goals:
    - Interactive exploration
    - Multi-perspective analysis
    - Multiple levels of generalization

OLAP Architectures

- **ROLAP**: Multidimensional queries → SQL on relational DB
- **MOLAP**: Materialized multidimensional views
- **HOLAP**: Hybrid of ROLAP and MOLAP

OLAP vs Data Mining

- Data mining: automatic pattern discovery
- OLAP: human-driven pattern interpretation
- OLAP is more flexible
- Both can be combined for enhanced analysis

OLAP Requirements

- Defined via **FASMI test** and **Codd rules**
- Relevant requirements:
    - High query efficiency
    - Pre-computation and indexing
    - Multidimensional generalization via **data cubes**

Data Cube Model

- Introduced to support **sub-totals and histograms** efficiently
- Avoids complex SQL queries with exponential unions

Explanation of Figure 1: An Example of Data Cubes

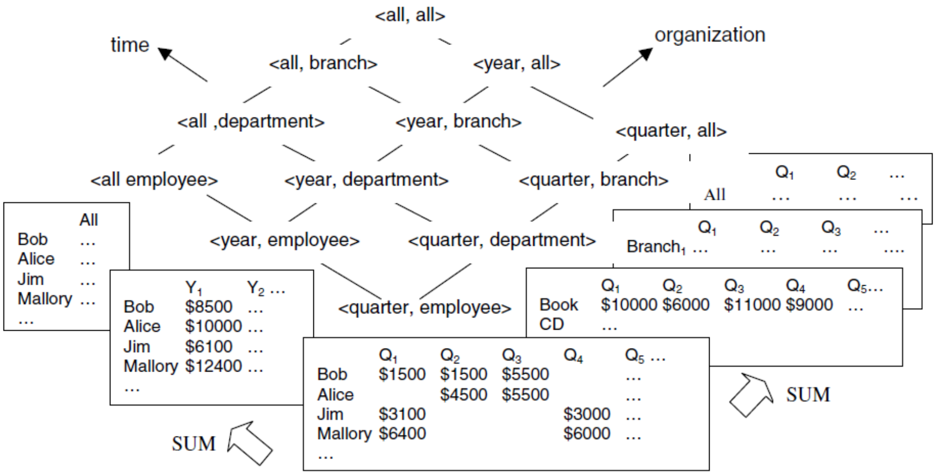

Fig. 1. An Example of Data Cubes

- Dimensions:
  - **Time** (quarter → year → ALL)
  - **Organization** (employee → department → branch → ALL)
- Each dimension forms a **dependency lattice**
- Product of lattices → **cuboid lattice**
- Each cuboid <T, O> is a 2D array of cells <t, o>
- Cells also form a dependency lattice

Cube Population

- Base table schema: (quarter, employee, commission)
- Core cuboid: <quarter, employee>
- Aggregation function: **SUM**
- Empty cells treated as zero (depending on aggregation)
- Example:
  - Cell <Y1, Bob> = sum of <Q1–Q4, Bob> = 8500

4.2.2 Related Work

- OLAP security is underdeveloped compared to relational DBs
- **Access control models**:
  - DAC, RBAC, FAF
- **Inference control techniques**:
  - Restriction-based (deny unsafe queries)
  - Perturbation-based (add noise)
- **Cell suppression & partitioning**:
  - Effective for 2D tables
  - Intractable for higher dimensions
- **Privacy-preserving data mining**:
  - Uses perturbation

- Not suitable for precise OLAP insights
- **k-anonymity**:
  - Ensures each record indistinguishable from k−1 others
- **Information-theoretic approaches**:
  - Too strict for aggregation-based disclosure

4.3 Security Requirements

4.3.1 The Threat of Inferences

- OLAP users can infer sensitive data from legitimate queries
- **1-d inference**: inferred from exactly one descendant
  - *Example 1*: <Q5, Bob> inferred from <Q5, Book>
- **m-d inference**: inferred from multiple descendants
  - *Example 2*: SUM-based inference using <year, employee> and <quarter, department>
  - *Example 3*: MAX-based inference even without outbound knowledge
  - *Example 4*: Combined SUM, MAX, MIN allows full reconstruction

4.3.2 The Requirements

A practical OLAP security solution must balance:

- **Security**: Prevent unauthorized access and indirect inference
- **Applicability**: Avoid unrealistic assumptions
- **Efficiency**: Maintain interactive query performance
- **Availability**: Avoid unnecessary restrictions
- **Practicality**: Minimal changes to existing systems

Core challenge: Trade-off between provable security and system efficiency

4.4 A Three-Tier Security Architecture

Motivation

- Two-tier (data + queries) inference control:
  - High runtime cost
  - Ignores OLAP characteristics

Explanation of Figure 2: Three-Tier Inference Control Architecture

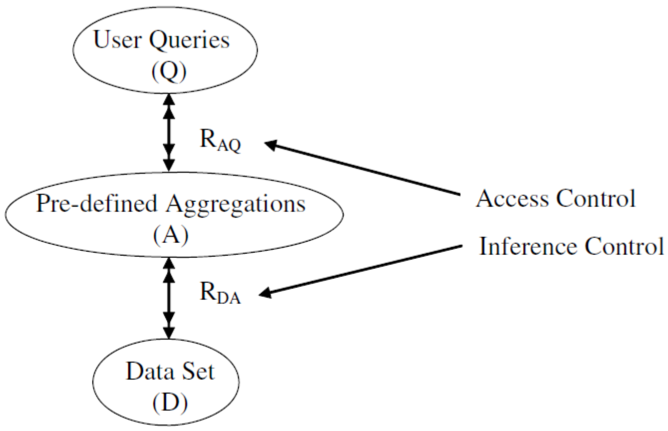

Fig. 2. A Three-Tier Inference Control Architecture

- Introduces **aggregation tier** between data and queries
- Three tiers:
  1. Data tier
  2. Aggregation tier
  3. Query tier

Properties of Aggregation Tier

1. Secure w.r.t. data tier via inference control
2. Comparable in size to data tier
3. Partitioned into blocks for localized inference control

Benefits

- Heavy inference checking done **offline**
- Online phase reduced to **access control**
- Exploits **materialized views (data cubes)**

4.5 Securing OLAP Data Cubes

4.5.1 SUM-only Data Cubes

Cardinality-Based Method

- Inference depends on:
  - Number of empty cells
- Uses **linear equations** and **RREF**
- **Figure references**:
  - Table 1: Core cuboid modeling
  - Table 2: Aggregation equations
  - Table 3: RREF showing inferable variable

Table 1. Modeling A Core Cuboid

	Q_1	Q_2	Q_3	Q_4	ALL
Bob	x_1	x_2	x_3		8500
Alice		x_4	x_5		10000
Jim	x_6			x_7	6100
Mallory	x_8			x_9	12400
ALL	10000	6000	11000	9000	36000

Table 2. Modeling the Aggregation Cuboids

$$\begin{pmatrix} 1&1&1&0&0&0&0&0&0 \\ 0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0 \\ 0&0&0&0&0&0&0&1&1 \\ 1&0&0&0&0&1&0&1&0 \\ 0&1&0&1&0&0&0&0&0 \\ 0&0&1&0&1&0&0&0&0 \\ 0&0&0&0&0&0&1&0&1 \\ 1&1&1&1&1&1&1&1&1 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{pmatrix} = \begin{pmatrix} 8500 \\ 10000 \\ 6100 \\ 12400 \\ 10000 \\ 6000 \\ 11000 \\ 9000 \\ 36000 \end{pmatrix}$$

Table 3. The Reduced Row Echelon Form M_{rref}

$$\begin{pmatrix} 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&-1&0&0&0&0 \\ 0&0&1&0&1&0&0&0&0 \\ 0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&-1 \\ 0&0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0 \end{pmatrix}$$

Key results:

- No empty cells → inference-free
- Too many empty cells → inference unavoidable
- Middle range → requires checking

Parity-Based Method

- Uses **even-numbered MDR queries**
- Converts inference problem to **graph bipartiteness**
- **Odd cycle ⇒ inference**
- Efficient detection via BFS

4.5.2 Generic Data Cubes

- Overcomes SUM-only limitation
- Uses **preventing-then-removing** strategy

Access Control

- Sensitive data may exist in aggregation cuboids
- Introduces:
    - Below() for lattice-based partitioning
    - Slice() for dimensional partitioning
- Authorization objects include ancestors

Lattice-Based Inference Control

Explanation of Figure 3: Preventing m-d Inferences

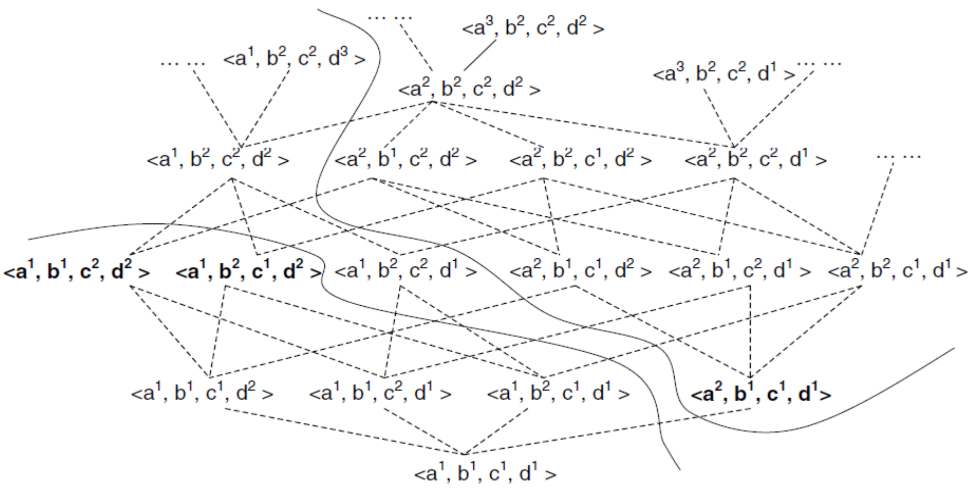

Fig. 3. An Example of Preventing m-d Inferences

- Uses **descendant closure**
- Only one minimal descendant allowed
- Prevents m-d inferences without detection
- Iterative removal of 1-d inferences

Theorem 3

- Descendant closure is:
    - Necessary
    - Sufficient
    - Maximal for preventing m-d inferences

4.6 Conclusion

- OLAP systems face serious **inference-based security threats**
- Traditional inference control is insufficient
- Three methods reviewed:

    1. Cardinality-based
    2. Parity-based
    3. Lattice-based (most general)
- **Three-tier architecture** is key for efficiency
- Preventing-then-removing approach avoids infeasible inference detection
- Provides a promising direction for **secure OLAP systems and data warehouses**