Trust Management

Vassilis Prevelakis

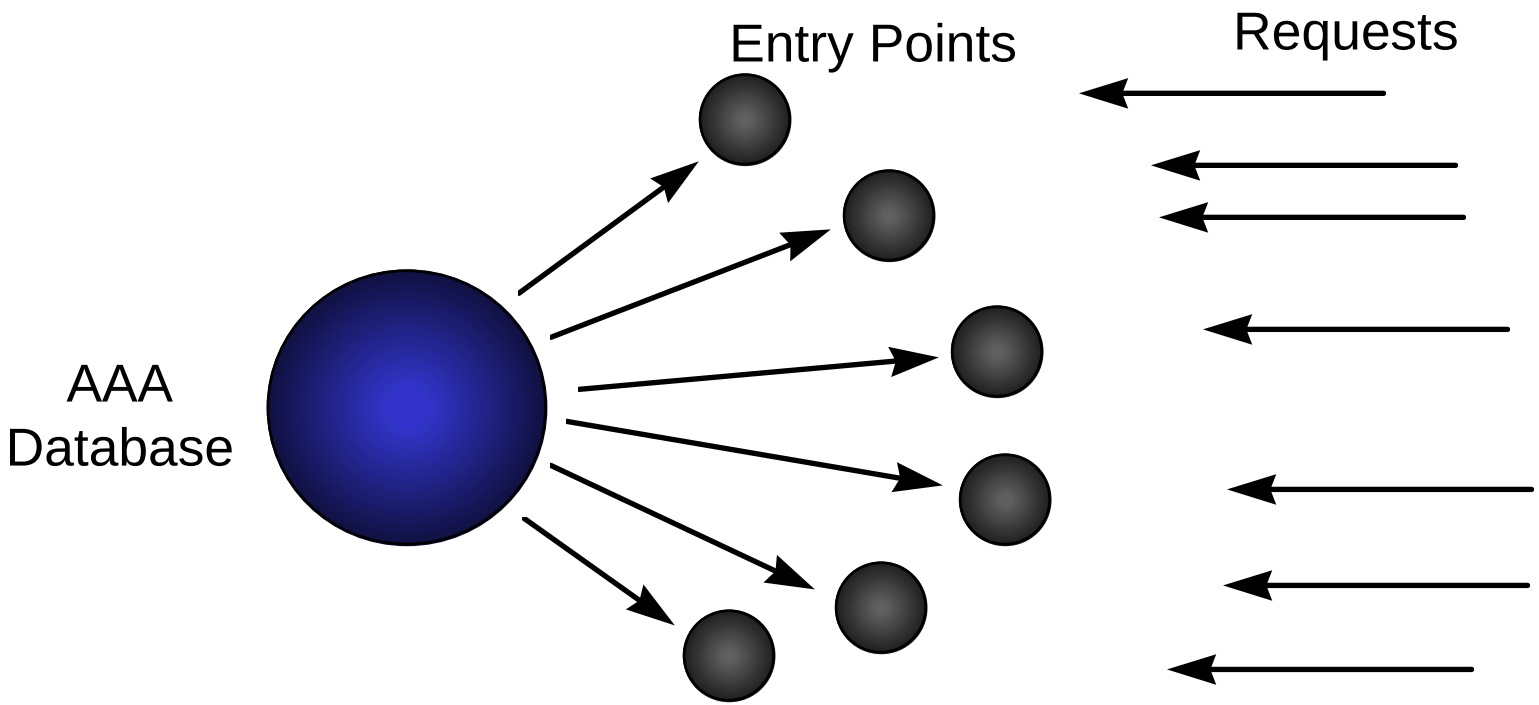
Computer Science Department

Drexel University

Concepts

- Authentication and Access Control
  - authentication means binding an “entity” to a “name”
  - access control means determining whether the “name” can access some resource or perform an action
- => Need to maintain
  - identity information
  - access control information
- Example
  - entering a building vs
  - using an ATM (off line case)

Authentication and Access Control



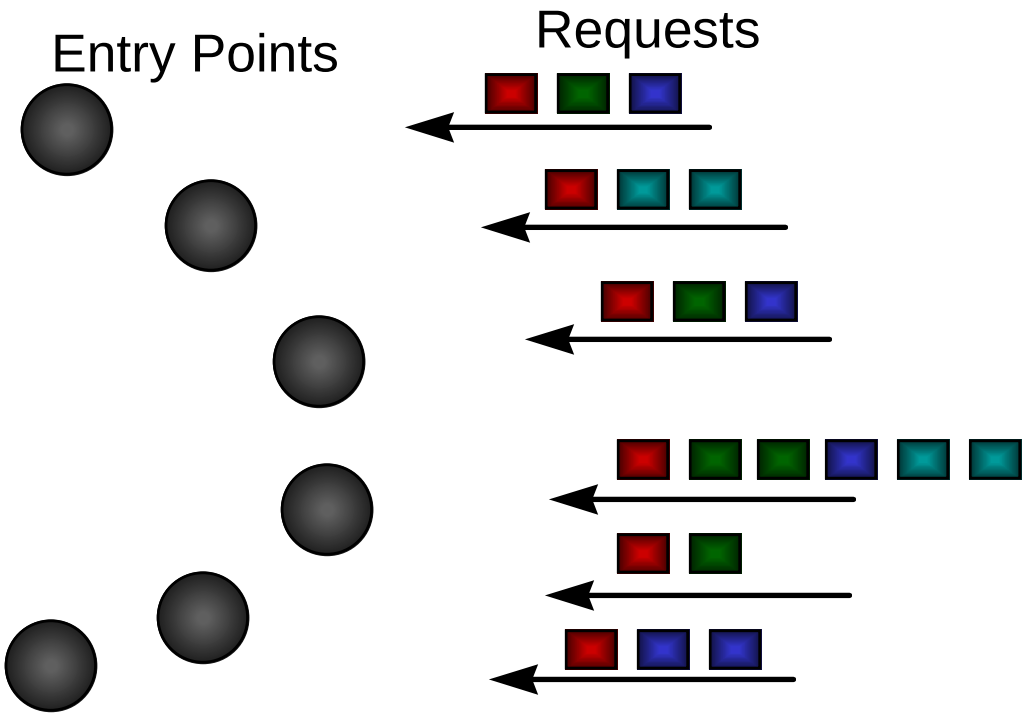
- How do we communicate AAA info to entry points

AAA = authentication authorization accounting

Authentication and Access Control

- How do we communicate AA info to entry points
  - use on-line system (e.g. Radius)
  - use Kerberos for authentication and ACLs for access control
  - maintain copies of AA info on entry points
- Problems
  - scalability
  - failure recovery
  - coordination (weakest link)

Trust Management



- submit all information with request (no need for AA database)

Trust Management

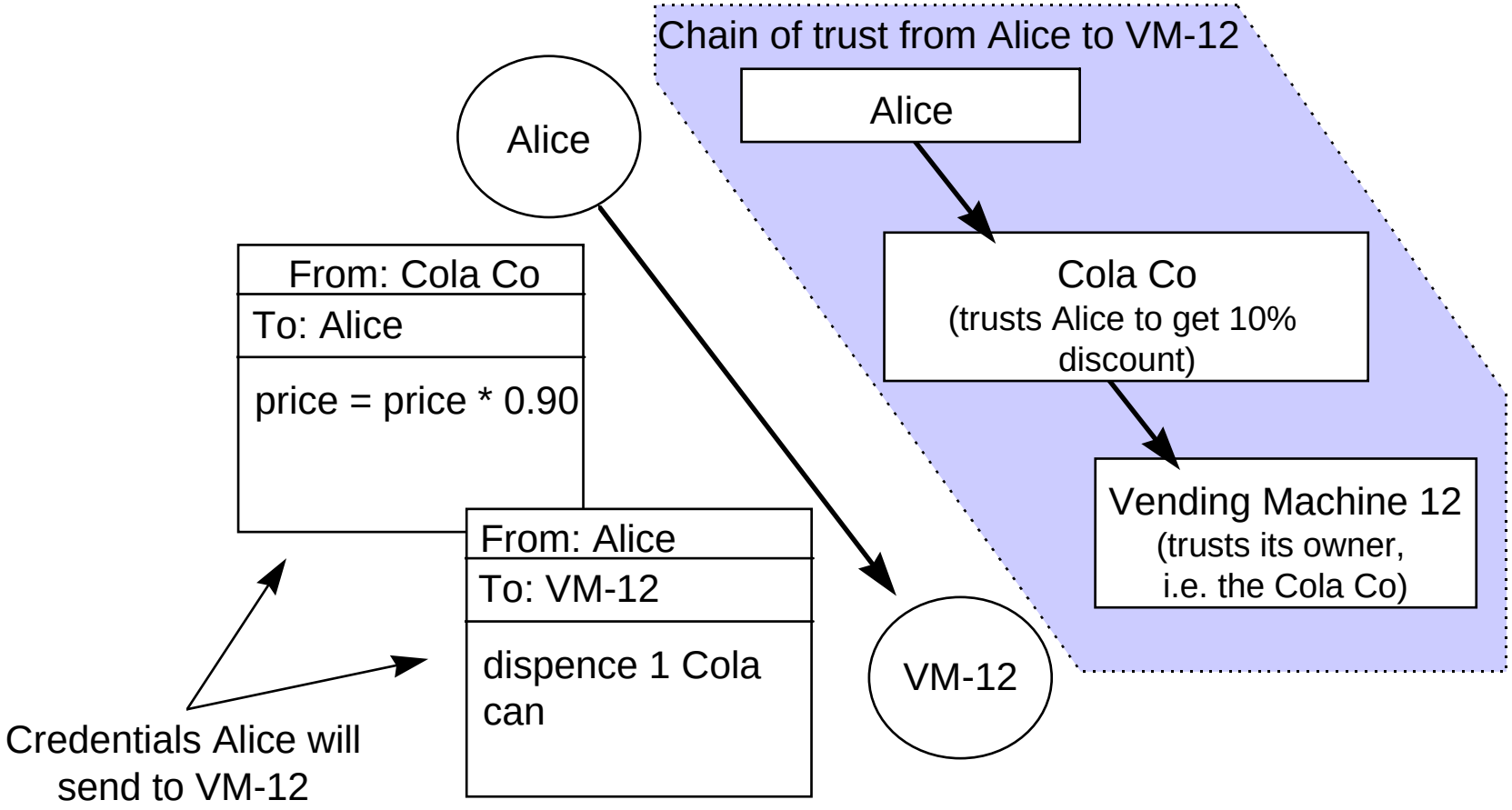
- How do we trust submitted information
  - entry point knows
    - the (public) key of the entity it trusts (e.g. its owner, operator etc.)
      - called the **root key**
    - what actions are acceptable (minimum policy -- more on this later)
  - requester has to establish a “chain of trust” between
    - the (private) key of the requester (used to sign the request)
    - the root key of the entry point
  - trust management credentials are used to
    - encapsulate permissible actions or privileges
    - bind them to a (public) key (licensee)
    - are issued (signed) by a key (authorizer)

Example

- Vendor wants to give discount to “preferred” customers
  - needs to instruct vending machines to give 10% discount to authorized users
  - usual problems
    - list of users not known in advance, lots of vending machines, etc.
- Vendor issues a credential to each “preferred” customer
  - command example (pseudo code)

    if (date < 20071231 && price > $1) then price = price * 0.9
  - Authorizer is vendor (signs credential)
  - Licensee is customer’s public key

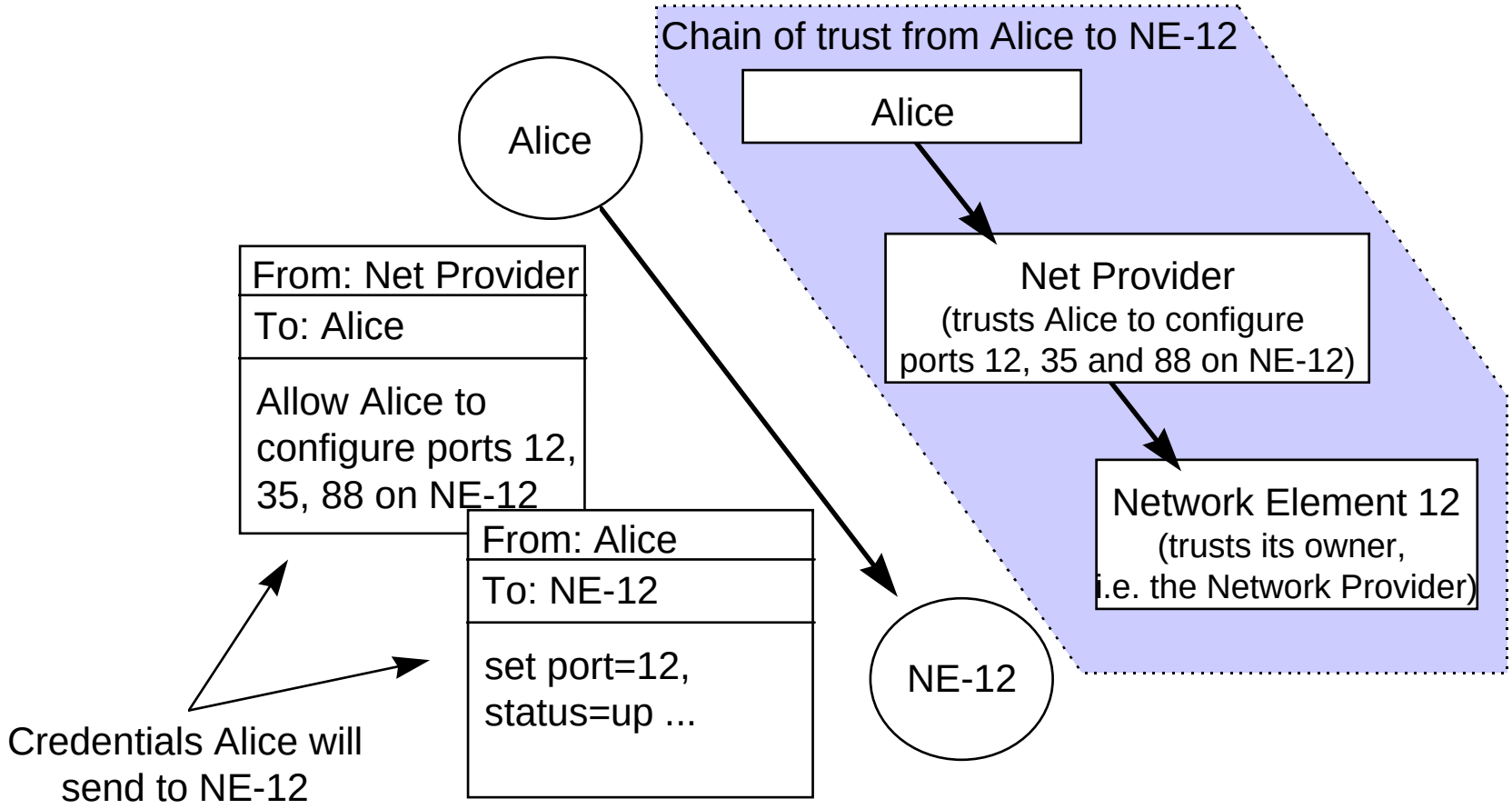
Trust Model


Chain of trust from Alice to VM-12
Alice
Cola Co
(trusts Alice to get 10% discount)
Vending Machine 12
(trusts its owner, i.e. the Cola Co)
Alice
From: Cola Co
To: Alice
price = price * 0.90
From: Alice
To: VM-12
dispence 1 Cola can
VM-12
Credentials Alice will send to VM-12

Example

- Customer establishes session with vending machine
  - any mechanism is acceptable
  - can deal with issues such as replay attacks, secrecy etc.
- Customer sends request with discount credential
- Vending machine
  - matches request signature with public key included in vendor credential
  - evaluates request
  - declares outcome (final sale price)
  - note that payment can be effected using trust management (micropayments) or other traditional form (e.g. coins)

Trust Management for Network Control



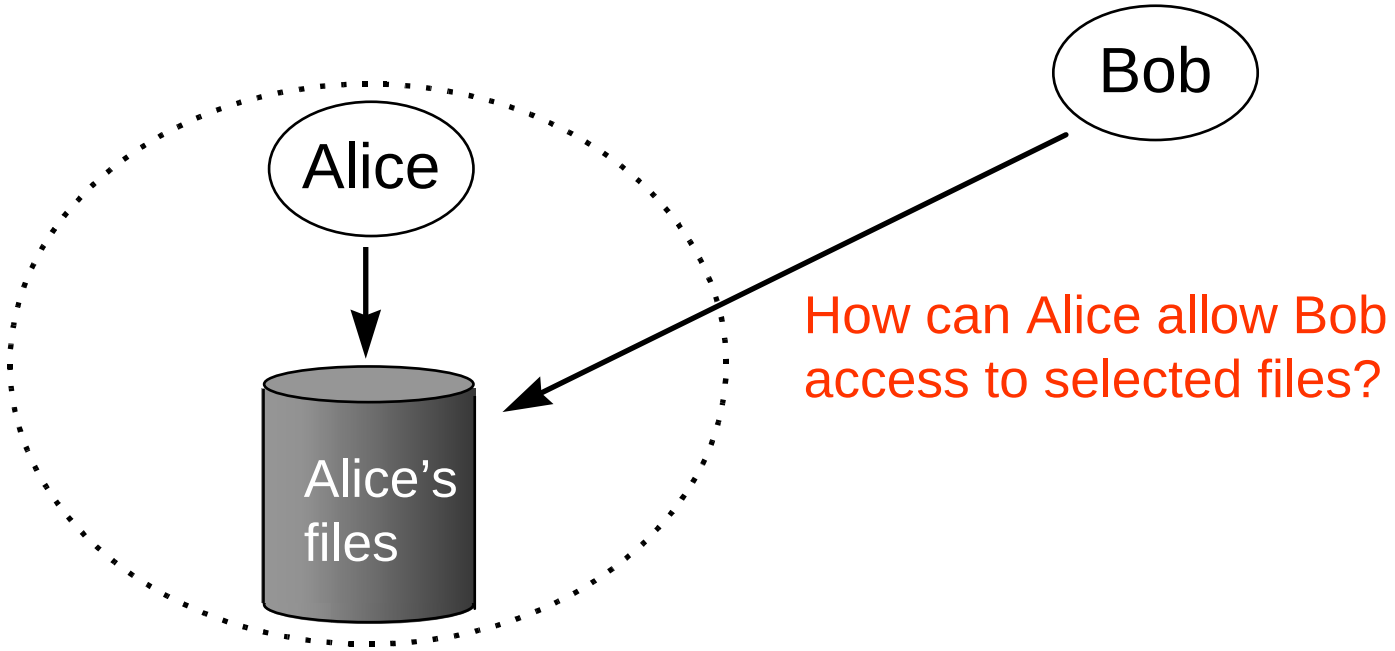
Trust Management Benefits

- Uniformity (everything is policy)
  - both global and per-credential privileges are expressed in the same way
- Scalability
  - Each entry point can **decide on its own**, but decisions are **consistent**
  - **Requester** needs to collect and submit all necessary credentials (analogy with public sector)
  - Caching of policy can improve efficiency
- Security
  - Time expired credentials deal with issues such as revocation or policy changes
- Flexibility
  - expressibility (wide range of actions, privileges, etc)
  - (controlled) delegation

Example
Collaboration and the Internet

- Typical Requirements
  - Accommodate groups that cross geographical and administrative boundaries
  - Traditional tools developed for use in closed groups
  - Security and Privacy are extremely important
    - confidentiality
    - integrity
  - Techniques should work regardless of location or mode of operation

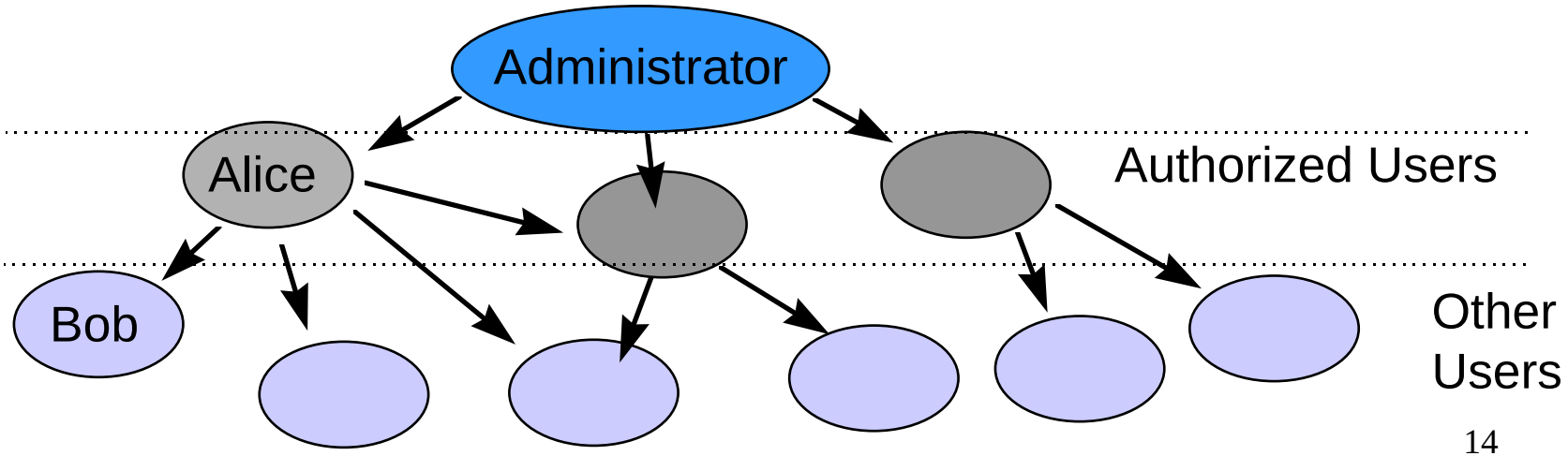
Collaboration across Domains



Alice wants to collaborate with Bob in writing a report.

WebDAVA Design

- Web-based: utilizes the http protocol
- Authorized users can delegate access to users that may be unknown to the system
- Flexible file access conditions
- Administrator can specify overall policy



KeyNote System

- KeyNote is a policy definition language
- Policy is encapsulated in signed credentials
- KeyNote Policy engine verifies and evaluates sequences of credentials granting access if a chain of trust can be established
- WebDAVA uses the KeyNote Trust Management System
  - Users issue KeyNote credentials
  - File access credentials are KeyNote-based

WebDAVA Credentials

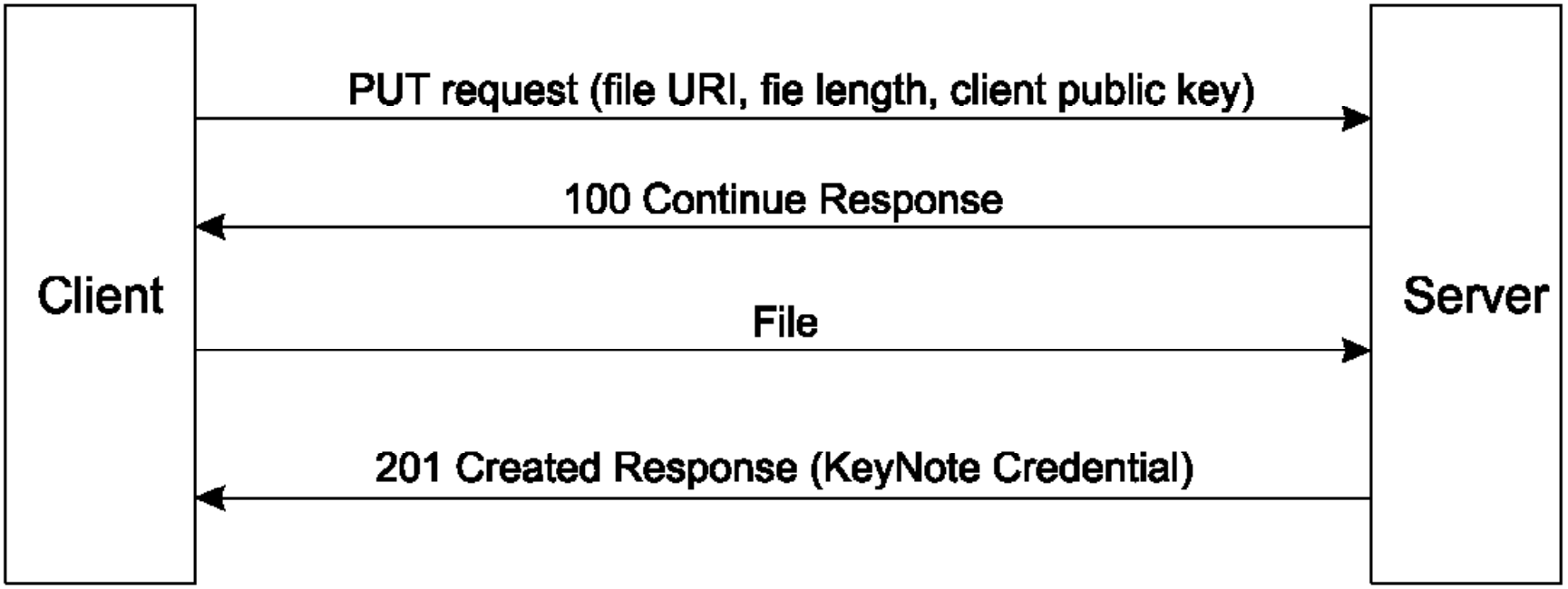
- Administrator gives Alice access to file

```
Authorizer: ADMINISTRATOR'S_PUBLIC_KEY
Licensees: ALICE'S_PUBLIC_KEY
Conditions: (AppDomain == "WebServer") &&
  (File_UID == "666240") -> "RWX";
Comment: Owner (Alice) can do anything!
  UID:666240
Signature: SIGNED_BY_ADMIN'S_PRIVATE_KEY
```

- Alice gives Bob R/O access to file

```
Authorizer: ALICE'S_PUBLIC_KEY
Licensees: BOB'S_PUBLIC_KEY
Conditions: (AppDomain == "WebServer") &&
   (localtime >= "20071115000001") &&
   (localtime <= "20071115235959") &&
   (method == "GET") &&
   (File_UID == "666240") -> "RWX";
Comment: UID:666240
Signature: SIGNED_BY_ALICE'S_PRIVATE_KEY
```

WebDAVA Protocol



Uploading file to server

Challenge-Response Scheme

- Protects against replay attacks
- Will be modified to provide protection against DoS attacks as well.
- Server sends **`401 Unauthorized`** message that includes

```
WWW-Authenticate: Keynote nonce=<nonce> \\
      server_key=<server_key>\r\n
```

The client then sends

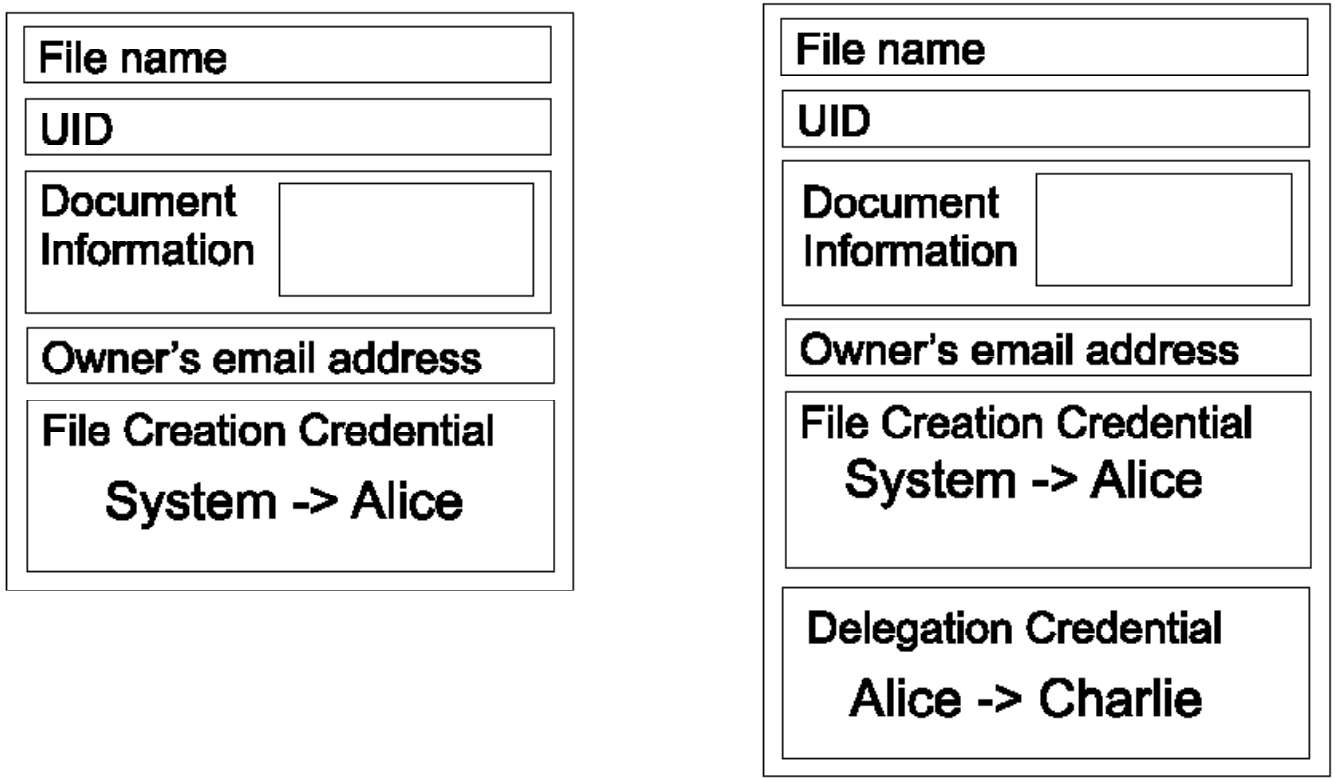
```
Authorization: client_key=<client_key> \\
      credential=<original_credential>\n\n \\
      <nonce_credential>\r\n
```

WebDAVA Prototype

- Java-based client resides on the client workstation
  - Communicates with server using the http protocol
  - Manages credentials
  - Signs requests with user's private key *(Why?)*

WebDAVA Prototype

- File Access Block (FAB) combine file access information in a compact request:



Key Exchange

- How does Alice find Bob's key?
  - Currently no PKI exists
  - DNSsec (once deployed) may be used to store and distribute keys
  - Interim Solution: provide a key-server application
    - users store their keys on the key server.
    - users download other people's keys from the key-server.

Discussion

- Revocation
  - general problem with capability-based systems
  - default policy may be used to exclude keys or restrict their power
  - provide a CRL on the server

Conclusions

- Traditional Access Control mechanism do not work with Web-based access.
  - *e.g.* UFS embeds policy with the file system
- WebDAVA Provides
  - portable client
  - works over http
  - provides scalable access control scheme that can cross admin domains
- Techniques demonstrated by WebDAVA can be applied to other disciplines, *e.g.* distributed file systems, OS resourse allocation strategies, wireless services *etc*.

BIG QUESTIONS

- Who writes the policies?
- Who selects the appropriate policy?
- How do we know the policy is correct?

AND MOST IMPORTANT ....

Human Accessible Security Policies (HASP)

- what happens when the *authorizer* is clueless ?
  - lots of clueless people in the world need to formulate policy
    - release personal information to airline company
    - provide access to medical data for treatment in hospital
    - delegate rights to subordinates to handle a specific task
    - allow electricity company access to turn off your air conditioner
    - relax security on PC to allow installation of software product
    - etc.
  - how do we deal with such issues?

HASP

- Consider analogy with software
  - in the “good old days” we had custom software
    - expensive (end-user needed to hire programmer)
    - home made (end-user needed to learn how to program)
  - now we have “shrink-wrapped” software
    - market dictates which is “best”
    - model accommodates open-source and commercial products
- Can we apply the same model to policies?
  - availability
  - trust
  - commercialization
  - usability

HASP (discussion)

- Availability: How can we make the policies available to end-users
  - on-line market place?
    - need to allow end-users to search for policies
    - how do end-users know they have a good match?
  - or, receive them from companies they deal with
    - analogy with contracts
- Trust: do we trust “shrink wrapped” policies?
  - create a link between implementation and description
  - “creative commons” approach
  - legal provisions

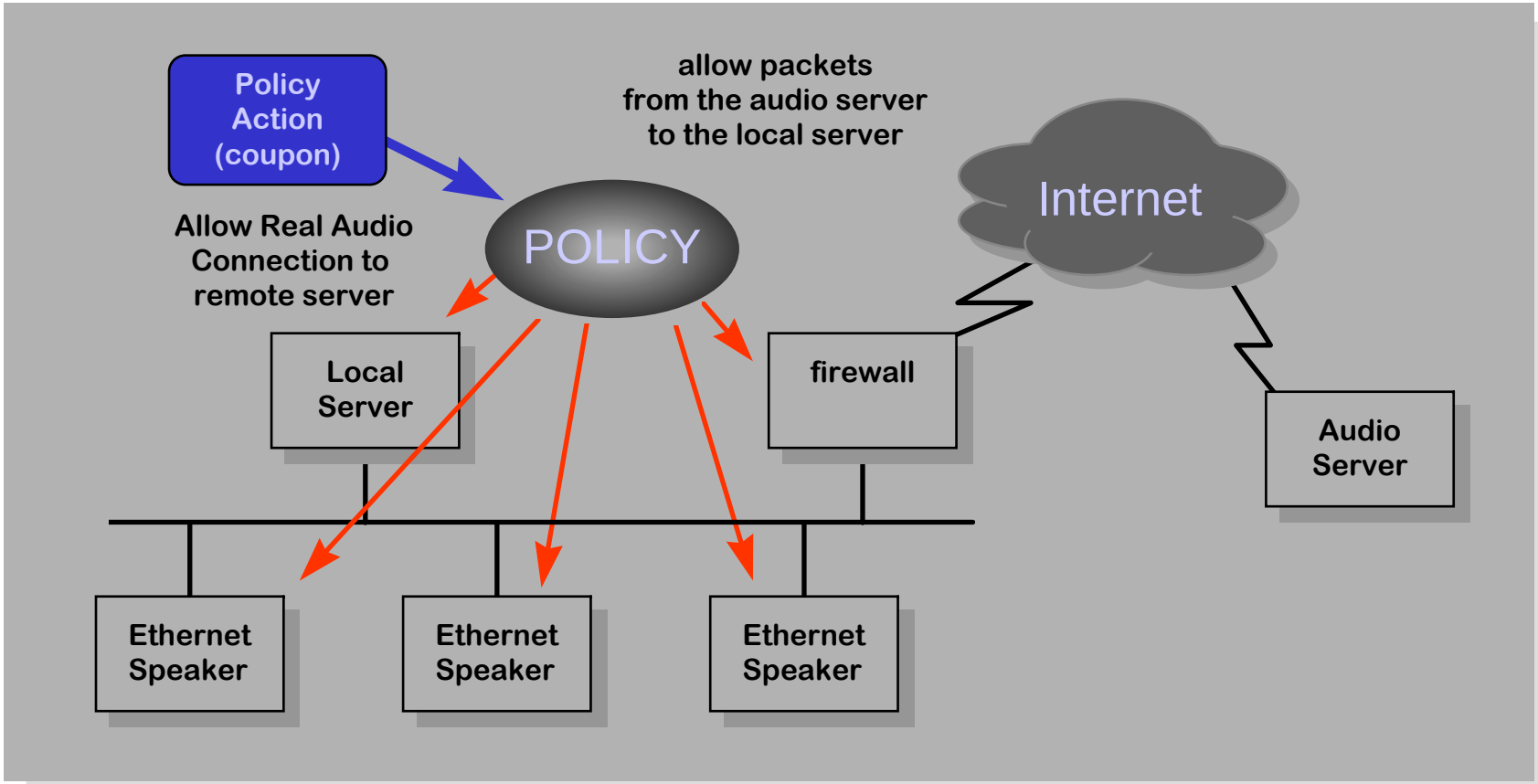
HASP (discussion)

- Commercialization (will the market adopt it?)
  - policies can be sold like software products
  - open-source policies
  - “value added services” e.g. notaries, consultants, attorneys
- How do we deal with policy conflicts
  - Policy Reconciliation
  - Ignore newer/older policy
  - Closely related to usability issue

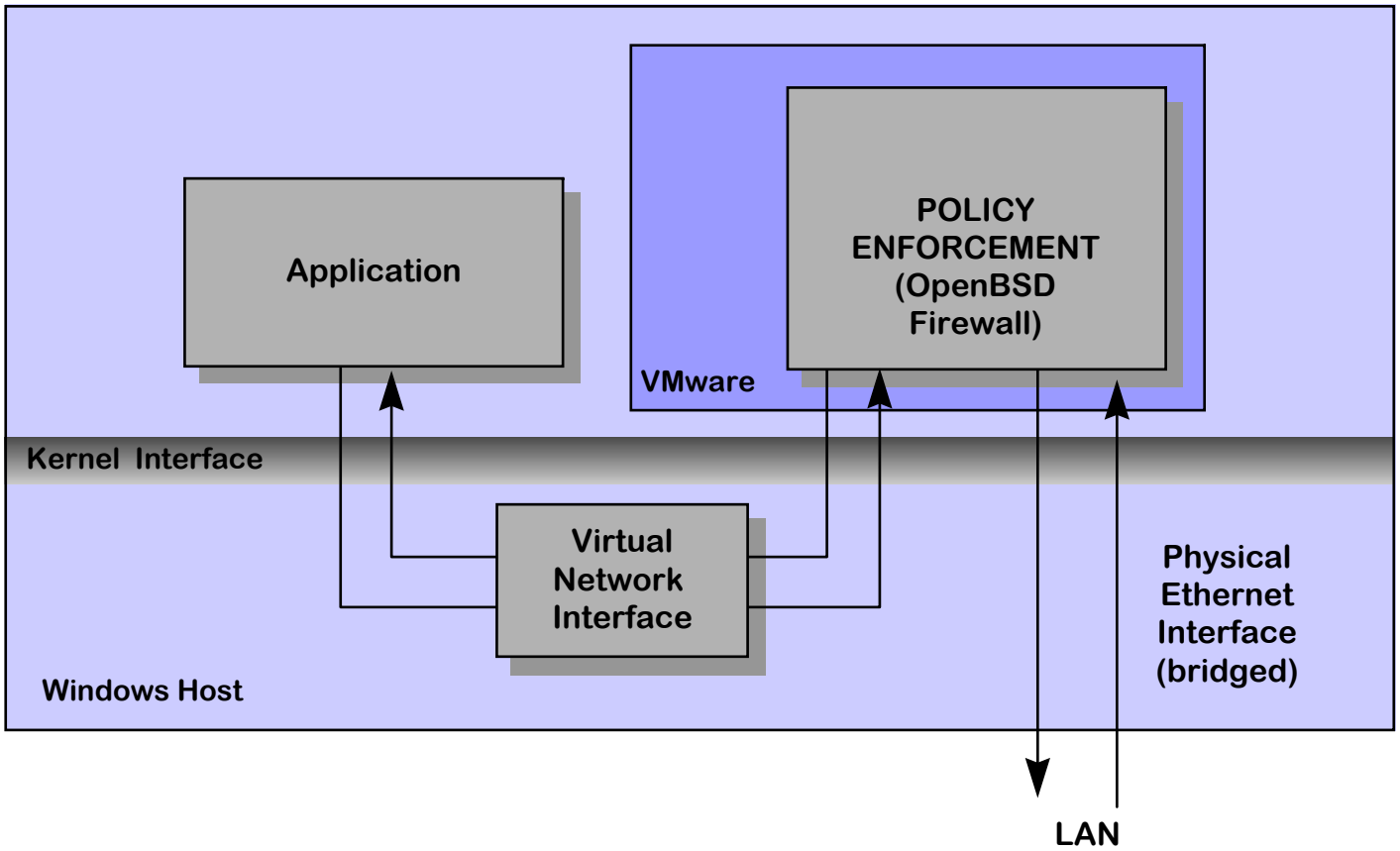
HASP (discussion)

- Usability (can the end-users use it)
  - open question -- depends on integration with applications
  - Firewall example
    - policies are delivered as physical cards and inserted into firewall
      - when the card is removed, the policy becomes invalid
    - policies are downloaded from policy web site

HASP Prototype 1



HASP Prototype 2



Conclusions

- HASP Benefits
  - policies are associated with tasks
  - hides policy details/implementation from user
  - allows transparency/external audits of policies
  - provides good tradeoff between need to specify policy and level of typical end-user
- Open Issues
  - linking policy implementation to natural language
  - locating “best” policy
  - usability

QUESTIONS?

Examples of Credentials

```
Keynote-Version: 2
Local-Constants:
      ISP_KEY = "rsa-base64:7231f..."
      ROUTE_KEY = "rsa-base64:33a41..."
Authorizer: ISP_KEY
Licensees: ROUTE_KEY
Conditions: app_domain == "Band-X" &&
      currency == "USD" &&
      &bandwidth <= "50Mbps" &&
      link_name == "Dublin-NYC" &&
      &amount >= 0.44
      && date < "20071120 -> "true";
Signature: "sig-rsa-sha1-base64:ab1XXA..."
```

```
Keynote-Version: 2
Local-Constants:
      ALICE_KEY = "rsa-base64:Mcg..."
      ISP_KEY = "rsa-base64:7231f..."
Authorizer: ALICE_KEY
Licensees: ISP_KEY
Conditions: app_domain == "BAND-X" &&
      currency == "USD" && amount == "0.44"
      && nonce == "eb2c3dfc8e9a" &&
      date == "20071120" -> "true";
Signature: "sig-rsa-sha1-base64:Qsd..."
```