Exploiting KeyNote in WebCom: Architecture Neutral Glue for Trust Management

Simon N. Foley, Thomas B. Quillinan,
John P. Morrison, David A. Power, James J. Kennedy.

Department of Computer Science,
University College,
Cork, Ireland.

Conclusions

WebCom provides the basis for complete separation of trust management from functionality: security critical code need not be intertwined with functionality within application.

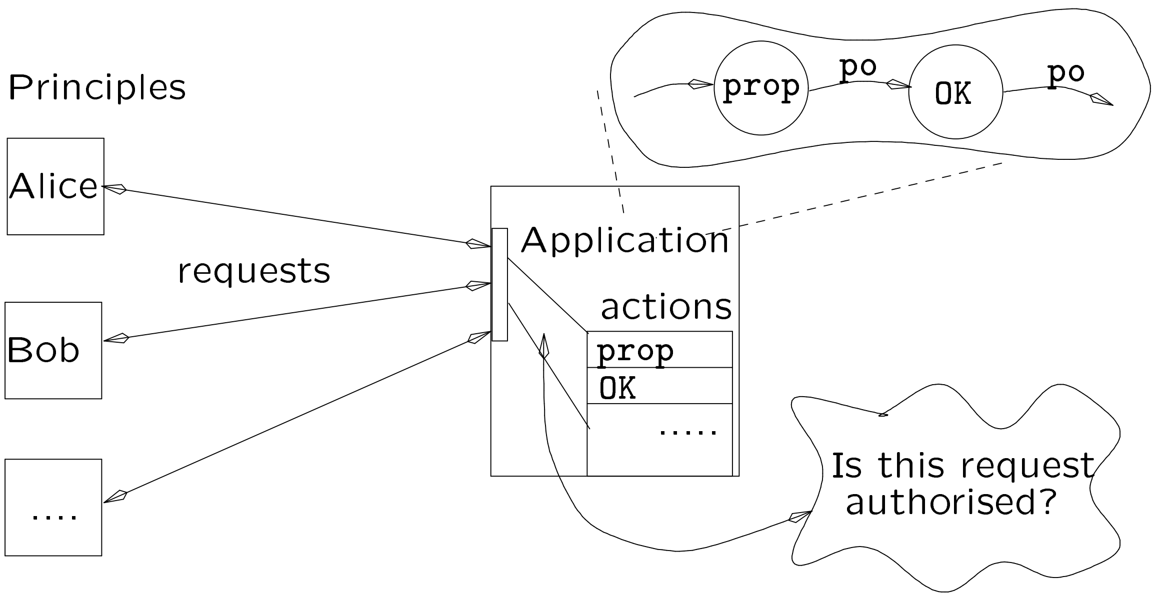
Application components may be regarded as untrusted.

Security-critical WebCom scheduler acts as a security wrapper for operations.

Trust Management system to help decide how to distribute, control and synchronise applications.

Authorisation

A simple purchase order processing application



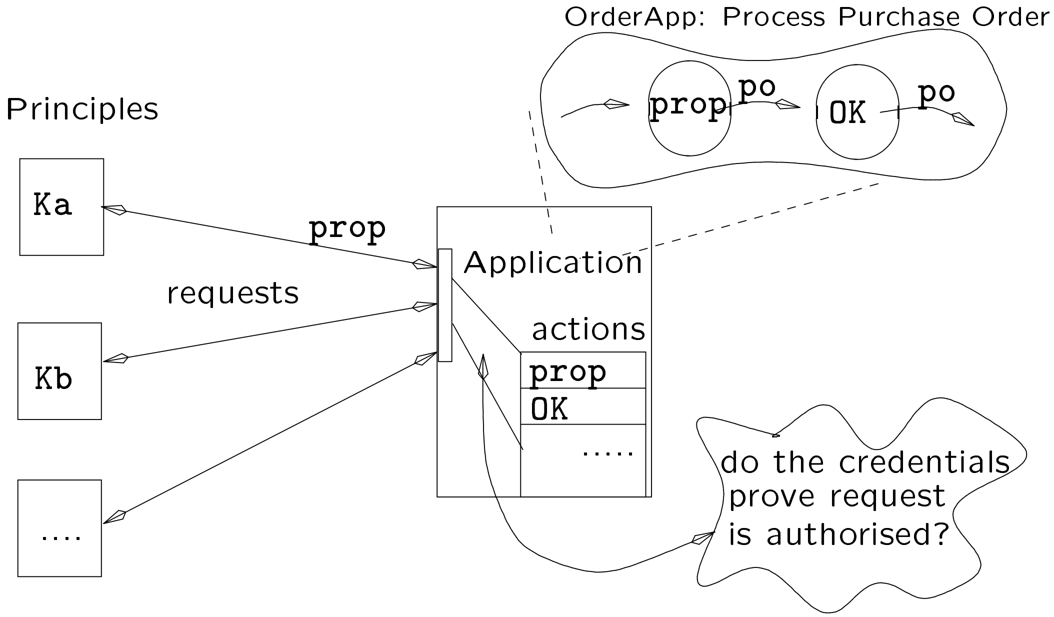
Having authenticated the principle, the application must determine whether requests by the principle for certain actions should be allowed.

Credentials and Authorisation

Use cryptographic credentials to delegate authority and trust.

```
Comment: Ka trusted to propose PO
Authorizer: "Kmgr"
licensees:  "Ka"
Conditions: app_domain=="OrderApp"
            && operation=="prop";
Signature:  .....
```

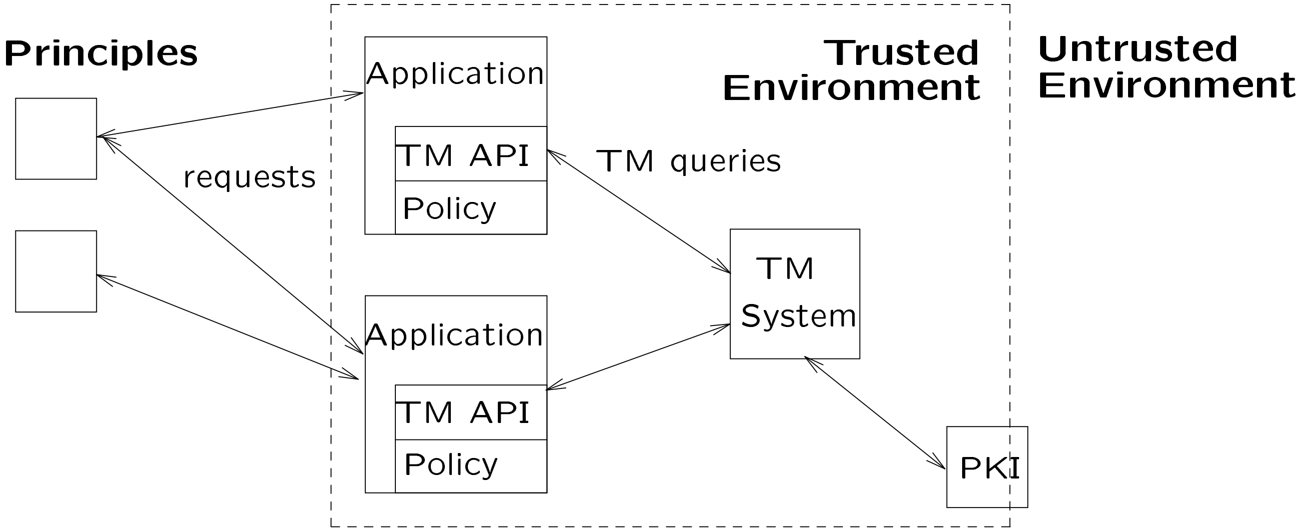
```
Comment: Kb is trusted to OK POs
Authorizer: "Kmgr"
licensees:  "Kb"
Conditions: app_domain=="OrderApp"
            && operation=="OK"
Signature:  .......
```



Given a policy (public keys, trusted in known ways) and a collection of credentials, the application must determine whether a particular key is authorised to request a particular action.

The KeyNote Trust Management Architecture

Trust Management provides a framework for reasoning about security policies, security credentials and trust relationships between cryptographic keys.



Example: Trusted Application Fragment

```
// given authenticated request to carry out an operation op:
authorizer= key making this request;
attribset= {app_domain="OrderApp", operation=op};
policy= policy credential for OrderApp above;
credentials= as provided by requester;

rslt= kn_query(..,authorizer,attribset,policy,credentials);
if (rslt=="true" && op=="prop")
  {process an order proposal..... }
else
if (rslt=="true" && op=="OK")
  {allow  validation of order..... }
else
  {reject request}
```

Observations

Although the KeyNote approach has many advantages, including, flexibility, unified mechanism, separation of mechanism and policy:

Separation of Concerns: Separation of functionality and security in *design*.

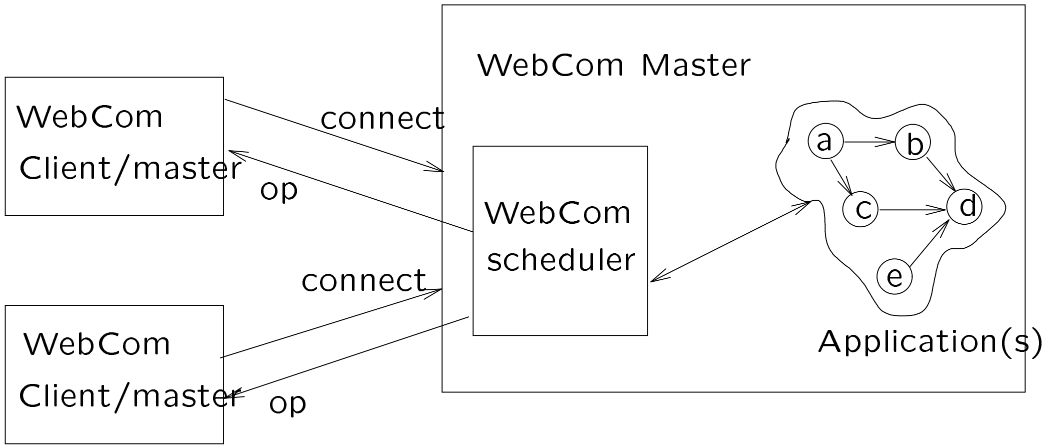
Coupling of security and functionality in *implementation*: security critical calls to Trust Management API are intertwined with functionality within an application.

Applications must be regarded as part of the trusted environment.

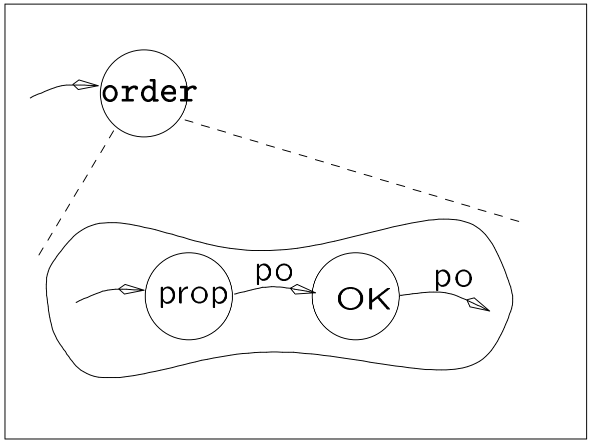
The WebCom Distributed System

WebCom is a system for distributing components for execution over a network.

WebCom applications are programmed as hierarchical *Condensed Graphs* of mobile components that represent tasks or further Condensed Graphs.



Example: WebCom Application



Condensed node `order` scheduled to run (fire) on a WebCom master.

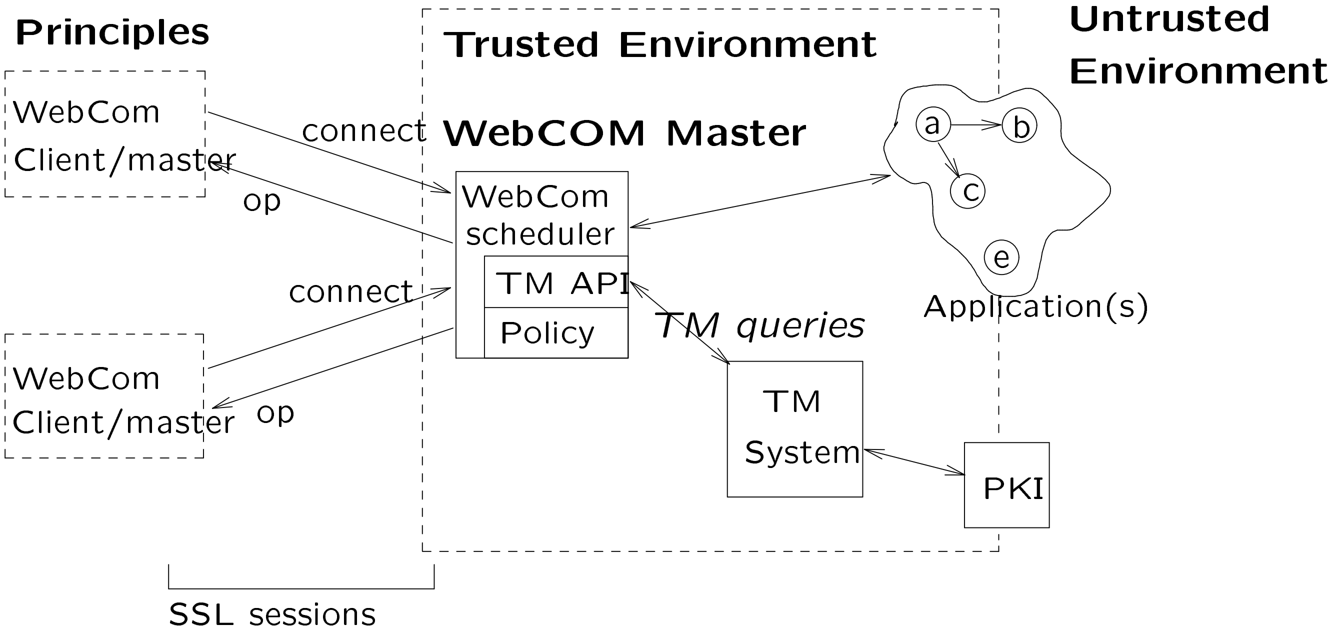
When `order` is fired, the nodes of its graph (`prop` and `OK`) can be scheduled to run on any WebCom client.

For example, `prop` runs on Alice's workstation, `OK` runs on Bob's workstation.

Component Glue: Condensed Graphs provide complete separation between implementation of components and synchronisation and communication.

A single distribution component (the WebCom Scheduler) is responsible for managing the scheduling and distribution of computation.

Integration of KeyNote in WebCom

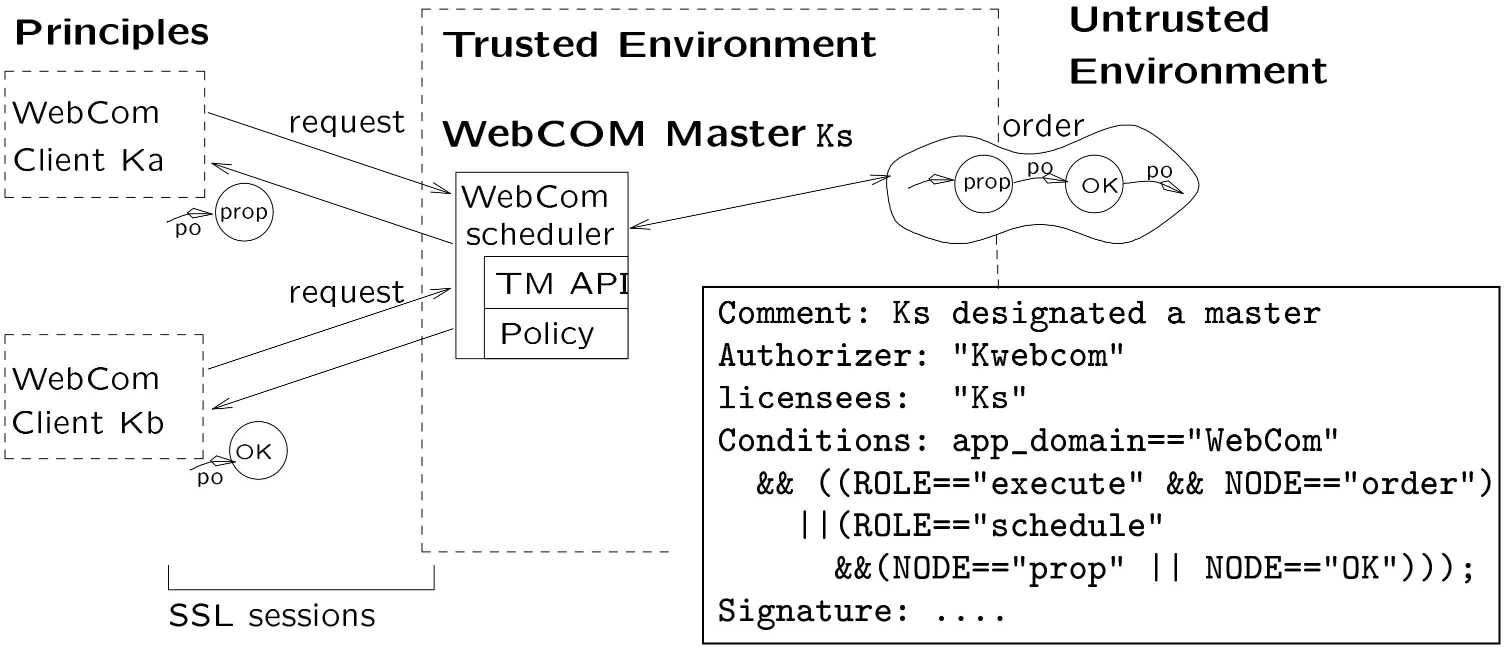


Security Glue: Condensed Graphs allow us to separate security critical calls to the TM API from application implementation.

The WebCom Scheduler becomes responsible for for securely managing the scheduling and distribution of computations.

WebCom Master Authorisation

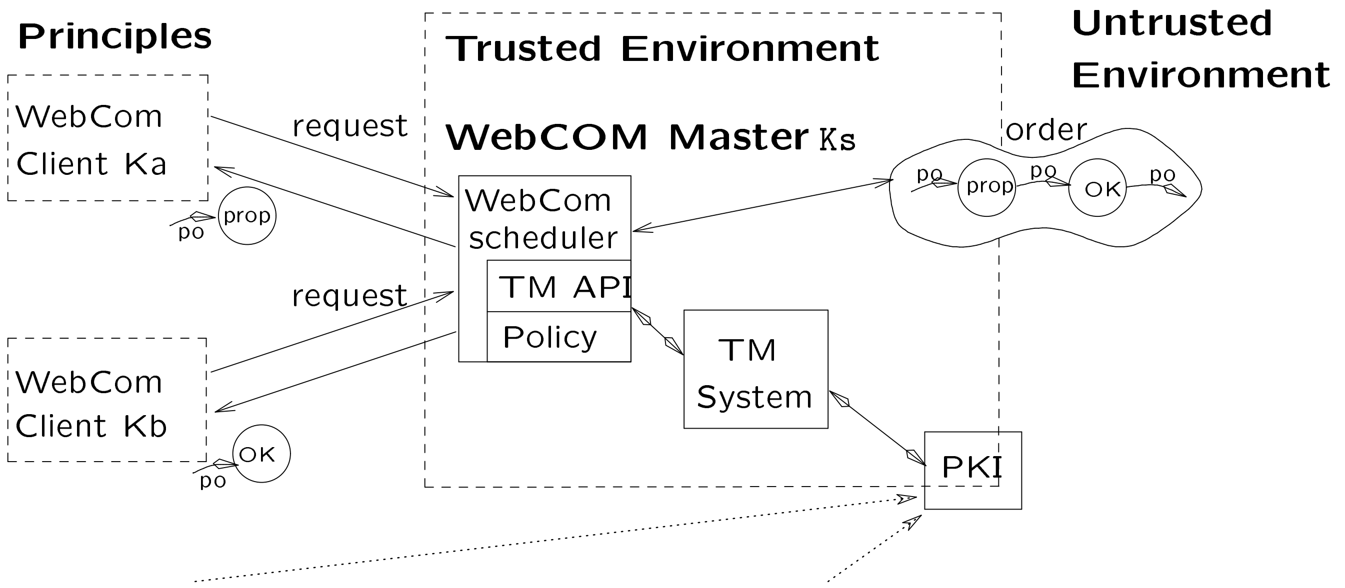
WebCom clients use KeyNote to determine whether the credentials provided by the authenticated master prove its authority to schedule operations.




```
Comment: Ks designated a master
Authorizer: "Kwebcom"
licensees:  "Ks"
Conditions: app_domain=="WebCom"
  && ((ROLE=="execute" && NODE=="order")
    ||(ROLE=="schedule"
      &&(NODE=="prop" || NODE=="OK")));
Signature: ....
```

Client Authorisation

WebCom Master must determine whether an authenticated client has the necessary authorisation to execute certain operations.




```
Comment: Ka authorized to execute prop
Authorizer: "Kwebcom"
licensees:  "Ka"
Conditions: app_domain=="WebCom"
  && ROLE=="execute" && NODE=="prop";
Signature: ....
```

```
Comment: Kb authorized to execute OK
Authorizer: "Kwebcom"
licensees:  "Kb"
Conditions: app_domain=="WebCom"
  && ROLE=="execute" && NODE=="OK";
Signature: ....
```

Conclusions

WebCom provides the basis for complete separation of trust management from functionality: security critical code need not be intertwined with functionality within application.

Application components may be regarded as untrusted.

Security-critical WebCom scheduler acts as a security wrapper for operations.

Trust Management system to help decide how to distribute, control and synchronise applications.