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.

Abstract

A practical trust management system for the WebCom distributed computing architecture is described. KeyNote credentials are used to determine the authorisation of X509 authenticated SSL connections between peers. WebCom provides a meta-language (glue) for bringing together the components of a distributed application in such a way that the components need not concern themselves with synchronisation or concurrency issues. The integration of KeyNote in WebCom enhances the trust management framework by providing for a complete separation of security from the components of a distributed application into a single security critical component. This encourages a loose coupling between application components and trust management, which, in addition to supporting off the shelf application components, results in applications that are easier to develop, understand, maintain and secure.

KEYWORDS: Trust Management; Authorisation; Mobile Computation; Meta-Computing; Distributed System Security;
Software Engineering; Condensed Graphs.

1 Introduction

Trust Management (TM) [4, 6, 19] is an approach to constructing and interpreting the trust relationships among public keys that are used to mediate security critical actions. Credentials are used to specify delegation of authority among public keys, and are used to determine if a signed request complies with a local authorisation policy. KeyNote [3, 4] is an expressive and flexible trust management scheme that provides a simple credential notation for expressing both security policies and delegation. KeyNote has been used to provide trust management for a number of applications including active networks [7] and to control access to Web pages [1].

A standard API to KeyNote is used by an application to make queries about whether requests (to the application) have authorisation or not. The formulation and management of security policies and credentials are separate from the application, making it straight forward to support trust management policies across different applications. However, a coupling remains between the trust management system and the application since KeyNote API calls are embedded within the application code. This paper addresses this concern by proposing a framework that avoids this intertwining of functionality and security-critical code.

In this paper we describe the development of a KeyNote based trust management system for the WebCom [17, 14] distributed computing architecture. Applications in WebCom are programmed as hierarchical *Condensed Graphs* of mobile components. These components represent tasks, or graphs of sub-components, that are made available for execution across a network. Condensed Graphs provide for a complete separation between the implementation of its components and their synchronisation and communication. It can be used as a meta-language for 'glueing' together new and existing off the shelf components.

This flexibility extends to trust management with the integration of KeyNote into WebCom. The advantage of using WebCom is that WebCom application components need not use the KeyNote API and, therefore, need not be aware, in a programmatical sense, of the trust management system. WebCom acts as a form of reference monitor, using the contextual information in the Condensed Graph of an application to construct KeyNote queries to determine the authorisation for the application components. This loose coupling of functionality and trust management leads to applications that are easier to develop, understand, maintain and secure.

The paper is organised as follows. Sections 2 and 3 outline the KeyNote and WebCom architectures, respectively. Sections 4, 5 and 6 describe how KeyNote is integrated into WebCom by using KeyNote credentials to determine the authorisation of X509 authenticated SSL connections. Section 7 sketches the implementation issues. Section 8 provides general observations and discussion.

2 Trust Management

When a request from an untrusted principle (key) is made to a networked application to execute a particular action, then, authentication notwithstanding, the application must determine whether the key(s) that made the request is authorised. Authorisation comes in the form of digitally signed public key credentials that bind public keys to the authorisation to perform various actions. For example, Alice may hold a credential, signed by her manager's private key, binding her public key to her authorisation to place orders up to a value of £100.00. Alice's public key signs an order request to the purchasing application; her credential provides proof of authorisation.

In practice, authorisation is achieved by a collection of credentials that exhibit the necessary trust relationships between their keys. For example, we may trust Alice's public key for orders up to £100.00, if her manager's public key is trusted to delegate orders up to £100.00 (or more), and so forth along a delegation chain that ends in a key that is known to be appropriately trusted. Given a policy (public keys, trusted in known ways), and a collection of credentials, a network application must determine whether a particular public key is authorised to request a particular operation.

EXAMPLE 1 A simple purchase order processing application runs on a server and accepts requests from clients. The request for operation prop is made to propose a new order while the request OK is used to inspect and validate the order. We expect that, in practice, a clerk will have the authority to propose orders and a supervisor will have the authority to validate orders; this authority will be delegated by their manager.

Assume that the owner of public key `Kmgr` is trusted to make requests to the order processing application. This is specified by the following KeyNote credential.

```
KeyNote-Version: 2
Comment: Unconditionally delegate trust for the
         ordering application to the key Kmgr
Authorizer: "POLICY"
licensees:  "Kmgr"
Conditions: app_domain=="OrderApp" &&
            (operation=="prop" || operation=="OK");
```

This is a special *policy* credential that defines the conditions under which requests from the licensee key `Kmgr` may be trusted by the application `OrderApp`. These conditions are defined using a C like expression syntax in terms of the *action attributes*, in this example, `app_domain` and `operation` which are used characterise the circumstances of a request. Note that for the purposes of illustration in this paper we use short simple names to represent public keys rather than encoding them as proper cryptographic keys.

The owner of public key `Kmgr` has the authority to delegate this trust to other keys and does so by signing the following credential for a clerk who owns public key `Kalice`.

```
Comment: Kalice is trusted to propose purchase orders
Authorizer: "Kmgr"
licensees:  "Kalice"
Conditions: app_domain=="OrderApp" && operation=="prop";
Signature:  ..<signed by Kmgr>..
```

In signing this credential, `Kmgr` delegates authority for proposing orders to the key `Kalice`. When Alice proposes an order (signed by `Kalice`), she presents this credential as proof of authorisation. We can confirm that this key is indeed authorised since, by default (policy), we trust `Kmgr` to propose and OK orders and `Kmgr` has delegated some of this trust to `Kalice`, by virtue of signing the credential. A similar credential may be issued to the supervisor Bob:

```
Comment: Kbob is trusted to validate purchase orders
Authorizer: "Kmgr"
licensees:  "Kbob"
Conditions: app_domain=="OrderApp" && operation=="OK"
Signature:  ..<signed by Kmgr>..
```

$\triangle$

An application may use a Trust Management (TM) scheme such as KeyNote [4] to determine whether requests to it are authorised, without the application having to know about how that determination is made. The KeyNote trust management architecture is depicted in Figure 1. A Public Key Infrastructure (PKI) is used to manage the credentials.

EXAMPLE 2 Continuing Example 1, when the `OrderApp` application queries the KeyNote TM system to determine whether it is safe to execute a particular request, it must specify the circumstances of the query. These circumstances include: *action authorisers*, corresponding to the key(s) that made the request; *action attribute set*, which is a set of action attribute name and value pairs that characterise the request; *policy* credentials, representing the keys that are trusted, and other *credentials* as provided by the requester and/or PKI.

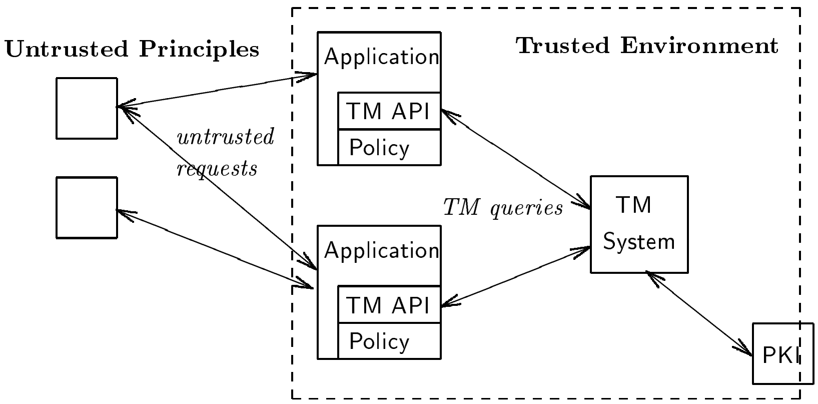


Figure 1: KeyNote Trust Management Architecture (From [3]).

For example, when `Kalice` requests an order proposal then the order application queries KeyNote with action authoriser `Kalice`, action attribute set {`app_domain` ← `"OrderApp"`,`operator` ← `"prop"`}, the policy credential for `Kmgr` above, and a set of signed credentials provided by Alice. KeyNote must determine if the given request is authorised based for the circumstances provided. The application interacts with KeyNote via calls to the KeyNote API. For example, the `OrderApp` might include the following code:

```
// 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="prop")
  {allow  validation of order}
else
  {reject request}
```

△

The KeyNote architecture provides a level of separation between the provision of security policy authorisation and application functionality. As a software engineering paradigm, techniques that support separation of concerns for security [4, 8], synchronisation [12], real-time constraints [2], and so forth are desirable since they lead to applications that are easier to develop, understand and maintain.

KeyNote provides this separation of concerns at a conceptual level. However, as illustrated in Example 2, calls to the KeyNote API must still be coded as part of the application system. While ensuring cohesive applications, there remains a coupling within the application between the functional and security concerns, that is, we do not have complete separation of concerns at the code level, since security-critical calls to the trust management API are intertwined with the functionality. This coupling means that applications are regarded as part of the trusted environment, as illustrated in Figure 1.

3 The WebCom Distributed System

The WebCom system [17] uses a variant of the client/server paradigm to distribute operations for execution over a network. The system uses implicit program parallelism, separates the application program from the underlying computation engine and is efficient yet compact. The heart of the WebCom system is the Condensed Graphs computational model that it employs [15, 18]. Applications are coded as hierarchical graphs which provide a simple notation in which lazy, eager and imperative computation can be naturally expressed. There are two types of distributable operation: nodes that represent atomic tasks and condensed nodes that represent subtasks encapsulated as subgraphs.

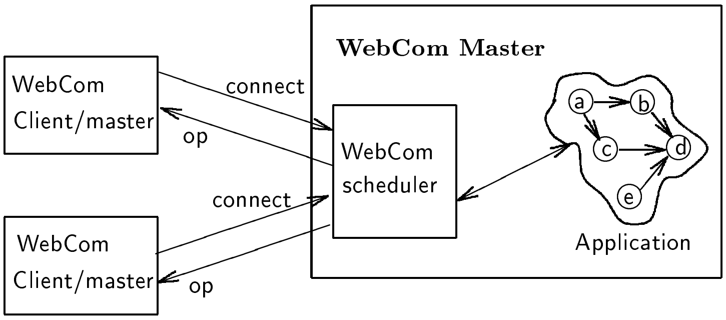


Figure 2: WebCom Architecture

The top-level architecture of WebCom, illustrated in Figure 2, consists of a master and an arbitrary number of clients. Clients connect to the master, which in turn assigns them operations for execution. Atomic operations may be scheduled to execute on any client. Condensed nodes (defining graphs) may be scheduled to run on designated client-masters, which have the ability to schedule its component operations to further clients.

An advantage of developing distributed applications as Condensed Graphs is that their implementation (graph) can be coded independently of the underlying system and/or network architecture. Atomic operations are value-transforming actions and can be defined at any level of granularity, ranging from low-level machine instructions to mobile-code programs such as applets, COM objects or COTS components. Atomic operations need not address synchronisation or concurrency concerns: such details are implicitly specified in the Condensed Graph and managed by WebCom. Therefore, WebCom provides for a separation of concerns at the code level, allowing a loose coupling between functionality and control.

EXAMPLE 3 Condensed Graphs are used to exploit parallelism in a computation and WebCom can schedule the computation across networks of workstations [17]. Figure 3 gives an example of a graph that schedules a distributed brute-force key search given known plain/cipher text. The key space is split into a series of intervals indexed as $0, 1, \ldots, maxindex$. Atomic operation

Int cr(Int interval);

searches a specified interval for the key. If found the key is returned, otherwise 0 is returned. Condensed node search is defined recursively and has a high degree of parallelism that can be exploited by WebCom which schedules its component cr operation to be executed on clients (and can schedule graph operation search to client-masters). Operation search is passed the initial value $maxindex$. By definition [15], the behaviour of a condensed node such as search is constructed as a Condensed Graph with a single entry node (E) and single exit node (X). Node ifel is an if-then-else operation, with boolean condition b and then and else clauses t and b, respectively. △

EXAMPLE 4 Condensed Graphs may also be used as a distributed job control language to describe application system scheduling requirements, such as work-flow. A clerk performs a prop operation to complete a purchase order and a manager executes an OK operation to inspect and authorise the order. The scheduling of this system is described by the condensed node order defined in Figure 4. A WebCom master schedules atomic actions prop and OK to appropriate connected clients. For example, [13] shows how

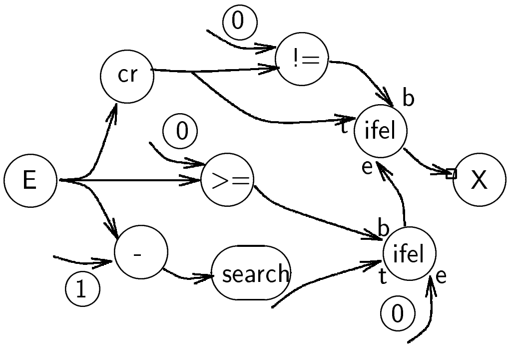
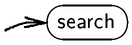


Figure 3: Recursive Definition of Condensed Node 

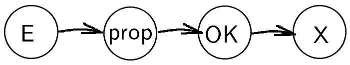
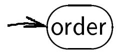


Figure 4: Definition of Condensed Node 

such operations can be implemented as Microsoft COM objects that are scheduled by WebCom according to the graph order. In this system, connecting clients specify the operations that they wish, or are willing, to have scheduled to them. In the next section we will describe how such clients may be authenticated and their authorisation determined. $\triangle$

4 Integrating KeyNote into WebCom

Figure 5 illustrates how the KeyNote trust management scheme (Figure 1) is integrated into WebCom by regarding WebCom as an application. The WebCom master authenticates its clients and uses their credentials to determine what operations it may schedule to them. Each WebCom client has a trust management architecture that is similar to the KeyNote architecture in Figure 1, authenticating the master and using the master's credentials to determine whether it is authorised to schedule the operation. We selected KeyNote because of its simplicity and expressiveness. KeyNote provides a simple notation for specifying both local security policies and credentials that can be sent over untrusted networks.

Example 5 For this presentation we assume that there is just one root public key Kwebcom as specified by the following KeyNote trusted policy credential.

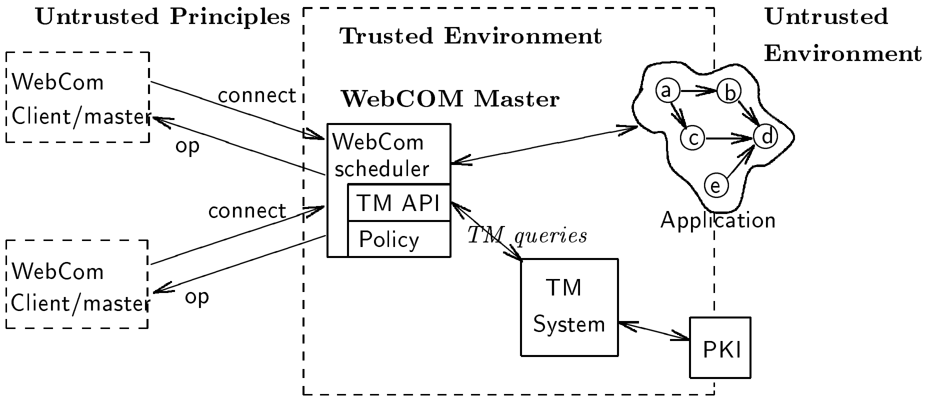


Figure 5: WebCOM-KeyNote Architecture

```
KeyNote-Version: 2
Comment: Unconditionally delegate trust for the
         WebCom application to the key Kwebcom
Authorizer: "POLICY"
licensees: "Kwebcom"
Conditions: app_domain=="WebCom";
```

This credential uses just one action attribute—`app_domain`—to specify that `Kwebcom` is unconditionally trusted to delegate authority related to the WebCom application. It is proposed in [4] that the Internet Assigned Numbers Authority (or some other suitable authority) will provide a registry of reserved `app_domain` names that provide the name of the application domain for which the credential is defined. △

One key difference between the WebCom-KeyNote architecture in Figure 5 and the original KeyNote architecture of Figure 1 is that in Figure 5, applications may, if desired, be regarded as untrusted since they do not contain any security critical code. The WebCom scheduler is the security-critical component and acts as a form of reference monitor, interfacing with the KeyNote system to decide how to distribute, control and synchronise applications.

5 WebCom Master Authorisation

WebCom masters schedule operations to be executed by WebCom clients. Some of these operations may access local resources on the client and therefore the WebCom master must prove that it has the authority to do so by furnishing the appropriate credentials to the client. Figure 6 illustrates how,

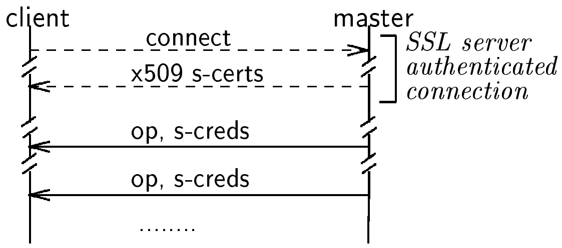


Figure 6: Master Authentication Message Sequence

in the current prototype, X509 certificates are used for authentication and KeyNote credentials are used for authorisation:

1. The connecting WebCom client establishes an SSL connection with the master, and in the process, the master presents an appropriate X509 certificate chain to its public key K_{master}. If the protocol run is successful then the client knows that it is communicating with K_{master}.

2. The master schedules operation `op` to the client over the SSL connection and sends the necessary credentials that prove that K_{master} is authorised to schedule `op`. The WebCom client uses KeyNote to determine if the schedule is authorised, given the credentials of the master.

When a WebCom client uses KeyNote to determine whether it is safe to execute a scheduled operation or action, it must specify the circumstances of the query as an *action attribute set*. This is a list of attribute names and value pairs that are agreed between the application writers and the writers of the related credentials. In addition to the existing attribute `app_domain`, two action attributes are proposed to identify the circumstances of a WebCom KeyNote query:

- Attribute name `ROLE`, which can have, for example, values `schedule` or `execute`, indicating whether the principle is authorised to schedule or execute.

- Attribute name `NODE`, which can have values identified by the operations that can be scheduled/executed.

EXAMPLE 6 Consider again Example 3. Assume a WebCom master owns public key Kmike and has been given the following KeyNote credential, signed by (the private key of) Kwebcom:

```
KeyNote-Version: 2
Comment: Kwebcom delegates scheduling authority to Kmike
Authorizer: "Kwebcom"
licensees:  "Kmike"
Conditions: app_domain=="WebCom" && ROLE=="schedule"
            && (NODE=="cr" || NODE=="search");
Signature: ..<signed by private key of "Kwebcom">..
```

This credential authorises `Kmike` to carry out an operation if the C-like expression in the `condition` evaluates to true, given the attribute values as defined in the action attribute set of the operation requested.

We assume that a suitable X509 certificate for Kmike is also available. When operation `cr` is scheduled to a client, by an authenticated `Kmike`, the client queries KeyNote, with action attribute set

$$\begin{array}{rcl} \texttt{app\_domain} & \leftarrow & \texttt{"WebCom"} \\ \texttt{ROLE} & \leftarrow & \texttt{"schedule"} \\ \texttt{NODE} & \leftarrow & \texttt{"cr"} \end{array}$$

and action authoriser `Kmike`. Given the above credential and the policy credential from Example 5, KeyNote returns `true` indicating that it is safe for the WebCom client to execute the operation.

KeyNote can support delegation chains and sophisticated trust relationships. For example, `Kmike` can delegate the authority to schedule to `Kmaude` only `cr` operations, but not to to delegate this authority any further.

```
...
Authorizer: "Kmike"
licensees:  "Kmaude"
Conditions: NODE=="cr" && _ACTION_AUTHORISER=="Kmaude";
...
```

The KeyNote reserved attribute `_ACTION_AUTHORIZERS` a pre-defined attribute of the action attribute set, defining the names of principles directly authorising an operation [4]. The credential above uses this attribute to prevent further delegation.

Any client will execute `cr` operations scheduled by `Kmaude`, but will not accept `search` operations. Given a collection of credentials, the keynote query searches the credential delegation graph for the appropriate paths that connect the trusted keys (policy) to the action authoriser (requesting key), and whose conditions are satisfied according to the action attribute set. The reader is referred to [4] for further details on how to specify KeyNote credentials. △

Note that a standard code-signing approach such as that used by Java [10], would not provide sufficient security infrastructure support for WebCom. A WebCom master schedules more than just code: it schedules mobile computations that are comprised of tasks and their inputs. A client must determine whether it should trust the master for the task it schedules along with the inputs to the task that the master provides and the subsequent output that is returned to the master.

6 WebCom Client Authorisation

WebCom client authorisation is done by the master to determine whether the client has the necessary authorisation to execute certain operations whose inputs or outputs may be security critical. The trust architecture for the WebCom master is given in Figure 5. Figure 7 illustrates how, in the current prototype, X509 certificates are used for client authentication and KeyNote credentials are used for authorisation. Master certificates and credentials

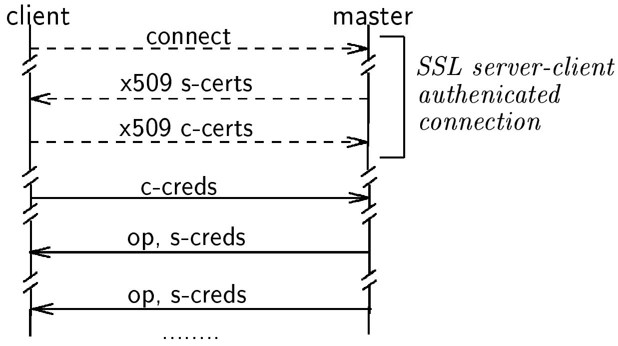


Figure 7: Master and Client Authentication Message Sequence

are used as in the previous section. However, clients must now present their credentials when the secure connection is first established if they wish to have operations scheduled to them.

A WebCom master maintains a list of clients that are currently connected and available to be scheduled operations. Before scheduling an operation to a client, the master must determine if the client is authorised to execute the operation. This can done by the master making a query to KeyNote with an action attribute set based on attributes `ROLE` and `NODE`, and by the action authoriser which is the client key. If the client is not authorised to execute any of the operations scheduled by the master then the SSL connection can be terminated.

Connecting WebCom clients declare themselves available to execute a range of possible scheduled operations. The WebCom master uses KeyNote to *search* the database of the credentials of all connected and available clients for an action authoriser (client key) that has the authorisation to execute the particular operation (as specified by the action attribute set). This search operation is an extension to the standard KeyNote API.

Example 7 Continuing Example 3, suppose that there are two WebCom clients, identified by public keys `Kcharles` and `Kclare`. `Kcharles` is authorised to execute `cr` operations.

```
...
Authorizer: "Kwebcom"
licensees:  "Kcharles"
Conditions: app_domain=="WebCom"
            && ROLE=="execute" && NODE=="cr";
...
```

`Kclare` is authorised to execute or schedule `cr` and `search` operations, that is, she has the authority to operate as a client-master.

```
...
Authorizer: "Kwebcom"
licensees:  "Kclare"
Conditions: app_domain=="WebCom"
            && _ACTION_AUTHORISER=="Kclare"
            && (ROLE=="schedule" || ROLE=="execute")
            && (NODE=="cr" || NODE=="search");
...
```

We assume that appropriate X509 certificates for these keys are available for authentication purposes.

When the clients connect (and are authenticated), the WebCom master adds their credentials to its client credential database. When the master is ready to schedule operation `cr`, it sends a search query to KeyNote, given action attribute set `(ROLE←"execute"; NODE←"op")`. This new KeyNote search function searches the client credential database for action authorisers (keys) that are authorised, and in this case could return either `Kcharles` or

Kclare. If the operation search is to be scheduled, then only Kclare could be returned by the KeyNote search. △

EXAMPLE 8 A WebCom master (Kserver) running on a trusted server is authorised to execute an order operation and schedule its components (Example 4). We assume that the order operation may have been scheduled to Kserver from some other WebCom master.

```
...
Authorizer: "Kwebcom"
licensees:  "Kserver"
Conditions: app_domain=="WebCom"
            && _ACTION_AUTHORISER=="Kserver"
            && ((ROLE=="execute" && NODE=="order")
              ||(ROLE=="schedule"
                  && (NODE=="prop" || NODE=="OK")));
...
```

WebCom clients Alice (Kalice) and (Kbob) are authorised to execute prop and OK operations, respectively.

```
...
Authorizer: "Kwebcom"
licensees:  "Kalice"
Conditions: app_domain=="WebCom"
            && ROLE=="execute" && NODE=="prop";
...
```

```
...
Authorizer: "Kwebcom"
licensees:  "Kbob"
Conditions: app_domain=="WebCom"
            && ROLE=="execute" && NODE=="OK";
...
```

When Kalice connects, requesting prop operations, she presents her credential(s). Her authorisation is validated and the master schedules the execution of prop on her client system. Note that in this example, Kalice neither offers, nor has the authority, to execute OK. Similarly, Bob connects, offering to execute OK, which is scheduled. The WebCom master uses the same SSL secured connection to send operations to execute, and receive results from, the WebCom clients.

This is a example of a simple static separation of duty policy, provided by a combination of credentials and Condensed Graphs. We are currently exploring how dynamic separation of duty using KeyNote support for credentials specifying threshold signatures might be transparently supported within our framework. △

7 Implementation

A number of working prototypes that use Condensed Graphs have been developed, demonstrating it usefulness as a general model of computation. Prototypes include a sequential Condensed Graph interpreter [15]; a PVM application programming interface for a cluster [16]; WebCom [17] referred to in this paper, and a specialisation of WebCom that supports Microsoft COM objects [13].

To date, we have integrated the trust management support scheme described by this paper into the existing Java-based WebCom system described in [17]. The integration was straightforward and the current prototype uses existing packages where possible:

- The JCSI [11] package provides cryptographic, X509 certificate and SSL support. KeyNote credentials are constructed using the keys extracted from their corresponding X509 certificates.
- The existing KeyNote API was extended by the addition of a search query function, as outlined in Section 6.
- The current reference implementation for KeyNote is implemented in C. Java Native Interface is currently used to provide a Java API to KeyNote. It is planned to eventually replace this with a Java implementation of the Keynote Interpreter.

8 Discussion

KeyNote credentials are used to determine the authorisation of X509 authenticated SSL connections between WebCom masters and clients. Client credentials are used by WebCom masters to determine what operations the client is authorised to execute; WebCom master credentials are used by clients to determine if the master had the authorisation to schedule the (trusted) mobile-computation that the client is about execute; this authorisation covers not just the scheduled code, but also its input data and output.

The relationship between the WebCom trust management system and local operation system protection mechanisms is not addressed in this paper. For example, in what protection domain should a client execute an operation that has been scheduled by an authorised master? In [9] we propose an abstract protection model for Condensed Graphs; our integration of KeyNote into WebCom can be regarded as one step towards implementing this protection model.

We use KeyNote to provide trust management for WebCom because of its ability to support expressive and fine-grained authorisation and delegation policies. KeyNote Credentials are typically short, yet expressive, and are, therefore, not expected to contribute significantly to the communication overhead between clients and masters. For the purposes of illustration, only simple examples are presented in this paper. The reader is referred to [5] for a discussion of the expressiveness of KeyNote.

The emphasis in this paper is one of using WebCom to provide complete separation between a distributed application and trust management. We have not yet considered PKI issues such as how best to control the dissemination of KeyNote credentials, revocation, and so forth. However, since our architecture uses X509 certificates for authentication then we expect that their associated PKI's should a degree of support for issues such as key revocation. These are topics for future research

The implementation described in this paper expresses security at the granularity of operation identifiers such as `prop`, `order`, and so forth. However, we expect that it will be straight forward to generalise this to include reference to the attributes used to pass input and output to and from these operations. During execution, the nodes in a Condensed Graph are represented as a triple: input parameter bindings, output binding and the operation applied [15]. These attributes can be referred to within credentials and form part of the action attribute set during a KeyNote query or search. With such an extension, Example 4 could be generalised to support credentials that authorise Bob to validate orders up to some limit.

WebCom masters may promote clients to become client-masters by passing condensed nodes (for example `search` from Example 3), the components of which are subsequently scheduled by clients of the promoted client [14]. We are currently extending the WebCom trust management architecture to exploit promotion. This uses credentials to control whether masters have the authority to promote clients, and whether clients, in turn, have the authority to be promoted. Writing credentials that specify this is straightforward, and can be done by enlarging the domain of attribute `ROLE` to include values `promoter` and `promotee`. In addition to promoting a client, the master must write the necessary credentials to delegate to the promoted client the authority to schedule the operations it will be managing.

We are also currently porting our implementation to the existing WebCom application that supports Microsoft COM objects [13]. This would be particularly useful since it will provide the basis of a trust management glue for existing COTS components, such as spreadsheets, databases, word-processors, and so forth.

9 Conclusion

The integration of KeyNote in WebCom enhances the trust management framework by providing for a complete separation of security from the components of a distributed application into a single security-critical component. While KeyNote provides for a loose coupling between trust management and an application at the conceptual level, the WebCom-KeyNote architecture takes this further by providing complete separation at the code level.

Application developers need not intertwine their application code with security-critical calls to the trust management system. Functional components such as `prop` and `OK` (Example 4) can be coded independently of security concerns; security concerns are expressed independently in terms of a credential based policy, and the Condensed Graph specifies the synchronisation and control concerns between these components. Only operation identifiers couple these concerns. This means that applications may be regarded as untrusted: the security-critical WebCom scheduler acting as a security wrapper for operations and using the trust management system to help decide how to distribute, control and synchronise applications. We think of this as an architecture neutral glue for trust management.

Acknowledgments

This research was supported by the Enterprise Ireland National Software Directorate. The authors would like to thank Úlfar Erlingsson and the anonymous reviewers for their helpful suggestions on the paper.

References

[1] Apache-ssl release version 1.3.6/1.36. Open source software distribution. `http://www.apache.org`.

[2] M Aksit et al. Real time specification inheritance anomalies and real time filters. In *European Conference on Object-Oriented Programming (ECOOP)*, pages 386–407. Springer Verlag LNCS 821, 1994.

[3] M. Blaze. Using the KeyNote trust management system. `http://www.crypto.com/trustmgt`, December 1999.

[4] M Blaze et al. The keynote trust-management system version 2. September 1999. Internet Request For Comments 2704.

[5] M Blaze et al. The role of trust management in distributed systems security. In *Secure Internet Programming: Issues in Distributed and Mobile Object Systems*. Springer-Verlag Lecture Notes in Computer Science, 1999.

[6] M Blaze, J Feigenbaum, and J Lacy. Decentralized trust management. In *Proceedings of the Symposium on Security and Privacy*. IEEE Computer Society Press, 1996.

[7] M. Blaze, J. Ioannidis, and A.D. Keromytis. Trust management and network layer security protocols. In *Security Protocols International Workshop*. Springer Verlag LNCS, 1999.

[8] S.N Foley. A kernelized architecture for multilevel secure application policies. In *European Symposium on Research in Security and Privacy*. Springer Verlag LNCS 1485, 1998.

[9] S.N. Foley and J.P. Morrison. Scheduling protection using condensed graphs. Technical Report TR-00-02, Department of Computer Science, University College, Cork, June 2000.

[10] L. Gong et al. Going beyond the sandbox: An overview of the new security architecture in the java development kit 1.2. In *USENIX Symposium on Internet Technologogys and Systems*, pages 103–112, 1997.

[11] DSTC Security group. JCSI Java Crypto and Security Implementation. See `http://security.dstc.edu.au`.

[12] C.V. Lopes and K.J. Lieberherr. Abstracting process-to-process relations in concurrent object-oriented applications. In *European Conference on Object-Oriented Programming (ECOOP)*. Springer Verlag LNCS 821, 1994.

[13] John P. Morrison et al. Architectural neutral glue for COM objects. In preparation, 2000.

[14] John P. Morrison and David A. Power. Master promotion and client redirection in the webcom system. In *PDPTA, Las Vegas USA*, 2000.

[15] J.P. Morrison. *Condensed Graphs: Unifying Availability-Driven, Coercion-Driven and Control-Driven Computing.* PhD thesis, Eindhoven, 1996.

[16] J.P. Morrison and R. Connolly. Facilitating Parallel Programming in PVM using Condensed Graphs. Proceedings of EuroPVM'99: Universitat Autonoma de Barcelona, Spain. 26-29 Sept 1999.

[17] J.P. Morrison, D.A. Power, and J.J. Kennedy. A Condensed Graphs Engine to Drive Metacomputing. Proceedings of the international conference on parallel and distributed processing techniques and applications (PDPTA '99), Las Vagas, Nevada, June 28 - July1, 1999.

[18] J.P. Morrison and M. Rem. Speculative computing in the condensed graphs machine. proceedings of IWPC'99: University of Aizu, Japan, 21-24 Sept 1999.

[19] R Rivest and B Lampson. SDSI - a simple distributed security infrastructure. In *DIMACS Workshop on Trust Management in Networks*, 1996.