Supporting Imprecise Delegation in KeyNote using Similarity Measures

Simon N. Foley
Department of Computer Science,
University College, Cork, Ireland.
s.foley@cs.ucc.ie

Abstract

With decentralized authorization comes the challenge of ensuring that when a credential is written, then it precisely captures the delegation that is intended. A request for a particular service may be unexpectedly rejected, despite the requester having credentials for what should be considered a comparable service. This paper considers how techniques that support imprecision in case-based reasoning systems can be used when delegating and checking authorizations in the KeyNote trust management system.

1 Introduction

Cryptographic delegation certificates/credentials specify delegation of authorization between public keys. When an application receives a request to execute a particular action, then authentication notwithstanding, the application uses credentials provided by the requester to determine whether or not the request is authorized. Trust Management approaches such as KeyNote [3] and SPKI [4] provide assistance to applications in making these decisions. Trust Management facilitates a decentralized approach: authorization may be determined without having to consult some central authorization server, and users may choose to further delegate their authority without having to refer to a Central Authority.

Applications that use KeyNote [3] to provide decentralized authorization are beginning to emerge. For example, the Apache web server (version 1.3.6, and higher) uses KeyNote credentials to provide authorization for web-page access. As applications that make authorization decisions based on schemes such as KeyNote proliferate, then the numbers of users maintaining credentials will rise. That users may choose to further delegate their authorization will further encourage its use, both in open and in closed systems.

With this decentralized authorization approach comes the possibility that users may be delegated authorizations that may not quite match their expectations. For example, Alice likes the particular service that Bob's mobile phone credential authorizes, she makes a duplicate, and requests Bob's service provider for (a credential authorizing) an identical service. However, suppose that Bob is a business and Alice is a private individual, then it is possible that Alice's request will fail as she does not hold business credentials. Rather than failing, Alice may be willing to accept an alternative credential offering authorization for an approximately similar service, to some degree that is acceptable to Alice.

This paper explores how KeyNote might be used in practice to support such degrees of imprecision when making authorization and/or delegation judgments. Supporting imprecision for retrieval systems has been extensively considered in the literature on similarity-based retrieval for Case-Based Reasoning (CBR) systems [7, 6]. Imprecision permits answers that may not formally meet the query condition, but can be considered 'close enough'. For example, a property database search for two bedroomed apartments with monthly rent under $1,000 might return, as one of its less-relevant answers, a three bedroomed apartment with monthly rent $900. We draw on these case based reasoning techniques to provide assistance in programming KeyNote credentials and making authorization decisions.

Section 2 briefly describes the KeyNote approach to decentralized authorization. The reader should refer to [2, 3] for a more detailed exposition. Section 3 considers how similarity measures can be used to introduce controlled imprecision during authorization. The reader is referred to [7, 6] for an introduction to similarity measures and Case Based Reasoning. . Section 4 considers the problem of ensuring a consistent treatment of similarity during certificate reduction. Our solution to this problem also provides a direction for a practical implementation.

2 Authorization and Delegation

Credentials bind public keys to the authorization to perform various actions. For example, Alice may hold a credential, signed by her manager binding her public key to the authorization to have `read` access to a particular file. In practice, authorization is achieved by a collection of credentials that exhibit the necessary trust relationships between their keys. For example, we trust Alice's public key (for `read`) if her manager's public key is trusted to delegate `read`, 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, then a network application must determine whether a particular public key is authorized to request a particular action. This *compliance checking* is done by the trust management system.

EXAMPLE 1 A telecommunications company provides service of varying quality to private and business customers. A service provider with public key `Ksteve` is trusted to grant access authorization to new customers. This is expressed in terms of the following KeyNote credential.

```
Authorizer: "POLICY"
Licensees:  "Ksteve"
Conditions: app_domain=="telephone" &&
       ( (Cust=="private" && 0<=@Qual && @Qual<=1)
       ||(Cust=="business" && 1<=@Qual && @Qual<=2));
```

This *policy* credential defines the conditions under which requests from the licensee key `Ksteve` may be trusted by the telecommunications company. These conditions are defined using a C-like expression syntax in terms of *action attributes* `app_domain`, `Cust` and `Qual` which characterize the circumstances of a request. In this case, the service provider has been granted authorization to establish `telephone` connections as a `private` or `business` customer, with quality in the ranges [0..1] and [1..2], respectively. Note that the KeyNote operation `@Qual` converts the string attribute variable `Qual` to the integer value it represents. For the purposes of illustration we use names to represent public keys (and do not provide signatures).

The service provider has the authority to delegate this trust to customers (represented by their public keys). Alice is granted authorization as a private customer (quality 1). She in turn, decides to permit Angela to use this service at a lower quality. This is reflected by the following credentials.

```
Authorizer: "Ksteve"
Licensees:  "Kalice"
Conditions: Cust=="private"
        && @Qual<=1;
```

```
Authorizer: "Kalice"
Licensees:  "Kangela"
Conditions: Cust=="private"
        && @Qual==0;
```

Angela presents both these credentials when requesting a (`app_domain`) `telephone` connection as a (`Cust`) `private` customer with (`Qual`) quality 0. This results in a compliance check by the Trust Management system which finds a valid delegation chain from the trusted key `Ksteve` to `Kangela`, and thus her request is authorized, △

3 Similarity Measures

Conventional Trust Management systems can be regarded as relying on an absolute similarity measure $\_ \approx \_$, where authorizations x and y are similar ($x \approx y$), if and only if they are equal. Generalizing this to a relative measure allows a degree of similarity to be given: $x \approx y$ returns a value indicating the degree of similarity between x and y, ranging from 0 (not similar) to 1 (identical). For example, for telephone authorization one might define that

$$[\texttt{Cust=private}, \texttt{Qual=1}] \approx [\texttt{Cust=business}, \texttt{Qual=1}] = 0.9$$

$$[\texttt{Cust=private}, \texttt{Qual=1}] \approx [\texttt{Cust=private}, \texttt{Qual=0}] = 0.8$$

This reflects that authorizations for private and business customers with service quality 1 are more 'similar' than for that for two private customers with service quality 0 and 1, respectively.

DEFINITION 1 Given a relative similarity measure, then an extended compliance check to determine whether a key K is authorized for action x must check whether K is authorized for some action x', where $x \approx x' \geq n$, for a degree of similarity $0 \leq n \leq 1$ that is acceptable to the application security policy. □

EXAMPLE 2 Given `Cust` attribute values $c, d \in \{\texttt{private}, \texttt{business}\}$ and `@Qual` attribute values $0 \leq (q, r) \leq 4$, then a general similarity measure for Example 1 is defined as

$$(c, q) \approx (d, r) \equiv \begin{cases} 1.0 - 0.1 \times |q - r| & \text{if } (c = d) \\ 0.8 - 0.1 \times |q - r| & \text{otherwise} \end{cases}$$

As noted above, this reflects a belief that the authorizations of a business and a private customer with the same quality are 'different' by 0.2. When the quality of one customer is incremented/decremented, then the similarity decreases by 0.1, and so forth. The following table reflects the similarity measure calculated for the authorizations that can be delegated by the service provider `Ksteve`. The pairs (c, q) represent (`Cust`,`Qual`) values.

$\approx$	(p,0)	(p,1)	(b,1)	(b,2)
(p,0)	1.0	0.9	0.7	0.6
(p,1)	0.9	1.0	0.8	0.7
(b,1)	0.7	0.8	1.0	0.9
(b,1)	0.6	0.7	0.9	1.0

Given the credentials from Example 1, then a Trust Management system that accepted a similarity degree of 0.8 or higher, would authorize Alice's (private customer, quality ≤ 1) request to make a business call with quality 1 since $(\texttt{p,1}) \approx (\texttt{b,1}) = 0.9$, but not a business quality call with level 2. Angela (`private` customer, quality 0), would be authorized to make a private quality 1 call, but would not be permitted to make business quality calls. The trust management system can increase or decrease its authorization flexibility by adjusting the similarity degree. △

4 Similarity and Certificate Reduction

In addition to judgments on authorization, a Trust Management system may also be used to provide assistance in determining whether it is safe to write and sign a new credential [5]. A principal presents a set of credentials and asks for a new credential containing a specific authorization. If the principal is authorized then it is safe to write and sign a new single credential. This *certificate reduction* [4] can be used to reduce the size of delegation chains and may also provide for a degree of anonymity [1]: long delegation chains may reveal sensitive information about how authority was acquired. For example, if Steve reduces the certificate chain to Angela's authority (Example 1) by issuing a single credential for the authorization she holds, then Angela may use the telecommunications service without revealing how she acquired the authority.

In this paper we consider only *defacto* delegation [5], that is, delegation in terms of a certificate reduction that does not confer any additional authorization on the requester. We do not consider *dejure* delegation [5] where the requester may acquire authorization, in addition to what is held. However, we note in [5] that both approaches may be implemented in terms of certification reduction and, therefore, the results in this paper are applicable to dejure delegation.

Delegation by reduction may generate a cascading effect along delegation chains and make the similarity approach ineffective. For example, a trust management system that accepts a similarity degree of 0.8 or higher, would not permit Angela (Examples 1,2) to receive a business service with quality 1, since $(\texttt{p,0}) \approx (\texttt{b,1}) = 0.7$. However, in this case, Angela is effectively authorized to receive a private service with quality 1. Therefore, she may present her delegation chain and ask `Ksteve` to reduce it and write the following credential for this service (similarity 0.8).

```
Authorizer: "Ksteve"
Licensees:  "Kangela"
Conditions: Cust=="private"
         && @Qual==1;
```

She may now present this new credential when requesting a business service with quality 1 (similarity 0.9), which is authorized.

This cascade problem can be prevented if credentials incorporate conditions relating the degree of similarity and/or approximation for which the credential was signed. In the situation above, Steve should record within Angela's credential the fact that it was issued based on authorization credentials presented with degree of similarity 0.8.

We use the following informal strategy when writing KeyNote credentials. Given a similarity relationship $\approx$ between the possible attribute values for an action attribute variable `Action`, then introduce an additional action attribute `Degree` and codify

$$\texttt{\&Degree <= Action} \approx X$$

within any credential that authorizes the X action to its licensee. Note that `&Degree` converts the string attribute `Degree` to a floating point number. The policy credential specifies, as policy, the degree of similarity n that is acceptable by including

$$\texttt{\&Degree >= } n$$

A compliance check by a trust management system to determine whether a key is authorized for action X must find a suitable value V for attribute `Degree` such that K is authorized for `Action=`X and `&Degree=`V. The value V may be regarded as reflecting how 'approximate', in this case, the authorization for A is. If a new credential is to be written for this authorization then it should reflect this degree of approximation, that is,

$$\texttt{\&Degree <= (Action} \approx X) - (1 - V)$$

For purposes of flexibility, the compliance check should find the *highest* possible value for V.

Example 3 Continuing Examples 1,2, private customer Alice, authorized for service qualities {0,1}, is issued the following revised credential.

```
Authorizer: "Ksteve"
Licensees:  "Kalice"
Conditions:
  Cust=="private" ->{&Degree <= 1.0;};
  Cust=="business"->{&Degree <= 0.9-0.2*(&Qual-1.0);};
```

The absence of an absolute value function `abs()` in KeyNote has resulted an a somewhat counter-intuitive coding of the $\approx$ relation in the credential. The reader may confirm that this correctly codifies the $\approx$ relation for authorizations that can be delegated from `Ksteve` (values from the table in Example 2).

A revised policy credential for `Ksteve` specifies that similarity to a degree of 0.8 or higher is acceptable:

```
Authorizer: "POLICY"
Licensees:  "Ksteve"
Conditions: &Degree>=0.8 &&
       app_domain=="telephone" &&
       ( (Cust=="private"  && 0<=@Qual && @Qual<=1)
       ||(Cust=="business" && 1<=@Qual && @Qual<=2));
```

Given these credentials, a compliance check verifies Alice's authorization to request a business quality 1 call, with a degree value of 0.9. That is, a compliance check, given action authorizer `Kalice` and action attribute values `Cust="private"`, `Qual="1"` and `Degree="0.9"`, returns `true`. Alice's request for a business quality 2 call would fail since no degree attribute value exists that satisfies the credential conditions along the delegation chain from `Ksteve` to `Kalice`.

Suppose that Alice issues the following credential to Angela:

```
Authorizer: "Kalice"
Licensees:  "Kangela"
Conditions: Cust=="private" &&
   &Degree <= 1.0 - 0.1*(@Qual);
```

Angela is primarily authorized for quality-0 private service. However, since the policy states that a similarity degree of 0.8 or higher is acceptable, then Angela is also authorized for quality-1 private service. In this case, the compliance check hold when the degree attribute is 0.9. If Angela had asked Steve to perform a reduction on these credentials, then the following credential would be generated.

```
Authorizer: "Ksteve"
Licensees:  "Kangela"
Conditions: Cust=="private" &&
   &Degree <= 0.9
```

Angela cannot use this credential to gain cascaded authorization for business service. △

EXAMPLE 4 A mobile phone service provider issues service credentials based on the following features.

0. `rc`: receive an incoming call,
1. `sc`: make an outgoing call,
2. `rv`: retrieve voicemail messages,
3. `rt`: receive text messages,
4. `st`: send text messages,
5. `rd`: receive data, such as fax,
6. `sd`: send data, such as fax,
7. `re`: retrieve email messages,
8. `se`: send email messages,
9. `uw`: use the web.

The feature set for a particular mobile phone is coded as a ten bit vector A that is indexed by $i \in [0..9]$, corresponding to the features $\texttt{rc}, \texttt{sc}, \ldots, \texttt{uw}$. $A_i = 1$ or $A_i = 0$ indicates the presence or absence of feature i, respectively.

One simple similarity measure between feature sets averages the number of different features, that is, $1 - (\sum_{i=0}^{n-1}(A_i \oplus B_i))/n$, where $(A_i \oplus B_i)$ gives the exclusive-or of the i^{th} bit positions of vectors A and B, and n is the total number of features. Including weights for each feature, generalizes this to the weighted nearest neighbor formula:

$$A \approx B \equiv 1 - \frac{\sum_{i=0}^{n-1} w_i \times (A_i \oplus B_i)}{\sum_{i=0}^{n-1} w_i}$$

where w_i gives the weight associated with feature i. Several Case Based Reasoning systems use a weighted nearest neighbor formula to provide the basis of a similarity measure between feature sets [7].

A challenge in using the nearest neighbor formula is the determination of suitable weights for features. For the current example, we assume that the weights are determined on the basis of the cost of the feature, and that the features above are listed in increasing order of cost. Basic voice communication is cheapest, while web browsing is most expensive.

For the sake of simplicity, suppose that $w_i = 2^i$, that is, feature costs rise exponentially with the value of i. While this is unrealistic in practice, it suits the purpose of the current example since the nearest neighbor formula above simplifies to

$$A \approx B \equiv 1 - \frac{A \oplus B}{2^n - 1}$$

when A and B are regarded as unsigned integers and $A \oplus B$ is a bitwise exclusive-or operation on these numbers.

Given the above weighting scheme, then the feature set with authorizations `rc`, `sc`, `rv`, `rt` and `st` is represented by the unsigned integer 31. This corresponds to the sum of weights associated with rc (2^0), sc (2^1), rv (2^2), rt (2^3), st (2^4). Using the general credential writing strategy described above, a KeyNote credential that delegates authorization for these features to Alice could be written as:

```
Comment: Authorize rc+sc+rv+rt+st (=31)
Authorizer: "Ksteve"
Licensees: "Alice"
Conditions: &Degree <= 1.0-(@Ops[+]31)/1023.0;
```

Where [+] represents an exclusive or operation[1]. Suppose that the following policy was in effect.

```
Authorizer: "POLICY"
Licensees: "Ksteve"
Condition: App_Domain == "Mobile" &&
           &Degree >= 0.91
```

Under this policy, Alice is also authorized to receive or to send data since we can compute:

$$\begin{aligned}\{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}\} \approx \{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}, \mathbf{rd}\} &= 0.968 \\ \{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}\} \approx \{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}, \mathbf{sd}\} &= 0.937\end{aligned}$$

However, Alice is not permitted to simultaneously receive *and* send data since,

$$\{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}\} \approx \{\mathtt{rc}, \mathtt{sc}, \mathtt{rv}, \mathtt{rt}, \mathtt{st}, \mathbf{rd}, \mathbf{sd}\} = 0.906$$

which is below the degree of similarity that is acceptable by the policy. This requires some explanation. Authorization for feature rd or wd is taken to mean that the user may open a connection to receive data or to send data, respectively. It is assumed that opening a connection to send and receive data requires a credential that authorizes both, simultaneously. The policy above ensures that Alice, while she may send and receive data, cannot open a two-way connection, corresponding to a modem connection, for example. △

[1] This is used for demonstration purposes only. The current KeyNote specification [3] does not provide bitwise operations; their incorporation would be straightforward.

5 Discussion and Conclusion

This paper considers how similarity techniques that are used by case-based reasoning systems might be adapted to support degrees of imprecision when delegating authority based on KeyNote credentials. Imprecision based compliance checking permits an operation when the credentials provided do not strictly provide the authorization, but can be considered 'close enough'. Similarity measures are used to specify how 'close' authorizations for different actions are to each other.

Section 3 redefines compliance checking in terms of similarity measures; a key is considered authorized for an action if it is authorized for another similar action, to some degree of similarity. This may lead to inconsistencies resulting from cascading effect along delegation chains. Section 4 considers this problem and describes how, by codifying the similarity measure within the credentials, it can be avoided.

The practical approach taken in this paper is informal. We have proven elsewhere, using a formal model of delegation, that our general strategy is consistent. Given certain consistency properties on $\approx$ then the compliance check defined in Section 3 can be re-cast in terms of a conventional compliance check by transforming credentials to extended credentials that appropriately encode the $\approx$ measure. A benefit of this result is that it provides a direction for the implementation of the compliance check. Given the action attributes of a request, then a value for `Degree` must be found such that the usual KeyNote compliance check holds. A KeyNote compliance check corresponds to a search of the delegation graph for a valid path to the action authorizer. This search can be modified to consider suitable values for `Degree`.

The approach proposed in this paper can be regarded as providing a basis for a security 'dial' for application systems. The dial controls the degree of imprecision in authorization that the system is willing to tolerate; it can be tuned to reflect the desired level of flexibility. This paper demonstrates that support for such an approach is feasible. Further research is needed that applies the experience that Knowledge Engineers have in developing similarity measures, to the development of similarity measures for authorization.

Acknowledgments

The motivation for Example 4 is due to Thomas Quillinan. My thanks to him for many thoughtful discussions on this work and to the anonymous referees for their useful feedback on an early version of this paper.

References

[1] T. Aura and C Ellison. Privacy and accountability in certificate systems. Technical Report HUT-TCS-A61, Helsinki University of Technology, Laboratory for Theoretical Computer Science, 2000.

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

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

[4] C Ellison et al. SPKI certificate theory. September 1999. Internet Request for Comments: 2693.

[5] S.N. Foley. Trust management and whether to delegate. In *International Workshop on Security protocols*, Cambridge, UK, 2001. Springer.

[6] Hugh Osborne and Derek Bridge. Models of similarity for case-based reasoning. In E.Cambouropolos M.Ramscar, U.Hahn and H.Pain, editors, *Procs. of the Interdisciplinary Workshop on Similarity and Categorisation*, pages 173–179, 1997.

[7] I Watson and F Marir. Case based reasoning review. *The Knowledge Engineering Review*, 9(4), 1994.