Aggregating Trust Using Triangular Norms in the KeyNote Trust Management System

Simon N. Foley, Wayne Mac Adams, Barry O'Sullivan

Cork Constraint Computation Centre,
Department of Computer Science,
University College Cork, Ireland
[s.foley,w.macadams,b.osullivan]@cs.ucc.ie

Abstract. A Trust Management model that provides a measure of the degree to which a principal is trusted for some action is proposed. At the heart of the model is the notion that triangular norms and conorms provide a natural and consistent interpretation for trust aggregation across delegation chains. It is argued that specifying how trust is aggregated is as important as specifying a degree of trust value in an attribute certificate and, therefore, in stating the degree to which a principal trusts another, the principal should also state how that trust may aggregate across delegation chains. The model is illustrated and has been implemented using a modified, but backwards-compatible, version of the KeyNote Trust Management system.

1 Introduction

Trust Management [1,4,9,20], as originally defined by [5], is an approach to constructing and interpreting trust relationships between principals such as users, groups, roles, hardware-devices, etc. These *well placed trust* relationships [25], defined in terms of relatively static attributes that are perceived by a trusting party, are constructed as a graph of credentials encoding the conditions under which a principal is willing to trust some action. Trust Management systems are intended to support decentralized security: individual trust statements are encoded as cryptographic certificates that can be safely distributed across the network and reasoned over without the need for trusted authorization servers mediating over centralized policy state.

While a Trust Management system determines whether a principal is trusted (authorized) for some action, reputation (trust) schemes such as [14,18] are used to provide some measure of the degree of trust between principals. For example, Slashdot Karma gives a measure of an individual's standing in that message board community. Many Trust Management systems provide a binary decision—whether or not a principal is trusted—and do not consider the degree to which a principle is authorized for an action. In this paper a model is developed whereby a Trust Management decision is given in terms of a measure/degree of trust.

There is much published research on how reputation trust between principals might be measured and is not the focus of this paper. In this paper we assume that principals assert subjective measures of well placed trust when defining the conditions under which they trust another principal. The challenge is then to determine how these trust values should *aggregate* across the trust relationships that make up an attribute certificate delegation graph. We argue that *Triangular Norms* provide a natural approach to aggregating trust. Triangular norms and conorms are classes of well-understood aggregation operators that are used to combine values in the metric space [0..1] [8,26] and have been used to aggregate knowledge for a variety of applications such as fuzzy-logic [8], risk management [11], multimedia databases [10] and medical decision support systems [6].

In this paper, a model of quantitative Trust Management is developed whereby delegation certificates may specify a degree of trust. Rather than prescribing specific aggregation operators, the model allows the user to also specify, as part of the delegation certificate, how the delegated trust may be aggregated. This model has been implemented in an extended version of the KeyNote Trust Management system. The extension provides backwards compatibility with the standard KeyNote system whereby KeyNote compliance values can be considered to implement degrees of trust, aggregated by Gödel fuzzy logic t-norm/conorm operators min and max.

The paper is organized as follows. Section 2 considers how triangular norms and conorms might be used to aggregate trust across a delegation network and argues that treating the calculation as an inclusion-exclusion or two-terminal network reliability style problem is not appropriate. A model for trust aggregation is described in Section 3 and Section 4 presents a series of KeyNote credentials to illustrate its use. Section 5 discusses the implementation of this model. Section 6 reviews related research and discusses the results of the paper.

2 Quantifying Trust

Let the statement $A \xrightarrow{x} B$ denote an assertion by principal A that she trusts (delegates trust to) the principal B to a degree $x : [0..1]$. We assume that A's trust of B increases linearly with the value of x, whereby $x = 0$ represents no trust and $x = 1$ denotes complete trust.

A set of trust delegation statements form a directed graph with labeled arcs between principals representing trust/delegation statements. Figure 1 illustrates a delegation graph with statements $A \xrightarrow{0.8} B$, $B \xrightarrow{0.9} D$ and so forth. Given such a graph, we are interested in determining how transitive trust, that is, the implicit degree of trust $trust(X, Y)$ from a principal X to principal Y, should be computed. For example, the implicit degree of trust from A to D in Figure 1.

2.1 Trust Aggregation using Max and Min

One strategy for computing $trust(X, Y)$ in a delegation graph is to compute the maximum degree of trust over all delegation chains from X to Y, whereby the degree of trust for a single chain is the minimum of the degrees along its path. For example, given the graph in Figure 1, the single chain from A to D

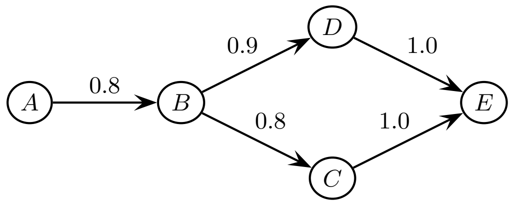

Fig. 1. Quantitative Delegation Graph

results in $trust(A, D) = \min(0.8, 0.9)$. The two chains $A \overset{0.8}{\to} B \overset{0.9}{\to} D \overset{1.0}{\to} E$ and $A \overset{0.8}{\to} B \overset{0.7}{\to} C \overset{1.0}{\to} E$ connect A and E and $trust(A, E) = \max(0.8, 0.7) = 0.8$. This max-min paths strategy is implemented as a breath first search of the delegation graph in the KeyNote Trust Management system [4] where (an enumerated set of) KeyNote compliance values can be effectively interpreted to represent degrees of trust measures.

The min operator may not always provide an intuitive aggregation of trust values along the path of a chain. For example, given Figure 1, while B considers its trust of D and C to be different, this distinction is lost in the calculation of $trust(A, D) = 0.8$ and $trust(A, C) = 0.8$, despite B's involvement in both chains. Had A's trust of B been less that 0.8 then this distinction would be apparent. An alternative operator for aggregating trust along a chain is the (probabilistic) product operator $\times$. Using this operator in Figure 1 gives $trust(A, C) = 0.64$ and $trust(A, D) = 0.72$ which could be regarded as better reflecting the knock-on effect of trust by B. Such use of probabilistic product to aggregate trust along a chain of trust is proposed as a way to provide more fine-grained treatment for GnuPGP's degree of trust measure [15].

Similarly, the max operator does not necessarily provide an intuitive operator with which to aggregate the degree of trust across multiple delegation chains. Continuing the example above, E's trust from B might be considered to be an *accumulation* of trust obtained via chains $B \overset{0.9}{\to} D \overset{1.0}{\to} E$ and $B \overset{0.7}{\to} C \overset{1.0}{\to} E$, and therefore, should be greater than $trust(B, E) = \max(0.9, 08)$. Computing $trust(B, E) = 0.9 + 0.8 - 0.8 * 0.9 = 0.98$ using probabilistic sum to aggregate trust across chains $B \overset{0.9}{\to} D \overset{1.0}{\to} E$ and $B \overset{0.8}{\to} C \overset{1.0}{\to} E$ may be regarded as better reflecting how E's trust accumulates from multiple chains. Probabilistic sum is also used in GnuPGP [15] to aggregate across different chains where the calculation of $trust(X, Y)$ is treated as an two-terminal network reliability-style problem [7,22].

2.2 Trust Aggregation using Triangular Norms and Conorms

Computing $trust(X, Y)$ requires selection of an operator (denoted $\otimes$) used to aggregate trust along a chain and an operator (denoted $\oplus$) used to aggregate across different chains. We argue that *triangular norms* and *triangular conorms*, respectively provide suitable aggregation operators.

A triangular norm (hereafter referred to as t-norm) operator $\otimes$ is commutative and associative; its monotonicity ensures that its use for aggregating trust values along a delegation chain does not result in an amplification of trust, that is, $x \otimes y \leq \min(x, y)$ for $x, y : [0..1]$ and min is the largest pointwise t-norm.

We use a triangular conorm (t-conorm) $\oplus$ to aggregate trust across different chains. A t-norm has a corresponding t-conorm under the DeMorgan style law: $x \oplus y = 1 - (1 - x) \otimes (1 - y)$. The max operator is the smallest t-conorm and for any t-conorm operator $\oplus$, then $x \oplus y \geq \max(x, y)$ for $x, y : [0..1]$. Thus, t-conorm based aggregation of trust chains provides monotonic trust, that is, adding further trust statements/chains to a delegation graph does not result in a decrease in the value of $trust(X, Y)$. The t-norm and t-conorm operators can be interpreted as forms of fuzzy disjunction and conjunction, respectively [8]. Table 1 defines a number of common t-norm and their respective t-conorm operators.

	t-norm $x \otimes y$	t-conorm $x \oplus y$
Probabilistic	$x \times y$	$x + y - x \times y$
Gödel	min	max
Lukasiewicz	$\max(0, x + y - 1)$	$\min(1, x + y)$

Drastic	if $x = 1$ then y
	elseif $y = 1$ then x
	else 0
		if $x = 0$ then y
		elseif $y = 0$ then x
		else 1

Compensating Trust(e)	if $e \leq (x, y) \leq 1$
	then $x \times y$
	else $min(x, y)$
		if $e \leq (x, y) \leq 1$
		then $x + y - x \times y$
		else $max(x, y)$

Table 1. Some t-norms and t-conorms.

2.3 Accumulating Trust

The max-min style calculation of $trust(X, Y)$ as a sum (t-conorm) of products (t-norm) along the chains between X and Y does not necessarily generalize to other t-norm/conorm operators. For example, suppose that probabilistic product is used as a chain-aggregator $\otimes_{\mathcal{P}}$ along paths $A \overset{0.8}{\rightarrow} B \overset{0.9}{\rightarrow} D \overset{1.0}{\rightarrow} E$ and $A \overset{0.8}{\rightarrow} B \overset{0.7}{\rightarrow} C \overset{1.0}{\rightarrow} E$ in Figure 1. This results in trust values 0.72 and 0.64, respectively; using probabilistic sum $\oplus_{\mathcal{P}}$ to aggregate across these chains results in calculation $trust(A, E) = 0.899$. This calculation may not necessarily reflect the trust intentions of A, for example, E might be owned by B. In Figure 1, A states that she trusts B to degree 0.8. However, as a consequence of statements of other principals, using this calculation results in E holding more trust (0.899) from A even though it originated exclusively via B. In this case the most trust (from A) that B should be able to delegate to E should be degree 0.8.

Intuitively, we might expect to be able to address this issue by reducing the subgraph connecting B to E to a single arc and computing:

$$trust(A, E) = trust(A, B) \otimes_{\mathcal{P}} trust(B, E) = 0.8 \otimes_{\mathcal{P}} (0.8 \oplus_{\mathcal{P}} 0.9) = 0.78$$

However, the graph in Figure 2 illustrates that such a rewriting strategy is not applicable in general. In this case it it not immediately clear how D's trust should be proportioned to C and E so as to ensure that E gets the maximal trust available while ensuring that D does not delegate more trust than the 0.9 degrees obtained from B.

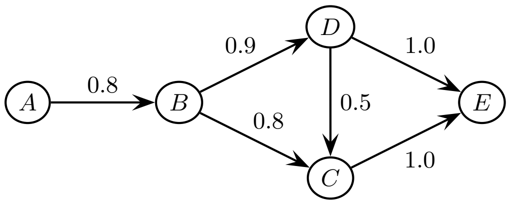

Fig. 2. Quantitative Delegation Graph

The probabilistic sum operator treats trust as a quantity that can effectively be accumulated as more statements about trust are made; the more statements that X makes delegating trust to Y then, the more that Y's trust of X accumulates, that is, $trust(X, Y)$ increases (bounded by 1). This gives rise to the following principle for computing trust.

Principle of Preservation of Trust A principal may not delegate more trust than it has accumulated. Note that this principle is expressed as a safety property in that it does not require conservation of trust (in a liveness sense).

Trust delegation may be treated as a network flow-style problem. However, we argue that it does not correspond to the two-terminal network reliability problem [22], nor its variation suggested in [15], as both schemes can result in a 'double-counting' of trust statements. For example, using [15] to compute trust as the probabilistic sum of the trust along paths $ABDE$, $ABDCE$ and $ABCE$, results in $trust(A, E) = 0.989$. This result violates the principle of trust accumulation as it implies that B effectively delegates more trust to E (0.989) than it accumulated from A (0.8).

In the next section we propose a model in which $trust(X, Y)$ can be used with any t-norm for trust aggregation along a chain and with any t-conorm to aggregate trust across different chains, while preserving the principle of trust accumulation.

3 A Model of Trust Aggregation

The calculation of $trust(X, Y)$ treats trust as a material-like quantity that flows through the delegation graph. In many cases the degree of trust delegated is the degree of trust that flows, for example, 0.8 degrees of trust 'flows' from A to B as a consequence of the delegation $A \overset{0.8}{\rightarrow} B$ in the graph in Figure 1. Given this trust held by B, suppose that we consider that 0.9 of this flows to D as a consequence of $B \xrightarrow{0.9} D$ (under t-norm $\otimes_{\mathcal{P}}$). For the reasons discussed in the previous section, permitting a further 0.8 (of the trust from A to B) to flow from B to C will violate the Principle of Preservation of Trust since probabilistic sum of A's trust flowing, via B to C and D must be less than or equal to the trust flowing to B. In this section we define the calculation of $trust(X, Y)$ in terms of a search for a set of flow of trust labels for the delegation graph that preserves the Law of Preservation of Trust.

Let $trust(X, Y)$ denote a degree of trust that arbitrary principal Y can accumulate from principal X over a delegation graph. Given nodes X, Y and Z and arc $Y \xrightarrow{x} Z$ in the delegation graph, then let the value $flow(X, Y, Z)$ represent some portion (ranging from nothing (0) to everything ($trust(X, Y)$)) of the trust that Y holds, originating from X, that is passed on to Z by Y.

The value of $trust(X, Y)$ is a solution to the following constraint problem and is based on a search for a suitable configuration of the function $flow$ that ensures the principle of preservation of trust.

- If no directed path exists in the graph from X to Y then $trust(X, Y) = 0$.
- The (cross chain aggregation of) flow from principal Y to others cannot exceed the degree of trust from X that is accumulated by Y.

$$trust(X, Y) \geq \bigoplus_{\forall Z, x | Y \xrightarrow{x} Z} flow(X, Y, Z).$$

- A node may not accumulate more trust than the cross chain aggregation of the trust that it receives from others.

$$trust(X, Y) \leq \bigoplus_{\forall Z, x | Z \xrightarrow{x} Y} (flow(X, Z, Y) \otimes x).$$

The intuition behind this calculation is to effectively find flow labelings for the graph such that a subsequent max-min/network reliability style calculation of $trust(A, B)$ preserves the principle of preservation of trust accumulation.

As an example, Figure 3 depicts a solution of this problem: each node $Z \in \{C, D, E\}$ is decorated with the value $trust(A, Z)$ and each delegation statement $X \xrightarrow{x(y)} Y$ indicates the degree of trust x and the value $flow(A, X, Y) = y$. Note that in this solution, all available trust flows from B to D to E. However, as a consequence, none of the available trust held by B may flow to C; any higher proportion and the Principle of Preservation of Trust is violated as the flow from A to B exceeds 0.8. Note that this is just one solution to the above problem, it is not necessarily optimal. Figure 4 gives an alternative solution. Section 5 considers the implementation of $trust(X, Y)$ in practice.

4 Trust Aggregation in QKeyNote

Sections 2 and 3 consider the quantification of *unconditional* trust, that is, the degree of trust delegated from one principal to another is not related to any

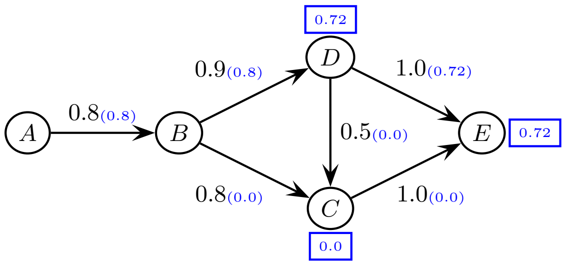

Fig. 3. Delegation graph solution including $flow$s originating from A

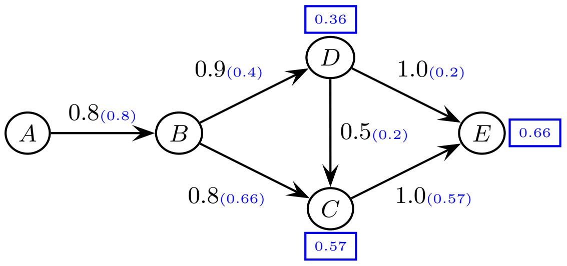

Fig. 4. Alternative solution including $flow$s originating from A

particular action or attributes of the principals. In this section quantified *conditional* trust is considered whereby a delegation statement $A \overset{p:x}{\rightarrow} B$ is interpreted to mean that a principal A asserts that she trusts another principal B for permission p (related to some action) to a degree $x : [0..1]$. This section explains how the KeyNote Trust Management system [4] can be extended to support these delegation statements thereby providing decentralized support for quantitative trust management. A KeyNote delegation credential encodes the delegated 'permission' as a condition over attribute values. The approach is illustrated using examples based on a photograph sharing trust policy [13] which uses a probabilistic network to represent trust relationships between variables (attributes).

4.1 KeyNote Compliance Values as Degrees of Trust

A KeyNote credential is a statement that delegates trust from an authorizer to a licensee constrained by some condition. The condition is given using a C-like expression syntax in terms of attributes that are used to characterize action(s) for which the licensee is trusted (by the authorizer). For example, consider a server that permits trusted users access stored photographs based on image tags. The server owner specifies that Alice (public key `Ka`) is trusted to access photographs tagged as public. This is specified by the KeyNote policy credential:

```
Authorizer: POLICY
Licensee: Ka
Conditions: App_Domain=="PhotoShare" && Tag == "public";
```

The condition for this delegation is defined in terms of action attributes `App_Domain` and `Tag`. For the purposes of this example we assume that a photograph is tagged either as `public` or `private`. When Alice requests access to a particular photograph, the application uses the KeyNote query engine to determine whether there is a chain of trust from `POLICY` to the requesting key `Ka` with conditions satisfying the [`App_Domain`$\leftarrow$`"PhotoShare"`; `Tag`$\leftarrow$`"public"`].

In practice, the condition on a KeyNote credential evaluates to a value from a user-enumerated (ordinal) set of *compliance values* that range from `_MIN_TRUST` to `_MAX_TRUST`, for example, the compliance set={`false,true`}, for the above policy. When a return value is not specified in the condition field then the default return value is `_MAX_TRUST` when the condition is satisfied, otherwise `_MIN_TRUST` is returned. The following credential, where Alice delegates trust to Bob `Kb`, specifies a compliance value of 0.7. This value defines Alice's degree of trust in Bob concerning public photographs.

```
Authorizer: Ka
Licensee: Kb
Conditions: (App_Domain=="PhotoShare"
&& Tag=="public") -> 0.7;
Signature: ....
```

Compliance values in KeyNote conventionally define an enumerated and discrete datatype, for example, ⟨`0,0.3,0.5,0.7,1.0`⟩ with `_MIN_TRUST=0.0` and `_MAX_TRUST=1.0`. A conventional KeyNote query corresponds to a calculation of the maximum compliance value returned over the minimum compliance value along all delegation chains whose conditions satisfy the attribute binding given in the query.

4.2 Aggregating Compliance Values in QKeyNote

The KeyNote query engine has been extended to support compliance values from the real datatype [0..1] and a revised query algorithm (QKeyNote) is used to implement $trust(X, Y)$ for computing compliance values. If operators max and min are selected as the aggregation operators then a QKeyNote query behaves as a conventional KeyNote query.

Continuing the photograph sharing example, the policy is extended to authorize Clare (`Kc`), who in turn, delegates authority to Bob (`Kb`).

```
Authorizer: POLICY
Licensee: Kc
Conditions:
(App_Domain=="PhotoShare")
-> { {Tag=="public"} -> 0.6;
     {Tag=="private"} -> 0.2; }
Signature: ....
```

```
Authorizer: Kc
Licensee: Kb
Conditions:
(App_Domain=="PhotoShare"
&& Tag=="public") -> 0.8;
Signature: ....
```

Given these and the earlier credentials, a QKeyNote query as to whether Bob is trusted for [`App_Domain`$\leftarrow$`"PhotoShare"`; `Tag`$\leftarrow$`"public"`] returns compliance value 0.844 when probabilistic operators are used to aggregate, that is, it calculates $trust(\texttt{POLICY}, \texttt{Kb}) = (1 \otimes_{\mathcal{P}} 0.7) \oplus_{\mathcal{P}} (0.6 \otimes_{\mathcal{P}} 0.8) = 0.844$.

QKeyNote supports two predefined attributes that are treated similarly to other predefined attributes such as `_MAX_TRUST` and `_ACTION_AUTHORIZERS`. Attribute `_TNORM` specifies the t-norm operator that the QKeyNote query engine has been configured to use when aggregating trust along a chain. Attribute `_TCONORM` specifies the t-conorm operator that the QKeyNote query engine has been configured to use when aggregating trust across multiple chains. The default query engine configuration is `_TNORM="min"` and `_TCONORM="max"` providing the 'conventional' KeyNote query.

We argue that specifying how delegated trust is aggregated should be as important to the authorizer as is specifying the degree of trust within a credential. Therefore, a delegation of trust can be made conditional on the operators used to aggregate the trust by constraining the values of `_TNORM` and `_TCONORM` in its condition. For example, suppose that the server owner decides that Alice is trusted to access private photographs as long as probabilistic sum and product are used to aggregate this trust. This credential may be used only when the query engine is configured to use probabilistic sum and product for aggregation.

```
Authorizer: POLICY
Licensee: Ka
Conditions: (App_Domain=="PhotoShare"
&& _TNORM=="probProduct" && _TCONORM=="probSum
&& Tag=="private") -> 0.2;
Signature: ....
```

A partial order can be defined over the t-norm and t-conorm operators. Given t-norms $\otimes_t$ and $\otimes'_t$ then

$$\otimes_t \sqsubseteq \otimes'_t \equiv \forall x, y : [0,1] \bullet x \otimes_t y \leq x \otimes'_t y$$

The operator min is the top t-norm under this ordering, for example we have $\otimes_{\mathcal{P}} \sqsubseteq \min$. A similar ordering exists over t-conorms with max providing the bottom t-conorm. These orderings can be used within a credential to facilitate aggregation operator selection. For example, Alice trusts Bob to share private photographs on condition that he can never use this trust with other credentials to accumulate more than 0.6 degrees of trust. In this case the t-conorm used to aggregate across chains must be the max operator, while the t-norm can be any operator less than or equal to the min operator.

```
Authorizer: Ka
Licensee: Kb
Conditions: (App_Domain=="PhotoShare"
&& _TNORM<="min"  && _TCONORM=="max"
&& Tag=="private") -> 0.6;
Signature: ....
```

This credential cannot be used if the query engine is configured with probabilistic product and sum. However, for example the condition can be satisfied the query engine is configured with probabilistic product as t-norm and max as t-conorm. If a credential condition does not refer to `_TNORM` and `_TCONORM` then the delegation is applicable for any aggregation operator. Figure 5 lists the aggregation operators (and ordering) available in our current implementation of QKeyNote. Currently values for `_TNORM` and `_TCONORM` must be provided as part of the query. We are currently investigating how search for optimal values for these operators might form part of the query $trust(X, Y)$.

$$\begin{tikzcd}
\text{Gödel min} & \text{Drastic } \oplus_{\mathcal{D}} \\
\text{Compensating } \otimes_{\mathcal{C}}^{e} \arrow[u, dashed] & \text{Lukasiewicz } \oplus_{\mathcal{L}} \arrow[u] \\
\text{Probabilistic } \otimes_{\mathcal{P}} \arrow[u, dashed] & \text{Probabilistic } \oplus_{\mathcal{P}} \arrow[u] \\
\text{Lukasiewicz } \otimes_{\mathcal{L}} \arrow[u] & \text{Compensating } \oplus_{\mathcal{C}}^{e} \arrow[u, dashed] \\
\text{Drastic } \otimes_{\mathcal{D}} \arrow[u] & \text{Gödel max} \arrow[u, dashed]
\end{tikzcd}$$

Fig. 5. Type system aggregation operator orderings $\sqsubseteq$

4.3 Accumulating Compliance Values in QKeyNote

The delegation graph built by the QKeyNote query engine supports multiple delegation statements between arbitrary principals so long as each delegation statement is unique. Credential uniqueness ensures that a malicious principal cannot accumulate more trust than he holds by presenting multiple copies of the same credential. For example, based on an evolving relationship, Alice issues a credential to Bob, every day, specifying that she incrementally trusts him to degree 0.01 to access private photographs:

```
Authorizer: Ka
Licensee: Kb
Conditions: (App_Domain=="PhotoShare"
&& _TNORM=="probProduct" && _TCONORM=="probSum"
&& Tag=="private" && Nonce = "123456"
&& Date >= "20100101" && Date<= "20101231"
&& _Action_Authorizers==Kb)  -> 0.01;
Signature: ....
```

The nonce provides credential freshness and Bob may accumulate and use multiple credentials over time, subject to an expiry date. Bob may not delegate this trust further (`_Action_Authorizers==Kb`). For example, Bob may present three credentials (different nonces) to prove a 0.0297 degree of trust from Alice (for accessing private photographs). In this case each delegated degree of trust is very small and could be regarded as form of *micro-trust* that can be accumulated over time into a worthwhile relationship. Rather than expending the cost of a public-key signature to generate and validate each each micro-trust statement, a hash-based micropayment scheme such as [12] could be used, whereby each micro-payment on a hash-chain corresponds to a micro-trust credential of value 0.01. Following [12], a micro-trust contract, analogous to a micropayment contract, is issued as KeyNote credential by Alice, to Bob indicating that she is willing to eventually trust Bob to a high degree, that will be accumulated as a series of micro-trust 'payments'.

4.4 Compensating Aggregation Operators

One concern over using conventional arithmetic and/or probabilistic sum is that when making decisions humans do not necessarily aggregate in a linear manner [28], that is, there may be potential for non-linearity in the way that they perceive combinations. In [6,11,19] a compensating uni-norm operator $\odot^e$ is described for aggregating in a non-linear manner, for neutral element $e : [0..1]$. Intuitively, this uni-norm operator may be thought of as a combination of probabilistic product when operand severity values are less than e, and probabilistic sum when operand severity values are greater than e. Using this operator, for example with $e = 0.6$, to aggregate trust along a chain causes the calculation to be less sensitive to aggregation of higher trust values along the chains. While flexible, this uni-norm does not behave exclusively as a t-norm and causes trust amplification along a chain if operand values are greater than the neutral element due to aggregation via probabilistic sum. Similarly, if the operator is used to aggregate across chains operand values that are less than the neutral element then probabilistic product may result in non-montonic trust; that is, it is not a t-conorm and adding further delegation statements cause a reduction in trust.

QKeyNote provides a t-norm operator $\otimes_c^e$, as a safe trust-aggregating version of the uni-norm $\odot^e$ that provides a degree of compensation around neutral element e. This operator is a combination of probabilistic product when operand severity values are less than e, and min when operand severity values are greater than e and is defined in Table 1. A corresponding compensating trust t-conorm operator $\oplus_c^e$ is a combination of probabilistic sum when operand severity values are greater than e, or the max operator otherwise.

Following the micro-trust example above, suppose that Alice does not want her micro-trust statements to aggregate her trust of Bob beyond 0.5 degrees. This is achieved by requiring micro-trust to be aggregated using the compensating trust t-norm configured with a neutral element of 0.5:

```
Authorizer: Ka
Licensee: Kb
Conditions: (App_Domain=="PhotoShare"
&&  _TCONORM<="compTrust(0.5)"
&& Tag=="private" && Nonce = "123456"
&& Date >= "20100101" && Date<= "20101231"
&& _Action_Authorizers==Kb)  -> 0.01;
Signature: ....
```

5 Implementing $\boldsymbol{trust(X, Y)}$

There are many possible solutions for configurations of $flow$ and $trust$ that satisfy the the constraints given in Section 3. Two approaches to finding a solution are considered in this section.

Optimized Search. The problem can be expressed as a constraint satisfaction problem (CSP) [21, 23] that searches for a solution while maximizing the value of $trust(X, Y)$. This approach has been implemented using `choco` [27], a Java library for solving CSPs and supports optimization over real-valued variables. This implementation could be used in computing trust values in KeyNote delegation graphs. However, this search violates the principle of efficient trust evaluation [24] as it is computationally expensive and not suited to providing real-time answers involving search over large delegation graphs.

Heuristic Search. Rather than searching for values of $flow$ that yield an optimal $trust(X, Y)$, we use fixed heuristics that distribute $flow$ over outgoing arcs in the delegation graph. Given a node Z on a path between X and Y then the value of the *flow* from Z could be based on an even distribution of the value of $trust(X, Z)$ across the cardinality of the set of individual statements $Z \xrightarrow{z} Z'$ in the graph whereby Z' is connected to Y. For example in Figure 4, the distribution (under probabilistic sum) of $trust(A, D) = 0.36$ across outgoing flows $flow(A, D, E) = 0.2$ and$flow(A, D, C) = 0.2$, en-route to destination E.

This heuristic search approach is implemented using a topological sort of the delegation graph to direct the assignment and/or division of flows within the delegation graph while preserving the principle of conservation of trust. This strategy has been used in the implementation of the QKeyNote query engine discussed in Section 4.

When a trust query is made it is possible that the $flows$ selected during a previous search may contradict choices of $flow$ from earlier trust queries. For example, assuming an optimal search across the graph in Figure 3, then an earlier query $trust(A, D)$ returning 0.72 implies that a later query $trust(A, C)$ should return 0.0. Alternatively, if $trust(A, C)$ is queried first, then it could return the optimal trust value 0.64, with the subsequent query $trust(A, D)$ returning 0.0. Thus, an implementation of $trust(X, Y)$ may be *stateful*, in the sense that past trust queries may influence the results of future queries against the graph.

Stateful Trust Compliance Let partial function $trust_\Sigma(X, Y)$ record past trust queries by returning the result of the most recent trust value computed between X, Y. Given a delegation graph and $trust_\Sigma$, then a principal Y is trusted to at least degree x by principal X, if there exists a solution for this query such that $trust(X, Y) = x$ and for all principals Z, W then $trust_\Sigma(Z, W) \leq trust(Z, W)$.

Stateful trust compliance is useful for enforcing history-based security policies such as Chinese walls and dynamic separation style policies. For example, interpreting the policy in Figure 3, both D and C have the *potential* to interact with A as trusted principals. However, once A queries D's trust (presumably as a consequence of some interaction with D), then C's level of potential trust diminishes. The disadvantage of a stateful implementation of *trust* is that it requires a globally coordinated trust state, for example relying a centralized graph of delegation statements and $trust_\Sigma(X, Y)$ function. While a centralized approach is common in reputation trust systems, a goal of trust management is to support decentralized delegation statements in the form of cryptographic credentials. For the purposes of this paper we regard trust queries as stateless, determining the degree of trust regardless of past queries. Investigating the implementation of statefull queries is a topic for future research.

6 Discussion

This paper proposes a generalization of the KeyNote Trust Management system whereby KeyNote compliance values in the metric space [0..1] can be aggregated using arbitrary triangular norms and conorms. This provides a basis for Trust Management queries that return the degree to which a principal is trusted for some action. Note that the results in this paper are not limited to KeyNote, which was chosen for the sake of ease of exposition and implementation.

We argue that specifying how trust is aggregated, both along chains and across chains, is as important as specifying a degree of trust value and therefore should form part of the trust information provided in a credential. It is not the intention of this paper to prescribe any particular aggregation operator, rather we provide a framework in which different aggregation operators can be used in a consistent way. While a relatively intuitive distinction exists between the use of the fuzzy (max/min) operators and the probabilistic operators, exploring trust-based interpretations for the other Triangular norms is a topic for future research.

By encoding degrees of trust as compliance values and by using special attributes `_TNORM` and `_TCONORM` to specify optional aggregation operators within delegation credentials it is not necessary to change the KeyNote language. While the KeyNote query engine has been modified to implement the model described in Section 3 its behavior is consistent with the standard KeyNote implementation when the aggregation operators are min/max (default). Thus, backwards compatibility is provided. Furthermore, QKeyNote credentials can be used in a standard KeyNote query, at least to the extent that standard KeyNote effectively aggregates according to min/max, thereby providing a 'safe' interpretation under the aggregation operator type ordering. The current model assumes fixed aggregation operators for any given query (subject to the credential conditions). Investigating how a query can support different aggregation operators to be simultaneously used across a graph is a topic for future research.

The current model and implementation does not currently support threshold delegation in KeyNote. We are investigating how a form of threshold delegation can be emulated by using probabilistic sum. In this case a conventional KeyNote credential that delegates authority to the conjunction of two principals can be rewritten as two individual credentials, each delegating authority to one principal, with, for example degree of trust 0.5 and probabilistic sum as `_TCONORM`. A KeyNote decision is considered acceptable if the degree of trust is greater than the trust of either individual 0.5.

A wide range of literature exists on many different forms of reputation trust and trust metrics and the reader is referred to [18] and [24] for an in-depth review. Closest to the underlying trust model proposed in this paper are schemes that treat the problem as an inclusion/exclusion style calculation across a probabilistic network such as [15,22]. However, Section 2 demonstrates the problem of double-counting trust when performing such a calculation. Furthermore, [15,22] does not generalize to arbitrary t-norm/conorm aggregation operators and does not consider integration with a Trust Management system. Jøsang et. al. [17] avoid double-counting by removing weakest (trust) delegation statements from the graph until a series of delegation chains remain that can be safely aggregated, resulting in a sub-optimal trust calculation. In [16] an improved strategy is proposed that uses edge-splitting to transform a delegation-statement (shared across different chains) into separate statements, one for each path, with the trust of the original statement distributed across the separate statements. We conjecture that these strategies uphold the Principle of Preservation of Trust; investigating the use of these strategies in an alternative implementation of the QKeyNote interpreter is a topic for future research.

Bistarelli et. al. use multi-trust [2, 3] to model reputation trust within the RT Trust Management language. In the multi-trust model trust relationships are represented in terms of the degree of trust from a single trustor to a group of trustees; a c-semiring defines how the trust should be aggregated, using probabilistic product along chains and max for aggregating trust across separate chains. Adapting this model to the trust calculation scheme described in Section 3, and thereby supporting other t-norms and t-conorms, is a topic for future research.

Acknowledgments This research is supported by Science Foundation Ireland grant 08/SRC/11403. The authors would like to thanks the anonymous reviewers for their helpful feedback.

References

1. Becker, M., Fournet, C., Gordon, A.: Design and semantics of a decentralized authorization language. 20th IEEE Computer Security Foundations Symposium (Jan 2007)
2. Bistarelli, S., Martinelli, F., Santini, F.: A semantic foundation for trust management languages with weights: An application to the RT family. In: ATC. Lecture Notes in Computer Science, vol. 5060, pp. 481–495. Springer (2008)
3. Bistarelli, S., Santini, F.: Propagating multitrust within trust networks. In: SAC '08: Proceedings of the 2008 ACM symposium on Applied computing. pp. 1990–1994. ACM, New York, NY, USA (2008)
4. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.D.: The Keynote trust-management system, version 2 (September 1999)
5. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: Proceedings of the IEEE Symposium on Research in Security and Privacy. pp. 164–173. IEEE Computer Society Press, Oakland, CA (1996)
6. Buchanan, B., Shortliffe, E.: Ruled Based Expert Systems, The MYCIN Experiment of the Stanford Heuristic Programming Project. Addison-Wesley, Reading, MA (1984)
7. Colbourn, C.: The Combinatorics of Network Reliability. Oxford University Press (1987)
8. Dubois, D., Prade, H.: A review of fuzzy sets aggregation connectives. Information Sciences 36, 85–121 (1985)
9. Ellison, C., Frantz, B., Lampson, B., Rivest, R.L., Thomas, B., Ylonen, T.: SPKI certificate theory (September 1999)
10. Fagin, R.: Fuzzy queries in multimedia database systems. In: PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. pp. 1–10. ACM, New York, NY, USA (1998)
11. Foley, S.N.: Security risk management using internal controls. In: WISG '09: Proceedings of the first ACM workshop on Information security governance. pp. 59–64. ACM, New York, NY, USA (2009)
12. Foley, S.N: Using trust management to support transferable hash-based micropayments. In: Proceedings of the 7th International Financial Cryptography Conference. Gosier, Guadeloupe, FWI (January 2003)
13. Foley, S.N, Rooney, V.: Qualitative analysis for trust management: Towards a model of photograph sharing indiscretion. In: Seventeenth International Security Protocols Workshop. Springer-Verlag LNCS (post-proceedings forthcoming) (April 2009)
14. Gilbert, E., Karahalios, K.: Predicting tie strength with social media. Proceedings of the 27th international conference computer-human interaction (Jan 2009).
15. Haenni, R., Jonczy, J.: A new approach to PGP's web of trust. In: EEMA'07, European e-Identity Conference. Paris, France (2007)
16. Jøsang, A., Bhuiyan, T.: Optimal trust network analysis with subjective logic. In: SECURWARE. pp. 179–184 (2008)
17. Jøsang, A., Hayward, R., Pope, S.: Trust network analysis with subjective logic. In: ACSC. pp. 85–94 (2006)
18. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decis. Support Syst. 43(2), 618–644 (2007)
19. Klement, E.P., Mesiar, R., Pap, E.: On the relationship of associative compensatory operators to triangular norms and conorms. International Journal of Uncertainty, Fuzziness and Knowledge based Systems 4(2) (1996)

20. Li, J., Li, N., Winsborough, W.: Automated trust negotiation using cryptographic credentials. Proceedings of the 12th ACM conference on Computer and Communications Security (Jan 2005).
21. Mackworth, A.: Constraint satisfaction. In: Shapiro, S. (ed.) Encyclopedia of AI (second edition), pp. 285–293. John Wiley & Sons (1992)
22. Mahoney, G., Myrvold, W., Shoja, G.: Generic reliability trust model. PST 5, 3rd (2005)
23. Montanari, U.: Networks of constraints: Fundamental properties and applications to picture processing. Information Science 7, 95–132 (1974)
24. Reiter, M., Stubblebine, S.: Authentication metric analysis and design. ACM Trans. Inf. Syst. Secur. 2(2), 138–158 (1999)
25. Riegelsberger, J., Sasse, M., McCarthy, J.: The mechanics of trust: A framework for research and design. Int. J. Hum.-Comput. Stud. 62(3), 381–422 (2005)
26. Schweizer, B., Sklar, A.: Probabilistic metric spaces. North Holland, New York (1983)
27. Team Choco: choco: an open source java constraint programming library. In: Third International CSP Solver Competition (2008). Website choco.emn.fr
28. Zimmermann, H.J., Zysno, P.: Latent connectives in human decision making. Fuzzy Sets and Systems 4, 37–51 (1980)