Matrix powers algorithms for trust evaluation in PKI architectures

Jean-Guillaume Dumas, Hicham Hossayni

Matrix powers algorithms for trust evaluation in public-key infrastructures

Jean-Guillaume Dumas* Hicham Hossayni†

July 27, 2012

Abstract

This paper deals with the evaluation of trust in public-key infrastructures. Different trust models have been proposed to interconnect the various PKI components in order to propagate the trust between them. In this paper we provide a new polynomial algorithm using linear algebra to assess trust relationships in a network using different trust evaluation schemes. The advantages are twofold: first the use of matrix computations instead of graph algorithms provides an optimized computational solution; second, our algorithm can be used for generic graphs, even in the presence of cycles. Our algorithm is designed to evaluate the trust using all existing (finite) trust paths between entities as a preliminary to any exchanges between PKIs. This can give a precise evaluation of trust, and accelerate for instance cross-certificate validation.

1 Introduction

The principle of a Public Keys Infrastructure (PKI) is to establish (using certificates) a trust environment between network entities and thus guarantee some security of communications.

For instance companies can establish hierarchical PKI's with a certification authority (CA) signing all the certificates of their employees. In such a setting there is a full trust between employees and their CA. Now if entities with different PKI structures want to communicate, either they find a fully certified path between them or they don't. In the latter case some degree of trust has to be established between some disjoint entities.

Ellison and Schneier identified a risk of PKIs to be "Who do we trust, and for what?" which emphasizes the doubts about the trust relationship between the different PKI components [4]. Several incidents, including the one in which VeriSign issued to a fraudulent two certificates associated with Microsoft [8],

*Laboratoire J. Kuntzmann, Université de Grenoble. 51, rue des Mathématiques, umr CNRS 5224, bp 53X, F38041 Grenoble, France, Jean-Guillaume.Dumas@imag.fr.

†CEA / Léti, 17 rue des Martyrs, 38054 Grenoble, France, Hossayni.Hicham@gmail.com

or even the recent fraudulent certificates for Google emitted by DigiNotar [1], confirm the importance of the trust relationship in the trust models, see also [14]. This leads to the need of a precise and a global evaluation of trust in PKI architectures. Another approach would be to use some fully trusted keys or authorities, like the Sovereign Keys or the Convergence project[1], or e.g. trust lists [20].

For example in a cross-certification PKI, an entity called Alice can establish a communication with another entity called Bob only after validating Bob's certificate. For this, Alice must verify the existence of a certification path between her trust anchor and Bob's certification authority (CA). This certificate validation policy imposes that each entity must have a complete trust in their trust anchors, and that this trust anchor has a complete (direct or indirect) trust relationship with other CAs.

In e.g. [11, 12, 13, 6] algorithms are proposed to quantify the trust relationship between two entities in a network, using transitivity. Some of them evaluate trust throughout a single path, while others consider more than one path to give a better approximation of trust between entities. However to the best of our knowledge they are restricted to simple network *trees*. In this paper we choose the last approach and use transitivity to efficiently approximate global trust degree. Our idea is to use an adapted power of the adjacency matrix (used e.g. to verify the graph connectivity or to compute the number of *finite* paths between nodes). This spectral approach is similar to that used also e.g. for community detection in graphs [5] and we use it to produce a centralized or distributed quantification of trust in a network. Moreover it allows to deal with *any kind of graphs, without any restrictions to trees nor dags*.

More generally, the aim of this paper is to propose a generic solution for trust evaluation adapted to different trust models. The advantage of spectral analysis is twofold : first the use of matrix computations instead of graph algorithms provides an optimized computational solution in which the matrix memory management is more adapted to the memory hierarchy; second, our algorithm can be used for generic graphs, even in the presence of cycles. Moreover, our algorithm is iterative so that a good approximation of the global trust degrees can be quickly computed: this is done by fixing the maximum length of the trust paths that are considered. The complexity of this algorithm is $O(n^3 \cdot \varphi \cdot \ell)$ in the worst case, polynomial in n, the number of entities (nodes of the graph), φ, the number of trust relationships (edges), and ℓ, the size of the longest path between entities. For instance the algorithm proposed in [12] worked only for directed acyclic graphs (DAG) and required the approximate resolution of the Bounded Disjoint Paths problem, known to be NP-Hard [19]. In case of DAGs the complexity of our algorithm even reduces to $O(n \cdot \varphi \cdot \ell)$.

Our algorithm is designed to evaluate the trust using all existing (finite) trust paths between entities as a preliminary to any exchanges between PKIs. This can give a precise evaluation of trust, and optimize the certificate validation time. These computations can be made, in a centralized manner by a trusty

[1] https://www.eff.org/sovereign-keys, http://convergence.io

independent entity, like Wotsap for a web of trust networks[2], by CAs in the case of cross-certification PKI (e.g. via PKI Resource Query Protocols [18]), or by the users themselves in the case of PGP web of trust. The latter can even happen in a distributed manner [3] or with collaborations [17].

Our algorithm works for generic trust metrics but can be more efficient when the metrics form a ring so that block matrix algorithms can be used. We thus present in section 2 different possible trust metrics and their aggregation in trust network. We then present the transformation of DAG algorithms for the computation of the aggregation into matrix algorithms in section 3. We do this in the most generic setting, i.e. when even the dotproducts have to be modified, and with the most generic trust metric (i.e. including trust, uncertainty *and verified distrust*). Finally, in section 4, we present our new polynomial algorithm for generic graphs.

2 Transitive trust metrics

2.1 The calculus of trust

There are several schemes for evaluating the (transitive) trust in a network. Some present the trust degree as a single value representing the probability that the expected action will happen. The complementary probability being an uncertainty on the trust.

Others include the *distrust* degree indicating the probability that the opposite of the expected action will happen [11]. More complete schemes can be introduced to evaluate trust: Jøsang [15] for instance introduced the Subjective Logic notion which expresses subjective beliefs about the truth of propositions with degrees of "uncertainty".

[12, 13] also introduced a quite similar scheme with a formal, semantics based, calculus of trust and applied it to public key infrastructures (PKI). We chose to present this metric for its generality and include in this section some definitions and theorems taken from [12]. The idea is to represent trust by a triplet, (trust, distrust, uncertainty). Trust is the proportion of experiences proved, or believed, positive. Distrust is the proportion of experiences *proved negative*. Uncertainty is the proportion of experiences with unknown character.

Definition 1 ([12, §5.1])**.** *Let d be a trustor entity and e a trustee. Let m be the total number of encounters between d and e in a given context. Let n (resp. l) be the number of positive (resp. negative) experiences among all encounters between d and e.*

- **The trust degree** *is defined as the frequency rate of the trustor's positive experience among all encounters with the trustee. That is, $td(d,e) = \frac{n}{m}$.*
- **The distrust degree***: similarly we have $dtd(d,e) = \frac{l}{m}$.*

[2] http://www.lysator.liu.se/~jc/wotsap/index.html

- **The uncertainty**: *denoted by ud is defined by:* $ud(d,e) = 1 - td(d,e) - dtd(d,e)$.

In the following we will denote the **trust relationship** by a triple $tr(a,b) = \langle td(a,b),\ dtd(a,b),\ ud(a,b)\rangle$ or simply $tr(a,b) = \langle td(a,b), dtd(a,b)\rangle$ since the uncertainty is completely determined by the trust and distrust degrees.

In these definitions, the trust depends on the kind of expectancy, the context of the experiences, type of trust (trust in belief, trust in performance), ..., see e.g. [13]. For simplicity, we only consider in the next sections the above generic concept of trust.

2.2 Aggregation of trust

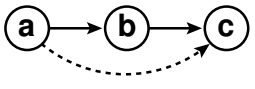
The main property we would like to express is *transitivity*. Indeed in that case keys trusted by many entities, themselves highly trusted, will induce a larger confidence. In the following we will consider a trust graph representing the trust relationships as triplets between entities in a network.

Definition 2 (Trust graph)**.** *Let $\mathcal{T} \subset [0,1]^3$ be a set of trust relationships. Let V be a set of entities of a trust network. Let E be a set of directed edges with weight in $\mathcal{T}$. Then $G = (V, E, \mathcal{T})$ is called a* trust graph *and there is an edge between two vertices whenever there exist a nonzero trust relationship between its entities.*

Next we define the transitivity over a path between entities and using parallel path between them as sequential and parallel aggregations. We first need to define a trust path:

Definition 3 (Trust path)**.** *Let $G = (V, E, \mathcal{T})$ be a trust graph. A trust path between two entities $A_1 \in V$ and $A_n \in V$ is defined as the chain, $A_1 \xrightarrow{t_1} A_2 \xrightarrow{t_2} ...A_{n-1} \xrightarrow{t_{n-1}} A_n$, where A_i are entities in V and $t_i \in \mathcal{T}$ are respectively the trust degrees associated to each trust relation $(A_i \xrightarrow{t_i} A_{i+1}) \in E$.*

The need of the sequential aggregation is shown by the following example. Consider, as shown on figure 1, Alice trusting Bob with a certain degree, and Bob trusting Charlie with a certain trust degree. Now, if Alice wishes to communicate with Charlie, how can she evaluate her trust degree toward him? For this, we use the sequential aggregation of trust to help Alice to make a decision, and that is based on Bob's opinion about Charlie.



$\longrightarrow$: Direct trust relationship

$\dashrightarrow$: Indirect (sequentially aggregated) trust relationship

Figure 1: Simple sequential trust aggregation

Definition 4 (Sequential aggregation of trust [12, Theorem UT-1]). *Let $G = (V, E, \mathcal{T})$ be a trust graph. Let a,b and c be three entities in V and $tr(a, b) \in \mathcal{T}$, $tr(b, c) \in \mathcal{T}$ be respectively the trust degrees associated to the entity pairs (a, b) and (b, c). The* sequential aggregation of trust *between a and c is a function f, that calculates the trust degree over the trust path $a \to b \to c$. It is defined by :*

$$f : \mathcal{T} \times \mathcal{T} \to \mathcal{T} \text{ with} \quad f(tr(a,b), tr(b,c)) = tr_f(a,c) = \langle td_f(a,c), dtd_f(a,c)\rangle$$
$$\text{where} \quad td_f(a,c) = td(a,b).td(b,c) + dtd(a,b).dtd(b,c)$$
$$dtd_f(a,c) = dtd(a,b).td(b,c) + td(a,b).dtd(b,c)$$

Lemma 1. *Definition 4 is sound.*

Proof. We consider the trust path $a \xrightarrow{x} b \xrightarrow{y} c$ and let $x = \langle x_t, x_d, x_u\rangle$, $y = \langle y_t, y_d, y_u\rangle$ and $z = \langle z_t, z_d, z_u\rangle = f(x, y)$. Then $z_u = 1 - x_t y_t - x_d y_d - x_t y_d - x_d y_t = 1 - x_t(y_t + y_d) - x_d(y_t + y_d) = 1 - (x_t + x_d)(y_t + y_d) = 1 - (1 - x_u)(1 - y_u)$. Since $0 \leq x_u \leq 1$ and $0 < leqy_u \leq 1$, we have that $0 \leq (1 - x_u)(1 - y_u) \leq 1$ so that $0 \leq z_u = x_u + y_u \leq 1$. Since x_u and y_u are positive by definition, it follows that $f(x, y)$ *is* a trust relationship. □

From the definition we also for instance immediately get the following properties.

Lemma 2. *f (Sequential aggregation) increases uncertainty.*

Proof. As in the proof of lemma 1, we let $x = \langle x_t, x_d, x_u\rangle$, $y = \langle y_t, y_d, y_u\rangle$ and $z = f(x, y)$. From $z_u = 1 - (1 - x_u)(1 - y_u)$ we have that $z_u = x_u + y_u(1 - x_u)$. Therefore as y_u and $1 - x_u$ are positive we in turn have that $z_u \geq x_u$. We also have $z_u = y_u + x_u(1 - y_u)$ and therefore also $z_u \geq y_u$. Moreover we see that if both uncertainties are non zero, then the uncertainty of $f(x, y)$ is strictly increasing. □

Lemma 3 ([13, Property 2]). *f (Sequential aggregation) is associative*

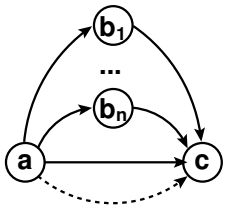
Proof. We consider the trust path $a \xrightarrow{x} b \xrightarrow{y} c \xrightarrow{z} d$ and let $x = \langle x_t, x_d\rangle$, $y = \langle y_t, y_d\rangle$, $z = \langle z_t, z_d\rangle$. Then $f(x, y) = \langle x_t y_t + x_d y_d, x_d y_t + x_t y_d\rangle$ and $f(y, z) = \langle y_t z_t + y_d z_d, y_d z_t + y_t z_d\rangle$, so that $f(f(x, y), z) = \langle (x_t y_t + x_d y_d)z_t + (x_d y_t + x_t y_d)z_d, (x_d y_t + x_t y_d)z_t + (x_t y_t + x_d y_d)z_d\rangle$. In other words, $f(f(x, y), z) = \langle x_t(y_t z_t + y_d z_d) + x_d(y_d z_t + y_t z_d), x_d(y_t z_t + y_d z_d) + x_t(y_d z_t + y_d z_d)\rangle = f(x, f(y, z))$. □

From this associativity, this sequential aggregation function can be applied recursively to any tuple of values of $\mathcal{T}$, to evaluate the sequential aggregation of trust over any trust path with any length ≥ 2, for instance as follows: $f(t_1, ..., t_n) = f(f(t_1, ..., t_{n-1}), t_n)$.

Now, the following definition of the parallel aggregation function can also be found in [13, § 7.2.2], it is clearly associative and is illustrated on figure 2.

Definition 5 (Parallel aggregation of trust [12, §6.2])**.** *Let $G = (V, E, \mathcal{T})$ be a trust graph. Let $a, b_1, \ldots, b_n, c$ be entities in V and $tr_i(a, c) \in \mathcal{T}$ be the trust degree over the trust path $a \to b_i \to c$ for all $i \in 1..n$. The parallel aggregation of trust is a function g, that calculates the trust degree associated to a set of disjoint trust paths connecting the entity a to the entity c. It is defined by:*

$$g : \mathcal{T}^n \to \mathcal{T} \text{ with } g([tr_1, tr_2, \ldots, tr_n](a, c)) = tr_g(a, c) = \langle td_g(a, c), dtd_g(a, c) \rangle$$
$$\text{where } \; td_g(a, c) = 1 - \prod_{i=1..n} (1 - td_i) \text{ and } dtd_g(a, c) = \prod_{i=1..n} dtd_i$$



⟶ : Direct trust relationship

⇢ : Indirect (parallely aggregated) trust relationship

Figure 2: Parallel aggregation of trust for multiple trust

Lemma 4. *Definition 5 is sound.*

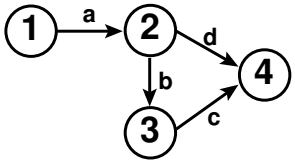
Proof. We start by proving it for two paths of length 1. Let $x = \langle x_t, x_d \rangle$ and $y = \langle y_t, y_d \rangle$. Then $g(x, y) = \langle 1 - (1 - x_t)(1 - y_t), x_d y_d \rangle$. It is clear that td_g and dtd_g are non negative. Now from the definition of trust relationship we know that $x_t + x_d \leq 1$ and $y_t + y_d \leq 1$ so that $x_d \leq 1 - x_t$ and $y_d \leq y_t - 1$. Therefore $x_d y_d \leq (1 - x_t)(1 - y_t)$ and $td_g(x, y) + dtd_g(x, y) \leq 1$. This generalizes smoothly to any number of paths by induction. □

Definition 6 (Trust evaluation)**.** *Let $G(V, E, \mathcal{T})$ be a directed acyclic trust graph, and let a and b be two nodes in V. The trust evaluation between a and b is the trust aggregation over* all *paths connecting a to b. It is computed recursively by aggregating (evaluating) the trust between the entity a and the predecessors of b (except, potentially, a). Denote by $Pred(b)$ the predecessors of b and by p_i the elements of $Pred(b) \setminus \{a\}$. The* trust evaluation *between a and b consists in first recursively evaluating the trust over all paths $a \to \ldots \to p_i$, then applying the sequential aggregation over the paths $a \to p_i \to b$ and finally the parallel aggregation to the results (and $(a \to b)$, if $(a \to b) \in E$).*

Remark 1. *Note that since the predecessors of b are distinct, after the sequential aggregations all the resulting edges from a to b are distinct. They are thus disjoint paths between a and b and parallel aggregation applies.*

Remark 2. *In the above definition of trust evaluation we favor the evaluation from right to left. As shown on the example below this gives in some sense prominence to nodes close to the beginning of the path, that is nodes closer to the one asking for an evaluation. This is illustrated on figure 3 where to different strategies for the evaluation of trust are shown: on the left, one with parallel, then sequential, aggregation; the other one with sequential, then parallel,*

From left to right, we get $f(a, g(f(b, c), d))$



From right to left (definition 6), we get $g(f(a, b, c), f(a, d))$

Figure 3: Two strategies for trust evaluation between node 1 and 4.

aggregation. Would f be distributive over g we would get the same evaluation. As we will see in section 4, this choice of evaluation can has an important impact in the presence of cycles.

The graph theoretic method proposed by [12, §6.3] for evaluating trust between two nodes in a DAG requires the approximate solution of the Bounded Disjoint Paths problem, known to be NP-Hard [19]. This algorithm has two steps: first an elimination of cycles via BDP, then a composition of sequential and then parallel aggregation. We will show in the next section that our matrix algorithm produces the same output on acyclic graphs. Moreover, in section 4, we will present a variant of this algorithm, still with polynomial time bound, that directly deals with generic graphs.

By storing the already evaluated relationships, the aggregation part of [12, §6.3] can for instance compute the global trust in a graph with 1000 vertices and 250 000 edges in about 1 minute on a standard laptop. In the following, we propose to rewrite this algorithm in terms of linear algebra [16]. Using sparse linear algebra the overall complexity will not change, and the analysis will be eased. Now if the graph is close to complete, the trust matrix will be close to dense and cache aware linear algebra block algorithms will be more suited.

In both cases, the linear algorithm will decompose the evaluation into converging iterations which could be stopped before exact convergence in order to get a good approximation faster.

Furthermore, using this linear algebra point of view, we will we able to generalize the algorithm to any directed graph.

3 Matrix Powers Algorithm For Directed Acyclic Graphs

In the previous section, we presented the trust propagation scheme introduced by [12], which consists of using the parallel and sequential trust aggregations for evaluating the trust between a network's entities. Indeed, our matrix powers algorithm can be implemented with different trust propagation schemes under one necessary condition: the transitivity property of the (sequential and parallel) trust propagation formulas. In this section, we propose a new algorithm for evaluating trust in a network using the powers of the matrix of trust. This algorithm uses techniques from graph connectivity and communicability in networks [5].

3.1 Matrix and monoids of trust

Definition 7. *Let $G = (V, E, \mathcal{T})$ be a trust graph, the* matrix of trust *of G, denoted by C, is the adjacency matrix containing, for each node of the graph ,G the trust degrees of a node toward its neighbors, $C_{ij} = \langle td(i,j), dtd(i,j), ud(i,j)\rangle$. When there is no edge between i and j, we choose $C_{ij} = \langle 0,0,1\rangle$ and, since every entity is fully confident in itself, we also choose for all i: $C_{ii} = \langle 1,0,0\rangle$.*

Definition 8. *Let $\mathcal{T}$ be the set $\mathcal{T} = \{\langle x,y,z\rangle \in [0,1]^3, x+y+z=1\}$, equipped with two operations "+" and "." such that $\forall(\langle a,b,u\rangle, \langle c,d,v\rangle) \in \mathcal{T}^2$ we have: $\langle a,b,u\rangle.\langle c,d,v\rangle = \langle ac+bd, ad+bc, 1-ac-ad-bd-bc\rangle$, and $\langle a,b,u\rangle + \langle c,d,v\rangle = \langle a+c-ac, bd, (1-a)(1-c)-bd\rangle$. We define as the* monoids of trust *the monoids $(\mathcal{T}, +, \langle 0,1,0\rangle)$ and $(\mathcal{T}, ., \langle 1,0,0\rangle)$.*

$\langle 0,0,1\rangle$ is the absorbing element of "." in $\mathcal{T}$. This justifies a posteriori our choice of representation for the absence of an edge between two nodes in definition 7.

We can also see that the set $\mathcal{T}$ corresponds to trust degrees $\langle td, dtd, ud\rangle$. In addition, the operations "." and "+" represent respectively the sequential and parallel aggregations of trust, denoted f and g in definitions 4 and 5.

Remark 3. *Note that "." is* not *distributive over "+". This fact can prevent the use of block algorithms and fast matrix methods. Now if the simpler metric without distrust is used (i.e. distrust is fixed to zero for every entity in the graph, and* will remain zero *all along our trust algorithms), then "." becomes distributive over "+". Thus in the following section we will present timings with or without taking distrust into consideration.*

3.2 d-aggregation of trust

Definition 9 (d-aggregation of trust)**.** *For $d \in \mathbb{N}$, the* d-aggregation of trust *between two nodes A and B, in an acyclic trust graph, is the trust evaluation over* all paths of length at most *d, connecting A to B. It is denoted $d\text{-}agg_{A,B}$.*

Definition 10 (Trust vectors product)**.** *Consider the directed trust graph $G = (V, E, \mathcal{T})$ with trust matrix C. Let $\overrightarrow{C_{i*}}$ be the i-th row vector and $\overrightarrow{C_{*j}}$ be the j-th column vector. We define the product of $\overrightarrow{C_{i*}}$ by $\overrightarrow{C_{*j}}$ by:*

$$\overrightarrow{C_{i*}}.\overrightarrow{C_{*j}} = \sum_{k\in V}^{k\neq j} C_{ik}.C_{kj}$$

Note that $C_{ii} = C_{jj} = \langle 1,0,0\rangle$ is the neutral element for "."*.* Therefore, our definition differs from the classical dot product as we have removed one of the $C_{ij} = C_{ii} \cdot C_{ij} = C_{ij} \cdot C_{jj}$, but then it matches the 2-aggregation:

Lemma 5. *The product $\overrightarrow{C_{i*}}.\overrightarrow{C_{*j}}$ is the 2-aggregated trust between i and j.*

Proof. We prove first that $C_{ik}.C_{kj}$ is the sequential aggregation of trust between i and j throughout all the paths (of length≤ 2) $i \to k \to j$ with $k \in V$. Let k be an entity in the network. There are two cases: whether k is one of the boundaries of the path or not. The first case is: $k = i$ or $k = j$:

- if $k = i$, then from the trust matrix definition 7, $C_{ii} = \langle 1, 0, 0\rangle \forall i$, thus we have $C_{ik}.C_{kj} = C_{ii}.C_{ij} = \langle 1, 0, 0\rangle.C_{ij} = C_{ij}$.
- if $k = j$, then similarly $C_{ik}.C_{kj} = C_{ij}.C_{jj} = C_{ij}.\langle 1, 0, 0\rangle = C_{ij}$

Therefore $C_{ik}.C_{kj}$ corresponds to the [sequential aggregation of] trust between i and j throughout the path (i, j) of length 1. This is why in the product, we added the constraint $k \neq j$ in the sum to avoid taking C_{ij} twice into account. Now the second case is: $k \neq i$ and $k \neq j$:

- if k belongs to a path of length 2 connecting i to j, then: i trusts k with degree $C_{ik} \neq \langle 0, 0, 1\rangle$, and k trusts j with degree $C_{kj} \neq \langle 0, 0, 1\rangle$. From definition 4, $C_{ik}.C_{kj}$ corresponds to the sequential aggregation of trust between i and i throughout the path $i \to k \to j$.
- If there is no path of length 2 between i and j containing k, then we have $C_{ik} = \langle 0, 0, 1\rangle$ or $C_{kj} = \langle 0, 0, 1\rangle$, and thus $C_{ik}.C_{kj} = \langle 0, 0, 1\rangle$ is also the aggregation of trust between i and j on the path traversing the node k.

Finally, we can deduce that $\overrightarrow{C_{i*}}.\overrightarrow{C_{*j}} = \sum_{k\in V}^{k\neq j} C_{ik}.C_{kj}$ corresponds to the parallel aggregation of trust between i and j using all paths of length ≤ 2, which is the 2-aggregated trust between i and j. Note that the latter is equivalent to $\overrightarrow{C_{i*}}.\overrightarrow{C_{*j}} = \sum_{k\in Pred.(j)\setminus\{i\}} C_{ik}.C_{kj} + C_{ij}$. □

Definition 11 (Trust matrix product)**.** *Let $C_{(ij)}$ and $M_{(ij)}$ be two trust matrices. We define the matrix product $N = C * M$ by: $\forall i, j \in \{1..n\}$*

$$N_{ij} = \begin{cases} \overrightarrow{C_{i*}}.\overrightarrow{M_{*j}} = \sum_{k\in V}^{k\neq j} C_{ik}.M_{kj} & \text{if } i \neq j \\ \langle 1, 0, 0\rangle & \text{otherwise} \end{cases}$$

Lemma 6. *Let (C_{ij}) be the trust matrix of a network of entities, whose elements belong to a trust graph G. The matrix M defined by: $M = C^2 = C*C$ represents the 2-aggregated trust between all distinct entity pairs and the total trust of entities for themselves.*

Proof. From definition 11, we have: $M_{ij} = \begin{cases} \overrightarrow{C_{i*}}.\overrightarrow{C_{*j}} = \sum_{k\in V}^{k\neq j} C_{ik}.C_{kj} & \text{if } i \neq j \\ \langle 1, 0, 0\rangle & \text{otherwise} \end{cases}$.
Thus, if $i = j$, then $M_{ii} = \langle 1, 0, 0\rangle$ as claimed. Otherwise, $i \neq j$ and according to lemma 5, M_{ij} the 2-aggregated trust between i and j. □

Now, according to definition 6 of the trust evaluation in a network, we can generalize lemma 6 to evaluate trust using all paths of a given length:

Theorem 1. *Let $G = (V, E, \mathcal{T})$ be an acyclic trust graph with matrix of trust C. Then C^d represents the d-aggregated trust between all entity pairs in V.*

Proof. We proceed by induction. Let $HR(d)$ be the hypothesis that C^d represents the d-aggregated trust between all entity pairs in V. Then $HR(2)$ is true from lemma 6. Now let us suppose that $HR(d)$ is true. First if $i = j$, then $(d+1)\text{-agg}_{i,i} = \langle 1, 0, 0\rangle = C^{d+1}_{i,i}$ from definition 11. Second, definition 6, gives:

$$\begin{aligned}
(d+1)\text{-agg}_{i,j} &= \sum_{k \in Pred.(j)\setminus\{i\}} d\text{-agg}_{ik}.C_{kj} + C_{ij} && \text{by def. 6}\\
&= \sum_{k \in Pred.(j)\setminus\{i\}} C^d_{ik}.C_{kj} + C_{ij} && \text{by } HR(d)\\
&= \sum_{k \in Pred.(j)} C^d_{ik}.C_{kj} && \text{since } C_{ii} = \langle 1, 0, 0\rangle\\
&= \sum_{k \in V}^{k \neq j} C^d_{ik}.C_{kj} && \text{if } (ik) \notin E,\ C_{ik} = \langle 0, 0, 1\rangle
\end{aligned}$$

Overall, $HR(d+1)$ is proven and induction proves the theorem. □

From this theorem, we immediately have that in an acyclic graph the matrix powers must converge to a fixed point.

Corollary 1. *Let $G = (V, E, \mathcal{T})$ be an acyclic trust graph with trust matrix C. The successive application of the matrix powers of C converges to a matrix C^ℓ, where ℓ is the size of the longest path in G. Which means that C^ℓ is a fixed point for the matrix powers:*

$$\lim_{n \to \infty} C^n = C^l$$

For the proof of this corollary, we need the following lemma.

Lemma 7. *Let $\lambda_{i,j}$ be the length of the largest path between i and j. Then either $\lambda_{i,j} = 1$ or $\lambda_{i,j} = max_{k \in Pred.(j)\setminus\{i\}}(\lambda_{i,k}) + 1$.*

Proof of corollary 1. We prove the result by induction and consider the hypothesis $HR(d)$ with $d \geq 1$:

$$\forall i, j \in V, \text{ such that } \lambda_{i,j} \leq d \text{ and } \forall t \geq \lambda_{ij}, \text{ then } C^t_{ij} = C^{\lambda_{i,j}}_{ij}$$

We first prove the hypothesis for $d = 1$: Let i and j be such that the longest path between them is of length 1. This means that in this acyclic directed graph, there is only one path between i and j, the edge $i \to j$. Now from definition 6, we have that $\forall t, C^t_{ij} = \sum_{k \in Pred.(j)\setminus\{i\}} C^{t-1}_{ik}.C_{kj} + C_{ij}$. However, $Pred.(j) \setminus \{i\} = \emptyset$ so that $C^t_{ij} = C_{ij}$, for all t. This proves $HR(1)$.

Now suppose that $HR(d)$ is true. Let i and j be two vertices in $G(V, E, \mathcal{T})$. We have two cases. First case: $\lambda_{ij} \leq d$. Then $\lambda_{i,j} \leq d+1$ and from the induction hypothesis, we have that $C_{ij}^t = C_{ij}^{\lambda_{ij}} \forall t \geq \lambda_{ij}$. Therefore $HR(d+1)$ is true for i and j. Second case: $\lambda_{ij} = d+1 \geq 2$. Then we have $\forall u \geq 0$ $C_{ij}^{d+1+u} = \sum_{k \in Pred.(j)} C_{ik}^{d+u}.C_{kj}$. Now, from lemma 7, the maximum length of any path between i and a predecessor of j is $\lambda_{i,j} - 1 = d$. Therefore, from the induction hypothesis, we have that $C_{ik}^{d+u} = C_{ik}^{\lambda_{ik}} = C_{ik}^d$ for all $k \in Pred.(j)$. Then $C_{ij}^{d+1+u} = \sum_{k \in Pred.(j)} C_{ik}^d.C_{kj} = C_{ij}^{d+1}$ which proves the induction and thus the corollary. □

From the latter corollary, we now have an algorithm to compute the trust evaluation between all the nodes in an acyclic trust network: perform the trust matrix powering with the monoids laws up to the longest path in the graph.

Theorem 2. *Let (C_{ij}) be the trust matrix corresponding to an acyclic graph with n vertices and φ edges whose longest path is of size ℓ. The complexity of the evaluation of the aggregated trust between all entity pairs represented by this trust matrix is bounded by $O(n \cdot \varphi \cdot \ell)$ operations.*

Proof. C is sparse with φ non zero element. Thus multiplying C by a vector requires $O(\varphi)$ operations and computing $C \times C^i$ requires $O(n\varphi)$ operations. Then, theorem 1 shows that C^j for $j \geq \ell$ is the j-aggregated trust between any entity pair. Finally, corollary 1 shows that $C^j = C^\ell$ as soon as $j \geq \ell$. □

The implementation of this algorithm took less than 1 second to perform an iteration (C^2) on the graph of section 2 with 1000 vertices and $250K$ edges. And it needed less than 6 seconds to return the final trust degrees.

4 Evaluation of trust in the presence of cycles

The algorithm induced by theorem 1 works only for directed acyclic graphs. Its advantage is thus restricted to the case when the distrust is *not* taken into consideration: then block or sparse algorithms can provide the BLAS[3] linear algebra performance to trust evaluation.

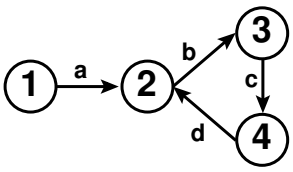
Now, in the presence of cycles in a network, the matrix powers algorithm will add the contribution of each edge of a cycle indefinitely.

Consider the graph of figure 4, with a, b, c, d the trust degrees corresponding to the links $1 \xrightarrow{a} 2 \xrightarrow{b} 3 \xrightarrow{c} 4 \xrightarrow{d} 2$. Its trust matrix C and applications of the matrix powers algorithm on this matrix are shown on figure 4, right.

For instance, the value:

$$\begin{aligned} C_{1,3}^5 &= 0 + C_{1,2}^4 C_{2,3} + 0 = (C_{1,2} + C_{1,4}^3 C_{4,2}) C_{2,3} \\ &= (C_{1,2} + (C_{1,3}^2 C_{3,4}) C_{4,2}) C_{2,3} = (C_{1,2} + (C_{1,2} C_{2,3}) C_{3,4}) C_{4,2}) C_{2,3} \\ &= (a + a.b.c.d).b, \end{aligned}$$

[3] e.g. ATLAS [23], GotoBLAS [9], MUMPS `http://graal.ens-lyon.fr/MUMPS` etc.



1	a	0	0
0	1	b	0
0	0	1	c
0	d	0	1

1	a	a.b	0
0	1	b	b.c
0	c.d	1	c
0	d	d.b	1

1	a	a.b	a.b.c
0	1	b	b.c
0	c.d	1	c
0	d	d.b	1

1	a+ a.b.c.d	a.b	a.b.c
0	1	b	b.c
0	c.d	1	c
0	d	d.b	1

1	a + a.b.c.d	(a+a.b.c.d).b	a.b.c
0	1	b	b.c
0	c.d	1	c
0	d	d.b	1

Figure 4: Graph with one cycle and its trust matrix C with C^2, C^3, C^4 and C^5.

corresponds to the aggregation on the paths $1 \to 2 \to 3$ and $1 \to 2 \to 3 \to 4 \to 2 \to 3$ linking 1 to 3. If we continue iterations for $n > 5$, we find that the algorithm re-evaluates infinitely the trust on the loop $3 \to 4 \to 2 \to 3$ to yield $C_{1,3}^{2+3k} = (a + C_{1,3}^{2+3k-1}.b.c.d).b$ with most probably an increase in uncertainty (e.g. from the many sequential aggregations and lemma 2).

4.1 Convergent iteration

To solve this issue, we propose to change the matrix multiplication procedure, so that *each edge will be used only once in the assessment of a trust relationship.* For this, we use a memory matrix R_{ij}. This stores, for each pair of nodes, all edges traversed to evaluate their trust degree. Only the paths containing an edge not already traversed to evaluate the trust degree are taken into account at the following iteration. Therefore, the computation of C_{ij}^{ℓ} for $n \geq 1$, becomes that given in algorithm 1.

By doing this modification of the matrix multiplication we will obtain a different trust evaluation. We have no guarantee that this new evaluation is somewhat more accurate but intuitively as in any path an edge is considered only once, cycles will not have a preponderate effect. Furthermore, we recover an interesting fixed point.

Theorem 3. *Let C be the trust matrix corresponding to a generic trust graph. Algorithm 1 converges to the matrix C^{ℓ} where ℓ is the longest* acyclic path *between vertices.*

Proof. Let C^{ℓ} be the evaluation of the ℓ-aggregated trust between all entity pairs after ℓ iterations where ℓ is the longest acyclic path between vertices. At

Algorithm 1 Matrix powers for generic network graphs

Input An $n \times n$ matrix of trust C of a generic directed trust graph.
Output Global trust in the network.

$\ell = 2$;
Repeat
  For all $(i, j) \in [1..n]^2$ with $i \neq j$ **do**
    $C_{ij}^{\ell} = \langle 0, 0, 1 \rangle$;
    $R_{ij}^{\ell} = \emptyset$;
    For $k = 1$ **to** n **do**
      $t = C_{ik}^{\ell-1}.C_{kj}$;
      If $(t \neq \langle 0, 0, 1 \rangle)$ **then**
        $C_{ij}^{\ell} = C_{ij}^{\ell} + t$;
        $R_{ij}^{\ell} = R_{ij}^{\ell} \bigcup R_{ik}^{\ell-1} \bigcup (k \to j)$; // using a sorted list union
      End If
    End For
    If $\left(\#R_{ij}^{\ell} \subset \#R_{ij}^{\ell-1}\right)$ **then** $C_{ij}^{\ell} = C_{ij}^{\ell-1}$;$R_{ij}^{\ell} = R_{ij}^{\ell-1}$; **End If**
  End For
Until $C^{\ell} == C^{\ell-1}$; $++\ell$;
return C^{ℓ};

this stage, for each pair i, j, all the edges belonging to a path between i and j will be marked in R_{ij}^{ℓ}. Therefore, no new $t = C_{ik}^{\ell}.C_{kj}$ will be added to $C_{ij}^{\ell+1}$. Conversely, at iteration $x < \ell$, if there exist an acyclic path between a pair i, j of length greater than x, then it means that there exists at least one edge e not yet considered on a sub-path from, say, u to v, of length x: $i \dashrightarrow u \dashrightarrow \overset{e}{\rightarrow} \dashrightarrow v \dashrightarrow j$. Then R_{uv}^{x} will be different from R_{uv}^{x-1} and so will be C_{uv}^{x} from C_{uv}^{x-1}. □

Theorem 4. *Let C be the trust matrix corresponding to a generic trust graph with n vertices and φ edges whose longest path between vertices is of size ℓ. The complexity of the global evaluation of all the paths between any entity is bounded by $O(n^3 \cdot \varphi \cdot \ell)$ operations.*

Proof. Using algorithm 1, we see that the triple loop induces n^3 monoid operations and n^3 merge of the sorted lists of edges. A merge of sorted lists is linear in the number of edges, φ. Then the overall iteration is performed at most ℓ times from theorem 3. □

By applying the new algorithm on the example of figure 4, we still obtain $C_{1,3}^{5} = (a + a.b.c.d).b$, but now $R_{1,3}^{5} = \{a, b, c, d\}$ and thus no more contribution can be added to $C_{1,3}^{5+i}$.

A first naive dense implementation of this algorithm took about 1.3 seconds to perform the first iteration (C^2) on the graph of section 2 with 1000 vertices and $250K$ edges. And it needed only 7 iterations to return the final trust degrees with high precision.

Remark 4. *More generally, the convergence will naturally be linked to the ratio of the dominant and subdominant eigenvalues of the matrix. For instance on some random matrices it can be shown that this ratio is* $O(\sqrt{n})$ *[7, Remark 2.19], and this is what we had experimentally. Of course, for, e.g., PKI trust graphs, more studies on the structure of the typical network would have to be conducted.*

4.2 Bounded evaluation of trust

In practice, the evaluation of trust between two nodes A and B need not consider all trust paths connecting A to B for two reasons:

- First, the mitigation is one of the trust properties, i.e. the trust throughout trust paths decreases with the length of the latter. Therefore after a certain length L, the trust on paths becomes weak and thus should have a low contribution in improving the trust degree after their parallel aggregation.
- Second, if at some iteration $n \geq 1$, we already obtained a high trust degree, then contributions of other paths will only be minor.

Therefore, it is possible to use the matrix powers algorithm with less iterations and e.g. a threshold for the trust degree, in order to rapidly compute a good approximation of the trust in a network. To determine the optimal threshold, we have conducted hundreds of comparisons between results of each iterations of our algorithm and the final trust degrees computed with Algorithm 1. We found that on average on $1K$-vertices random matrices, we needed 6 iterations to get an approximation of the trust degrees at 0.01, and only 7 iterations[4] (in 97% of the cases) to achieve an error rate less than 10^{-6}.

5 Conclusion and remarks

The actual public-key infrastructure models assume that the relationships between the PKI entities are based on an absolute trust. However, several risks are related to these assumptions when using PKI procedures.

In this paper, we have reduced the evaluation of trust between entities of a DAG network to linear algebra. This gives a polynomial algorithm to asses the global trust evaluation in a network. Moreover, depending on the sparsity of the considered graphs, this enables to use adapted linear algebra methods. Also the linear algebra algorithm decomposes the evaluation into converging iterations. These iterations can be terminated earlier than convergence in order to get a good approximation faster. Finally this enabled us also to generalize the trust evaluation to any directed graph, i.e. not necessarily acyclic, still with a polynomial complexity.

[4] From remark 4, we see that would the product be a classical matrix product, $\sqrt{n}$ being about 32, to get an approximation at 10^{-6} we would have needed on the order of $6/log_{10}32 \approx 3.98 \leq 4$ iterations to converge.

Further work include the determination of the optimal number of iterations necessary to get a good approximation of trust in minimal time. Small-world theory for instance could be of help [22, 2].

Also, there is a restriction in the current trust models, which imposes that Alice cannot communicate with Bob if there is no trust path between them. However, this limitation can be overcome by using the reputation notion jointly with the trust notion. If Alice can be sure that Bob has a good reputation in its friends circle and vice versa, then they can extend their trusty network and communicate safely [21].

Finally, the choice of the implementation model is a crucial subject. One approach is to adopt a centralized model, where all computation are done by a unique trusty entity. Then the reliability of the system depends entirely on the reliability of the trusted entity. This would typically be achieved by CAs. Another approach is a distributed model, where the entities must contact each others to share some trust degrees. This will enable each entity to evaluate the (at least *local*) trust in its neighborhood. On the one hand, this can be applied to large networks while preserving for each entity a low computational cost. On the other hand, each entity might have only a limited view of the whole network. Therefore, a dedicated Network Discovery Mechanism (NDM) is needed to expand the entities trust sub-network. This NDM can be crucial to determine the trust model safety [10]. Besides, the trust degrees could be a sensitive information. Therefore, the join use of trust matrices and homomorphic cryptosystems enabling a private computation of shared secret could be useful.

References

[1] H. Adkins. An update on attempted man-in-the-middle attacks. Technical report, Google Online Security Blog, Aug. 2011. http://googleonlinesecurity.blogspot.com/2011/08/update-on-attempted-man-in-middle.html.

[2] R. Albert, H. Jeong, and A.-L. Barabasi. Diameter of the world-wide web. *Nature*, 401:130–131, Sept. 1999.

[3] S. Dolev, N. Gilboa, and M. Kopeetsky. Computing multi-party trust privately: in $O(n)$ time units sending one (possibly large) message at a time. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1460–1465, New York, NY, USA, 2010. ACM.

[4] C. Ellison and B. Schneier. Ten risks of PKI: What you're not being told about Public Key Infrastructure. *Computer Security Journal*, 16(1):1–7, 2000. http://www.counterpane.com/pki-risks.pdf.

[5] E. Estrada and N. Hatano. Communicability graph and community structures in complex networks. *Applied Mathematics and Computation*, 214(2):500–511, 2009.

[6] S. N. Foley, W. M. Adams, and B. O'Sullivan. Aggregating trust using triangular norms in the keynote trust management system. In J. Cuéllar, J. Lopez, G. Barthe, and A. Pretschner, editors, *Security and Trust Management - 6th International Workshop, STM 2010, Athens, Greece, September 23-24, 2010, Revised Selected Papers*, volume 6710 of *Lecture Notes in Computer Science*, pages 100–115. Springer, 2011.

[7] G. Goldberg, P. Okunev, M. Neumann, and H. Schneider. Distribution of subdominant eigenvalues of random matrices. *Methodology and Computing in Applied Probability*, 2:137–151, 2000.

[8] F. Gomes. Security alert: Fraudulent digital certificates. Technical report, SANS Institute InfoSec Reading Room, June 2001. http://www.sans.org/reading_room/whitepapers/certificates/security-alert-fraudulent-digital-certificates_679.

[9] K. Goto and R. A. van de Geijn. High-performance implementation of the level-3 BLAS. *ACM Transactions on Mathematical Software*, 35(1):4:1–4:14, 2008.

[10] K. Govindan and P. Mohapatra. Trust computations and trust dynamics in mobile adhoc networks: A survey. *IEEE Communications Surveys and Tutorials*, 14(2):279–298, 2012.

[11] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 403–412, New York, NY, USA, 2004. ACM.

[12] J. Huang and D. Nicol. A calculus of trust and its application to PKI and identity management. In *Proceedings of the 8th Symposium on Identity and Trust on the Internet*, IDTRUST'09, pages 23–37, New York, NY, USA, 2009. ACM.

[13] J. Huang and D. Nicol. A formal-semantics-based calculus of trust. *IEEE Internet Computing*, 14:38–46, September 2010.

[14] A. Jøsang. Trust extortion on the internet. In C. Meadows and C. Fernandez-Gago, editors, *Security and Trust Management*, volume 7170 of *Lecture Notes in Computer Science*, pages 6–21. Springer Berlin / Heidelberg, 2012.

[15] A. Jøsang. Probabilistic logic under uncertainty. In *Proceedings of Computing: The Australian Theory Symposium (CATS'07)*, January 2007.

[16] L. V. Orman. Transitivity and aggregation in trust networks. In *Proc. of the 21st Workshop on Information Technologies and Systems (WITS 2010)*, Dec. 2010.

[17] M. Pala. A proposal for collaborative internet-scale trust infrastructures deployment: the public key system (pks). In *Proceedings of the 9th Symposium on Identity and Trust on the Internet*, IDTRUST '10, pages 108–116, New York, NY, USA, 2010. ACM.

[18] M. Pala and S. W. Smith. Peaches and peers. In *Proceedings of the 5th European PKI workshop on Public Key Infrastructure: Theory and Practice*, EuroPKI '08, pages 223–238, Berlin, Heidelberg, 2008. Springer-Verlag.

[19] M. K. Reiter and S. G. Stubblebine. Resilient authentication using path independence. *IEEE Trans. Comput.*, 47:1351–1362, December 1998.

[20] H. Rifà-Pous and J. Herrera-Joancomartí. An interdomain PKI model based on trust lists. In J. Lopez, P. Samarati, and J. L. Ferrer, editors, *Public Key Infrastructure, 4th European PKI Workshop: Theory and Practice, EuroPKI 2007, Palma de Mallorca, Spain, June 28-30, 2007, Proceedings*, volume 4582 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 2007.

[21] S. Schiffner, S. Clauß, and S. Steinbrecher. Privacy and liveliness for reputation systems. In F. Martinelli and B. Preneel, editors, *Public Key Infrastructures, Services and Applications - 6th European Workshop, EuroPKI 2009, Pisa, Italy, September 10-11, 2009, Revised Selected Papers*, volume 6391 of *Lecture Notes in Computer Science*, pages 209–224. Springer, 2010.

[22] S. Schnettler. A structured overview of 50 years of small-world research. *Social Networks*, 31(3):165 – 178, 2009.

[23] R. C. Whaley, A. Petitet, and J. J. Dongarra. Automated empirical optimizations of software and the ATLAS project. *Parallel Computing*, 27(1–2):3–35, Jan. 2001. http://www.netlib.org/utk/people/JackDongarra/PAPERS/atlas_pub.pdf.