Peerster - NoTrust

Team NoTrust

Raja Soufi, Pierre Thevenet

January 8, 2018

Abstract

The goal of our project is to extend Peerster's file sharing to include a reputation system, and include authenticity, confidentiality and integrity for files and messages. This will allow safer file transfer, and a fairer network.

1 Goals and Functionalities

1.1 Authenticity, Integrity, Confidentiality and Web-of-Trust

1.1.1 Verifying the origin of a file - Pierre

One of the goal is to add a way to verify the origin of a file. That is if a peer A receives a file created by C from the uploader B, A can verify if the true origin is indeed C. If it is, then it can accept the file, and if not, decide to discard the file and decrease the reputation of the uploader B (see 1.2.1).

When requesting a download, the user will have to specify the origin of the file. Then the program will download it from the uploader (the destination of the file request) and if the uploader sends a correct file, it will accept it, otherwise drop it.

In the mean time, any file transfer in the network between a downloader A and uploader B will be signed by B so that A can verify that there is no man-in-the-middle. In fact the downloader does not necessarily expect that it comes effectively from B, but when our system will update the reputation of B according to its upload to A, A will need to verify if the file is true to what B sent.

1.1.2 Confidentiality with Private Messages - Pierre

In the user's perspective, this will allow the possibility to send encrypted private messages to a (single) known destination. Because the peerster network is open to eavesdroppers, this will offer confidentiality of the content of a message in one-to-one communication. Similarly to a PGP encrypted email, only the content of the message will be encrypted.

1.1.3 Automated Web of Trust (AWOT) - Pierre

The previous functionalities are implemented using public key cryptography. Usually the public keys are shared using a centralized PKI (Public Key Infrastructure), but our project will avoid this and implement a decentralized way to distribute public keys, based on the PGP "Web of Trust" model [1].

Once the public keys are obtained, the previous functionalities (CIA) are straightforward, therefore our project is focused on the Automated Web of Trust rather than the possibilities offered by it, as they only serve as proofs of our working system.

Our system will build a table of relation between peers and public key, and assign a trust level to each relation, representing the confidence that a particular public key belongs to a certain peer.

In summary our project also consist in a solution to the key validation and distribution problem.

This enables all benefits from having public keys (e.g. authenticity, confidentiality...), while not relying on trusting a common authority.

The user of our system will have to bootstrap the program with trusted public keys he knows from other users of Peerster, this will be the only action performed by the user. The keyring will then automatically update itself and present a visualization.

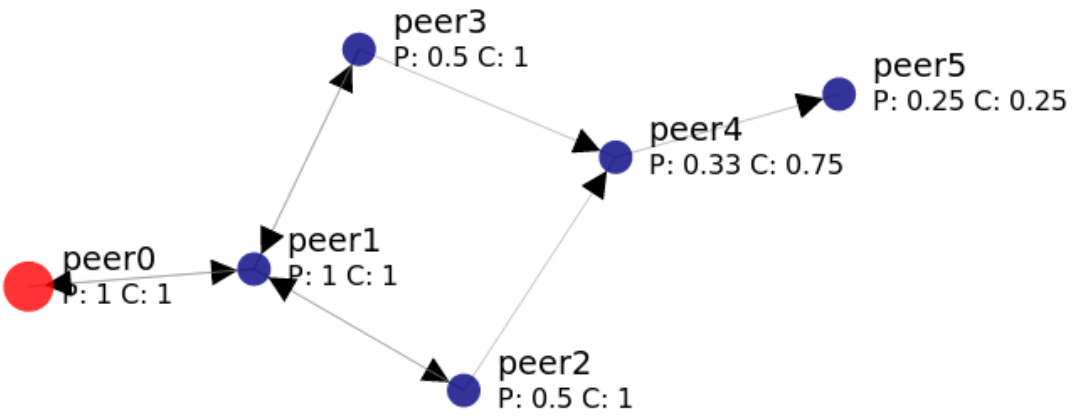

Figure 1: Example of a Key Ring obtained with our implementation, see Fig. 2 for explanations

We can imagine an extension in which the trust put in a particular peer depends on the user decision (think how a close friend can be more trustworthy than others).

However a visualization of the Web of Trust is given to the user (see Fig. 1) (via the gui localhost:8080/keyring by default). This is to demonstrate the output of the AWOT system.

1.2 Reputation

The goal of having a reputation system in our implementation of Peerster is to improve the efficiency and security of the network. Every peer A will compute a reputation for every other peer B based on three things:

- A's interactions with B
- The communicated reputation of B from other peers, weighted by those other peers' reputations relative to A
- The respective difference between the reputations of other peers communicated by B and those previously computed by A

Two types of reputation are computed for each peer. These are described below.

1.2.1 Signature-Based Reputation

Every time A receives signed data originating from C delivered by B (note that B and C might be the same peer), A can check if the data delivered by B actually comes from C by means of Public Key Cryptography (as seen in 1.1.1).

If the data turns out to be authentic, then A will increase B's reputation, otherwise A will decrease it. The amount by which A modifies B's reputation depends on the confidence level of C's public key that A holds, i.e. the degree to which we are confident that the public key we think is that of C is indeed that of C (see 2.1.3).

1.2.2 Contribution-Based Reputation

This type of reputation represents how much data B has delivered to A, relative to the total amount of data exchanged between the two. Yet, more importance will be given to more recent interactions, as we will see in more detail in 3.

For this, A will recompute this type of reputation after every interaction with B based on the

history of data flow between them. This means that in a situation where both peers send data to each other in equal amounts, each of them will have a maximal reputation for the other, whereas in a situation where A sends much more data to B than the other way around, B will have a high reputation for A and A will have a low reputation for B.

1.2.3 Usage

Having reputations of other peers will mainly serve two purposes. The first is related to choosing a confidence value for a public key association (see 4.2.2). The second, which affects more directly a peer's interaction with other peer's, is for choosing which peers to communicate with: A peer with a high reputation is one that we know (at least recently) has sends us authentic data regularly, and so it makes sense to prefer it over less reputable peer. That is after all the main purpose of having a reputations system. Other peers however may still be selected with low probabilities in order to "give other peers a chance" to increase their reputation and become more active.

2 Related Work

2.1 CIA and Decentralized PKI

2.1.1 Confidentiality, Integrity and Authenticity

This is a common problem, and solved using cryptography. Since in peerster we do not have a confidential channel to share a common key, we consider only public key cryptography. Two main solutions are RSA [6] and elliptic curve cryptography (ECC). While ECC may be more promising, RSA is more reviewed.

2.1.2 Decentralized PKI

Decentralized PKIs already exist, and the main goal for them is to redistribute the roles of the certificate authorities in the common centralized PKI approach, and usually this mean not storing all public keys at the same place, and trusting the system. KeyChains [7] and [2] are models for a decentralized PKI, the database of public keys are distributed which implies that a peer does not have every public keys, and that the retrieving of keys need more time than storing all keys in each system.

Implementing this in Peerster does not seem necessary, since every peer eventually save the entire set of active peers in the network (due to the specification). Our approach is to collect and save public keys for peers for each peerster client (total decentralization). Therefore this solution will save time for public key lookup (no decentralized database), at the cost of storage and network-redundant computations.

2.1.3 Automated Web of Trust - AWOT

The PGP Web of Trust Model [1] is a way to collect public keys using a social network, usually done in real life. For example, B signs the key of C, and gives the key to A, if A trusts B (which defines B as an "introducer"), A can trust that the given key indeed belong to C, and A can then signs the key of C.

In Peerster we would want it to be automated, which is not the case in the common PGP model.

One of the key point of the PGP model is the notion of trustworthiness of certificates and introducers. The trust of a certificate or introducer is decided by the user for each case, in Peerster we want it to be done automatically using the reputation and length of the certificate chain named "key ring" (see Fig. 2).

In the PGP model the level of trust is discrete (e.g. 'marginal', 'complete', 'untrustworthy' ...) but the meaning of this levels are not explicit and left for interpretation. We will therefore extend this model with a level of trust representing a 'probability' of confidence in the association key-peer. This idea is described in [5], which is a natural extension to the PGP model and seems reasonable.

2.2 Reputation

This is also a common problem, and different variations of a common approach have been studied in order to manage reputation in a distributed system while minimizing the possibilities for malicious peers to abuse the reputation system in order to gain reputation and do malicious activity.

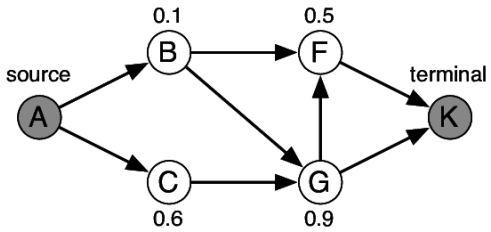

Figure 2: Example of a PGP Key Ring
Source : [5]
Here each vertex is a peer, and each edge Y $\rightarrow$ Z means that Y signed the key for Z. Each figure associated with a vertex Z correspond to the trust level given in the public key, we will name it the probability of Z, $Pr(Z)$.

For instance, [4] computes a global reputation for each peer, which uses local reputation values assigned to it by other peers, weighted by those peers' global reputations. [3] takes a similar approach but computes a trustworthiness of a node by doing a weighted average of the experiences of the nodes that interacted with that node. Another interesting approach is that of [8], in which each node is rated with a global reputation score which is based on weighted local scores from neighbors.

Our implementation will follow a similar approach, but will be slightly simpler for the sake of this project: Each peer A computes a local reputation for every other peer B, as we saw in 1.2. Yet, from time to time A will ask its n most reputable peers for their reputation tables, and will use the reputations in those tables, wheighted by the reputations of those n peers, to update its reputation table. Furthermore, when it receives the reputation tables of its n most reputable peers, A will compute some sort of Hamming distance between each peer's table and its own table and use that distance to update that peer's reputation.
However, when updating peers' reputation values with either of these two methods, the highest weight will still be given to the previous value we are updating. This way, values introduced by peers will only introduce small variations to the reputations in A's table, which will still be mostly controlled by A's own interactions with other peers.

3 Background

3.1 Cryptography

Since our project will be written in go, we will use standard cryptographic libraries for go. We will use RSA [6].

3.2 Web of Trust

We will implement the idea of [5].

3.2.1 Algorithm

In details, consider the network in Fig.2. For validating the public key of K, we follow the steps :

1. Compute the shortest paths from A to K

Here the shortest paths are $paths = \{(A, B, F, K), (A, B, G, K), (A, C, G, K)\}$

2. Then compute the "probability" p of the set of paths :
Define, for distinct peers $A_1, ..., A_n$ and path $path = (A_1, ..., A_n)$, $Pr(path) = \prod_{i=1}^{n} Pr(A_i)$, i.e. the confidence on each peer is independent from the confidence on the other peers.
$Pr(paths)$ is computed using the inclusion exclusion formula over the shortest paths.

Here $p = 0.581$

3. If $p > \lambda$ accept the key, else reject it.

Here, if $\lambda = 0.6$, reject the key : the certificate chain may be too long, or the confidence on the peers too low.

In short, the more shortest paths from source to the terminal and the lest hops in these paths, the higher the confidence. Adding to that the confidence level for each hop, and we get this algorithm.

3.2.2 Parameters

λ is the acceptance threshold. We will fix it to 0 for simplicity, meaning that we prefer to use cryptography with uncertain peer rather than not communicating with it.
The 'probabilities' assigned to each peers are computed using the reputation system and the distance from the source in the graph. We will use $min(rep_A, \frac{1}{d})$ as the probability of A, where rep_A is the current reputation of A and d is the distance from this peer in the key ring. This way we enforce the distance and the reputation to be acceptable.

3.2.3 Start

At first, a peer needs trusted peers to start collecting keys, this peers will be called "first introducers". They need to be given at startup, along with their public key (to effectively trust them).

3.3 Reputation

Most of the techniques used for this part have a rather mathematical aspect to them than algorithmical.

3.3.1 Signature-Based Reputation

For this type of reputation, we will use a simple formula with a nice property:

$$RS_n = \begin{cases} RS_{n-1} + f_+(c), & \text{if the received data is authentic} \\ \frac{RS_{n-1}}{f_-(c)}, & \text{otherwise} \end{cases} \quad (1)$$

Where RS_n is the signature-based reputation of peer B that sent the data at some point n, c is the confidence level of B's public key association, and $f_+(c)$ and $f_-(c)$ are functions of c that are used to control the impact of this new interaction on B's reputation based on the confidence level c. These can be chosen later on, but the important idea here is that behaving non-maliciously will only increase a peer's reputation linearly, while behaving maliciously will decrease it exponentially. This somehow makes it a lot harder for malicious peers to do a significant amount of malicious activity while still interacting in the system with a good reputation.

3.3.2 Contribution-Based Reputation

For this type of reputation, we will use an Exponentially Weighted Moving Average:

$$RC_n = \begin{cases} RC_initial, & \text{if } n = 1 \\ \alpha \cdot S + (1 - \alpha) \cdot RC_{n-1}, & \text{otherwise} \end{cases} \quad (2)$$

Where RC_n is the contribution-based reputation of peer B that sent/received data at some point n, $RC_initial$ is an initial contribution-based reputation constant that is assigned to all peers at the beginning, α is the main parameter of the Moving Average, and S is 1 if B was the sender during this interaction and 0 if it was the receiver.

Note that the value of α determines how much weight we give to the current interaction and to the history of interaction.

4 Design and Architecture

Our project is built on the common Peerster, adding to it our modules. In particular the transfer of messages (broadcast and private) and file sharing are the most important features our system will use.

4.1 Systems

Our system can be easily devided into two different but interconnected systems : the Reputation System (RS) and the Web of Trust system (AWOTS).
Figure 3 illustrates the overall structure of the system and the interactions between the two subsystems.

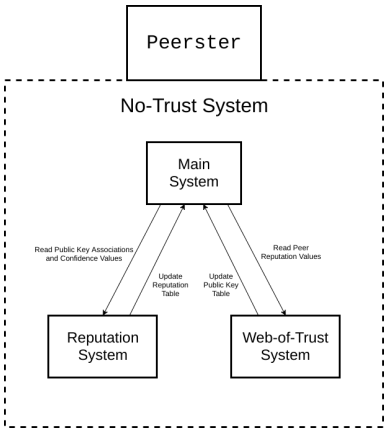

Figure 3: General architecture of the No-Trust system

Pierre will do the Automated Web of Trust System, and Raja will do the Reputation system.

4.1.1 Web of Trust System

This system will be responsible for updating a table in the Main System which will include $(id, k_{id}^{pub}, confidence_{id})$ for each known peer with identifier id, where k_{id}^{pub} is the (trusted) public key of this peer, and $confidence_{id}$ is the confidence level put in the association $id \leftrightarrow k_{id}^{pub}$.
This table will be used for getting the public keys of the different peers, and the confidence value will be used for the Reputation System.

At startup, trusted peers with their corresponding public keys will be given. Then each peer will start to sign its neighbors' public keys, and send the keys with signature to everyone in the network via rumor messages.
When receiving a signature and a key, the AWOTS will update the key ring, and run the algorithm described in 3.2.1 if enough information. If the key is accepted, then the peer is added to the trusted peers, and then continue the process. This steps will add more and more public keys to the table.
Periodically the system will update the confidence levels in the table by running the same algorithm on the already acquired keys.

Problems may arise, like collision of two different keys for the same peer (same identifier), in this case the system will pick the biggest subset of minimum paths to the identifier with the same key and "forget" the other paths (having different keys). Therefore for a key to be accepted, it has to be part of the maximum number of minimum paths.

4.1.2 Reputation System

This system will be responsible for updating a separate table in the Main System which will include (id, RS_n, RC_n) where RS_n and RC_n are the signature-based and contribution-based reputations respectively, as we saw in 3.3. This table will be used both to choose peers to interact with as well as to compute confidence levels in the Web of Trust System.

Given that the very definition of reputation relies on history, all peers will have the same initial reputation, which seems like the most natural state to start the system with. Then, as we saw earlier, the two reputation values (in each entry of the table) will be updated on every interaction with the corresponding peer, as well as periodically when peers ask their most reputable peers for their tables.
Obviously, we might get completely invalid reputation values from a peer (e.g. outside the set bounds), in which case we might consider penalizing that peer even more.

4.1.3 Main System

The main system will be implemented with base the common Peerster project.
We will add features like confidentiality, integrity and authenticity to the messages and files, in this main module. The difference in the messages will be set by some flags, which will indicate if the message is an encrypted file or has a signature. Then the messages can be demultiplexed according to this flags.

As discussed earlier, the reputation table will be used to choose a peer to interact with, for example when a peer wants to ask for the public key of another peer, or when it wants to send a status packet to update its rumors.
Highly reputable peers will be chosen with a very high probablity. For instance, assuming the table is sorted somehow based on the reputation values, we might select a peer from the upper half of the table with probability 0.9 and from the lower half with probability 0.1, and recursively select a half with probabilities that are more and more balanced with every iteration. This will occasionally give low reputable peers a chance to make a "come back" and contribute in the system.

4.2 Systems interoperability

4.2.1 Use of WTS for RS

The output of the Web-of-Trust system will be used when updating the signature-based reputations of peers (hence the name). The confidence level for each public key association is a perfect variable to be used as a liability measure: If we're not completely sure that the public key we have for a peer is indeed his public key, it wouldn't be completely fair to penalize him as much as if we were 100% sure of his public key.

4.2.2 Use of RS for WTS

The reputation system will be helpful for determining the probabilities as discussed in 3.2.2, by providing the signature-based reputation (see 1.2.1). This reputation gives information about the honesty of a peer : if this peer sent badly signed packets or not. Therefore we will assign the confidence level of each peer in function of its reputation.

4.2.3 Use of WTS for the cryptographic functionalities

The WTS will provide the public keys needed for encryption and verification of signatures.

5 Evaluation Plan

5.1 Cryptography

We will verify correctness of the cryptographic functionnalities (confidentiality, integrity and authentication) with some cases (e.g. sending well/badly signed files, sending encrypted messages and decoding them at receiver, ...), only to check that the standard libraries have been well implemented in our project. We will not evaluate the "crypto" and "crypto/rsa" libraries and will expect them to be correct.

5.2 Web of Trust

To measure the correctness and performance of our key distribution system (WTS), we will create a scenario with a network of Peerster nodes, and follow the updates of the key ring of each node, checking the time at which the ring stops updating, and checking the correctness of the ring. The state of the WTS system is the key ring, therefore checking the key ring is enough to evaluate the correctness of the system. The performance will be tested, but because of the nature of Peerster and the possibility of network bottlenecks, we will check that the key ring is complete after a reasonable amount of time. Moreover since the public keys are obtained via this system, testing in 5.1 will effectively test if the WTS works.

5.3 Reputation

To evaluate the reputation system, we will follow a similar approach as for the rest of the system. We will run the system in a controlled environment where we send particular specific messages that we know should trigger certain changes in the reputation tables in order to check that these changes are actually happening.

We will check for example that signature-based reputations are being decreased exponentially when receiving inauthentic data and increased linearly otherwise (as described by equation 1). We will also check that the moving average for contribution-based reputation discards history more and more over time (as described by equation 2). Finally, we will make sure that a peer chooses in most cases its most reputable peers before initiating interactions such as rumor mongering.

6 Discussions

6.1 Automated Web Of Trust

6.1.1 Example and Collisions of keys

Collision of two different keys for the same peer (same identifier) is possible. In this case the system will pick the biggest subset of minimum paths to the identifier with the same key and "forget" the other paths (which have different keys). Therefore for a key to be accepted, it has to be part of the maximum number of minimum paths.

Example 6.1 *In this scenario, we see a simplified keyring with the only important edges from A's perspective. We are interested in G's public key. The minimum paths in Fig 4 are (we do not consider the source nor the terminal since they do not count in the computations) :*

$$\{B, D\}, \{B, E\}, \{C, F\}$$

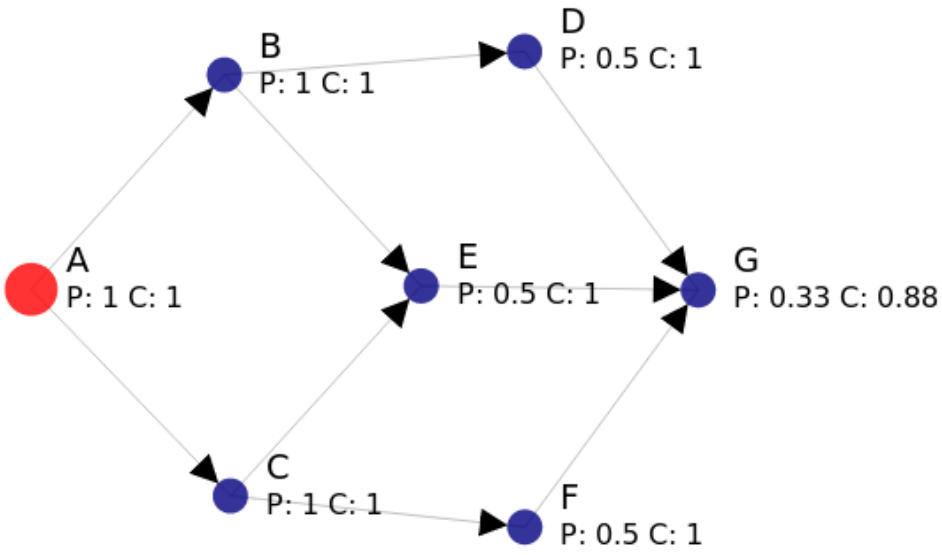

Figure 4: Key Ring with all good nodes

However the minimum paths in Fig 5 are :

$$\{B, D\}, \{B, E\}$$

The system do not consider the path $\{C, F\}$ anymore because F sent a key different from D and E and therefore is no longer in the same set of paths.

We can observer the confidence in G's key dropping from 88% to 75% due to F no longer being considered.

Of course if E becomes malicious too and advertises the same key as F, we will trust E and F because they become majority.

If E is malicious but sends a different key than F, the system will choose one of those three keys.

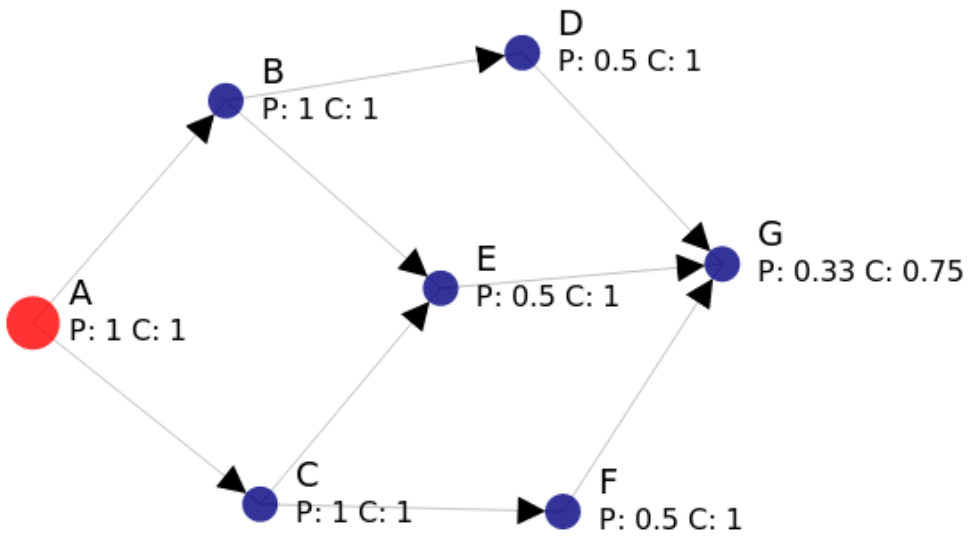

Figure 5: Key Ring with malicious F node

This setup can be reproduced using the tests given with the source code.

References

[1] Alfarez Abdul-Rahman. The pgp trust model. In *EDI-Forum: the Journal of Electronic Commerce*, volume 10, pages 27–31, 1997.

[2] Karl Aberer, Anwitaman Datta, and Manfred Hauswirth. A decentralised public key infrastructure for customer-to-customer e-commerce. *International Journal of Business Process Integration and Management*, 1(1):26–33, 2005.

[3] Zoran Despotovic and Karl Aberer. P2p reputation management: Probabilistic estimation vs. social networks. *Computer Networks*, 50(4):485–500, 2006.

[4] Minaxi Gupta, Paul Judge, and Mostafa Ammar. A reputation system for peer-to-peer networks. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, pages 144–152. ACM, 2003.

[5] Rolf Haenni and Jacek Jonczy. A new approach to pgps web of trust. In *EEMA07, European e-Identity Conference*, 2007.

[6] K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch. Pkcs #1: Rsa cryptography specifications version 2.2. RFC 8017, RFC Editor, November 2016.

[7] Ruggero Morselli, Bobby Bhattacharjee, Jonathan Katz, and Michael Marsh. Keychains: A decentralized public-key infrastructure. Technical report, University of Maryland, College Park College Park United States, 2006.

[8] Runfang Zhou and Kai Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on parallel and distributed systems*, 18(4), 2007.