Survey of Six Myths and Oversights about Distributed Hash Tables' Security

Sylvain Dahan and Mitsuhisa Sato
High Performance Computing System Laboratory
Institute of Information Science and Electronics,
University of Tsukuba, 1-1-1, Tennoudai, Tsukuba, 305-8573 Japan
sdahan@hpcs.cs.tsukuba.ac.jp

Abstract

Distributed Hash Tables (DHT) was not designed to be secure against malicious users. But some secure systems like trust and reputation management algorithms trust DHT with their data. Several propositions have been made to make DHTs appear secure but they fail in practice. We review some of those propositions and explain why they do not work. Our main conclusion is that DHT should not be used to create secure systems.

1. Introduction

People interact with each other. Some cheat, but most of the people do not. So, Community, then civilization can emerge and benefit everybody. Even if bartering is profitable most of the time, it can be costly for a person if the other one cheats. This problem leads several researchers to design trust and reputation management algorithms that tell to a user who he can trust or not by analyzing past transactions. To work, those algorithms need to store information about the past transactions inside a database.

In one hand, this database needs to be secured against malicious users because anyone that can modify the data, can also control the algorithm output. If an algorithm is fed with data that said A is trustworthy, even if A is not trustworthy in reality, then the algorithm has no other choice and tells that A is trustworthy. In the other hand, peer-to-peer Distributed Hash Tables (DHT), the last fashion distributed database, are reputed to be highly scalable and completely distributed. Some of their applications are file systems, look-up directories or database for trust and reputation management mechanisms.

However, peer-to-peer Distributed Hash Tables (DHT) were not designed to be secured against malicious nodes. Some issues about their routing mechanisms and node identification have already been documented [11,25–28,31] and many others are too obvious to be documented. Knowing the DHT's security weakness, it is difficult to understand why some people use them to build secure systems. They probably forgot one of the golden security rules: a security system is only as strong as its weakest link.

To hide this issue, the peer-to-peer trust and reputation management community has developed certain unfounded claims about DHT's security. Some of those claims are used and believed by most of the community, some others just by a few. Another reaction was to simply forget some of the issues. The aim of this article is to describe some of those myths and oversights and to explain what is wrong with them. All the described issues have already be documented in various place. The contribution of this article is to describe all of them in a unique place in a way that are understandable by non specialists. We hope that researchers will stop to use DHT to build secure systems after reading this document.

In "Securing P2P networks using peer reputations: is there a silver bullet?" [14], P. Dewan and P. Dasgupta already explained what should be done. This document focus on what is going wrong. Each myth comes with a list of articles co-wrote by the members of the TRAM'07's organizing committees that supports it. We consider believes are transitive: if A believes in M and B uses A but says nothing about M then we consider that B also believes in M.

We start by giving a short description of DHTs. Then we criticize several myths. Finally, the conclusion gives our personal vision about DHT and about how researchers should design their algorithms.

2. DHT Terminology

This section gives a simplified description of what is a DHT. Readers are encouraged to read the survey [22] to get more detailed descriptions.

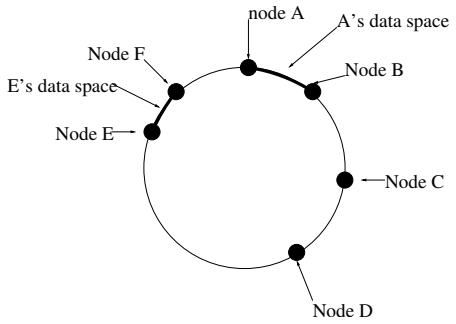

Figure 1. A DHT identity space.

A DHT can be view as a directory that stores identified data. Each data has an identifier and users access the data through the identifier. The list of identifiers is sorted to allow quick access to the data. The identifier space (*ID space*) is bounded but infinite. In Fig. 1, the ID space is represented by the big circle. This ID space is split and distributed between several users. To do it, each user has a name that is an element of the ID space and each user manages the data that are included between his name and the next user's name. This is possible because the ID space is a sorted list. We call *X's data space* the ID space part managed by a node X.

3. Current Implementations are Not Reliable

Several DHT implementations exist. Some like Chord and Pastry are widely popular and several researchers [6, 7, 10, 19, 32] wrote that they could be used to implement trust and reputation management algorithms. The problem is that none of them are secure.

Myth 1 *We can use a DHT like Chord or Pastry to build secure systems.*

To understand the issue with the Chord implementation, it is important to remind that software quality plays a big role in software security. Many software vulnerabilities come from bugs like buffer overflow, condition race or bad input checking. From the Chord Project home page [2], we can read:

> At this point no official release for Chord is available, but you are welcome to check out the latest version from the CVS repository. This version is experimental, under active development, and may frequently be broken.

The previous citation let us to presage that Chord has an important number of bugs. So, we have to consider the existence of bugs that allow an attacker to get complete control of the database or worst: like giving shell access via a buffer overflow. For this reason, Chord should not be considered as reliable and should not be used to build secure systems. What about Pastry?

Pastry has two implementations. The first, SimPastry, is just a simulator, the second, FreePastry, as the following limitation [4]: "*Minimal security (no support for insecure networks or malicious nodes)*".

In brief, none of those two systems propose a reliable implementation that can be used to build secure systems.

Fact 1 *Popular DHT implementations should not be deployed on insecure network without compromising the system security.*

We found that another DHT implementation, P-Grid, is popular in the trust and reputation management community [7, 29, 32]. It was difficult to judge the P-Grid security when this document was written: we were unable to find the P-Grid source code nor adverse security reviews. By default, the security community tends to untrust algorithms that did not have been reviewed sufficiently, and we encourage doing so.

4. Secure Identifier is an open issue

To make a DHT secure, several issues have been unsuccessfully addressed. One of them is the need of what they call *secure ID*. As we explained before, in a DHT, a user manages all the data include in its data space and this data space is defined by the user name. If the user can freely choose his name, then he can virtually manage any stored data. Secure ID algorithms have been proposed to solve this issue. Two of them are IP based identities and Public Key Infrastructures (PKI).

4.1. IP based identity

P. Dewan and P. Dasgupta have proposed to implement IP based safeguard [14]. The aim is to create an identifier based on the user IP address. The idea is that a user only has few IP addresses in disposition, and by logical consequence, few identities to choose from. This is similar to the IP based secure ID proposed by Chord

who generates the identity by hashing the user IP address with the SHA-1 algorithm [30].

Myth 2 *If a peer identifier is made by hashing its IP address, then we have a secure ID.*

In theory, IP spoofing makes this kind of secure ID inadequate. The concept of IP spoofing [12] is to modify an IP packet header in a way that the packet appears to come from another host. If this is coupled with the possibility to capture the response [24], then an attacker can virtually own any IP address. If an attacker can use any IP address, it makes no sense to create a system where the security lies in the limited number of available IP addresses.

Fact 2 *In theory, a peer can "spoof" the IP address of any other computer.*

However, this kind of attacks is difficult to put into practice and an attacker will probably not use this weakness. The real problem with the myth 2 is that using IP address as unique identifier is not possible in practice. Chord itself is a good illustration of this problem. The Chord team wrote "*Requiring (and checking) that nodes use IDs derived from the SHA-1 hash of their IP addresses makes this attack harder*" [30]. But their code [1] shows[1] something different:

```
bool is_authenticID (const chordID &x,
         chord_hostname n, int p, int vnode) {
  static int max_vnodes = 0;
  if (max_vnodes == 0) {
    bool ok = Configurator::only().get_int(
                "chord.max_vnodes", max_vnodes);
    if (!ok) {
      max_vnodes = 1024;
    }
  }
  if (p < 0 || p > 65535)
    return false;
  if (vnode > max_vnodes)
    return false;
  chordID ID = make_chordID (n, p, vnode);
  return (ID == x);
}
```

The `is_authenticID` function aims to limit the number of identifiers that a peer can use by checking that the identifier follows strict construction rules. But the default Chord limit is not really useful because a peer can choose between more than 6.7 millions[2] of different identifiers.

`max_vnodes` defines the maximum number of accepted virtual nodes by peers. "Virtual nodes" is a mechanism that allows a peer acting as several virtual peers. This is useful to put more load on powerful computers. In a security context, we can consider that a peer can only have one virtual node and then reducing the number of available identifiers to 65 534. But this number is still important and gives too many choices if the number of users is no enough, let us say 100 000 connected users.

Another step to reduce the number of available identifiers is to not take into account the port number. But this solution is not practicable because today, lots of computers share the same IP address, thanks to firewalls and Network Address Translation mechanisms. The consequence of only allowing one peer per IP address has too bad effects on corporate networks that only have few IP addresses that are visible from the Internet to be practical in a general context.

Fact 3 *In practice, it is not possible to have one IP address per peer without banning lot of users from the system.*

4.2. Public Key Infrastructure

The universal PKI dream has been proposed numerous times [10,29,32]. The idea is that the PKI gives a secure ID to every user. The problem is that the universal PKI dream does not work in practice. R. Anderson [9], N. Ferguson and B. Schneier [17] gave comprehensible explanations about PKI issues in their books.

In short, the universal PKI works the following way: it supposes the existence of a central Certificate Authority (CA) that knows everybody and who are trusted by everybody. We call this CA *god*. Then everybody has a certificate that contains his name, his signature and god's signature to prove the certificate authenticity. To authenticate a user A, we just need to make A signing a piece of paper and checking his certificate.

There are two major issues with it. Firstly, not everybody believed in the same god. Practically, it is not possible to have a CA that is trusted by everybody. Secondly, we never meet a god that knows everybody. In practice, by taking into account that fake documents exist, it is almost impossible for a CA to know the real identity of thousands of people that it never mets. This is why Verisign allows you to get a certificate under a false name for $19.95 [3].

Thus scalable secure ID is still an open issue and we should consider, for security reason, that a user can choose its identifier freely.

1 We took the liberty to reformat the code and to delete comments.

2 $1 \times 1024 \times 65534 > 6.7 \times 10^6$

5. Randomness is Not a Protection

K. Aberer [5] seams to believe that if there are lots of users, then the probability that a defined user stores a defined data is low.

Myth 3 *Because there are lots of users, the probability that a user stores [per accident] a data is small.*

The problem of this myth lies in the "per accident" assumption. It is true that if there are lots of users, then the probability that a random user stores a specified data is low. But this hypothesis is false ; a malicious user is not a common random user. Like explained in the secure ID section, if a user can choose his identifier, then he can store any data that he wishes and in his article, K. Aberer explicitly lets the user choosing his identifier.

Fact 4 *Even if there are lots of users, a malicious user will take the right name to be in the right place and so, he will manage his coved data.*

6. Replication is not a silver bullet

Most Researchers [5–7, 13, 19, 29, 32] are aware that in a peer-to-peer system, a malicious peer can modify the data that it stores. To solve this problem, they propose to duplicate the data.

Myth 4 *Because a peer can store a coveted data, storing the data in various place improves the robustness.*

There are two problems with this myth. First, a peer can have multiple identities. This is called a Sybil Attack and has been documented by J. R. Douceur [15]. Without secure ID, a malicious peer can choose a targeted data space. If he can do it once, he can do it for all the duplicates by using several identities.

Fact 5 *If a malicious peer can choose a targeted data space once, then it can choose as many targeted data spaces as it wants.*

By duplicating the data, we can actually reduce the security instead of increasing it. For example, if there are two copies, then the attacker as two times more chance to store a copy, and if it modifies one copy, how can we distinguish the authentic data?

If we have x copies, we need to be sure that an attacker store less than $\frac{x}{2} - 1$ copies and that the consensus protocol used to find the authentical value has no vulnerability. So depending on how easy it is for a malicious peer to store a coveted data, using duplicates can increase or decrease the reliability of the DHT.

Fact 6 *By having several copies, it is easer for an attacker to control some copies of the data and everything is lost if an attacker control the majority of them. The added protocol to discriminate the authentical data open a new path to attack.*

So replication is not a silver bullet. If it is implemented the right way and if secure ID is available, then replication is a good way to increase the robustness of the stored data. But firstly, there is no working secure ID mechanism that we are aware about. Secondly, we need to consider the pro and cons of using replication mechanisms because the replication algorithm adds a new attack vector to the system.

7. Access Control is Needed

Common DHTs do not implement read, write and delete access control. Cryptography has been proposed to resolve this issue. Encryption can be used to deny read access and signature can be used to ensure data integrity. Deletion is a more difficult matter. It is possible to deny deletion access to everybody, but the cost in the long run — in term of data processing — can become considerable. However, none of these techniques assure data availability: an attacker can always store lot of data under a specified identifier to drown the authentic data in the middle of the fake ones.

But the real issue is not how to implement a secure DHT access control mechanism, but how to define correct access policies. Proposed trust and reputation algorithms need to give write access to every peer. So, how is it possible to protect against data sabotage if anyone needs and has write access to run the algorithms?

Oversight 1 *Because the algorithms are peer-to-peer, every peer is granted with database write access. Because everyone has write access to all the data, anyone can easily subvert the database.*

The lack of access control is a common flow in trust and reputation management algorithm. One notable exception is the TrustGuard algorithms proposed by M. Srivatsa *et Al.* [29]. They resolve this issue by making two hypotheses: a peer can prove that a transaction really happened and the database only accepts data concerning proved transaction. However, the presented deletion algorithm is not secure and can be abused to erase any data.

Another solution is to run a trust algorithm to know if some data are trustworthy or not. The problem with this solution is how we check the data that are used to check the trustworthiness of a data?

R. Anderson wrote a good introduction [8] about access control issues that can be used as a starting point for discussion about how to define access control policies.

8. Taking Botnet into account

Some overconfident researchers [19, 32] propose to use their trust and reputation management algorithms in financial applications. But, it seams that they do not take into account the various tools that are available to organized crimes. Botnets [16, 23] are one of these tools and its usage has a devastating effect on an important hypothesis:

Myth 5 *It is practically impossible for an attacker to control 100 000 or more computers.*

The truth is that, even for no technical people, it is easy to get control of several hundreds of thousands computers, thanks to the botnets black market [18]. A botnet is a set of computers that has been compromised by a worm which continuously attacks other computers and that listen for orders to execute. This way, a botnet owner has an army of computers at his command. Botnets are mainly used to send spam, to install spyware (some companies pay few dollars for every install) and to do extortion (if a company refuses to pay, they shutdown its entire network with a distributed denial of service attack). Usually, a botnet controls at least 10 000 computers and a botnet with 1.5 million computers has already be shutdown by the Dutch police [20, 21].

Fact 7 *Organized crime has potentially access to millions of computers.*

It is true that it is a bit expensive to rent a botnet to get priority slots on a files sharing service. But if there is some money to steal, then the designer must take botnets into account.

This is a serious issue, because all the reviewed algorithms suppose that the attacker control a minority of computers. If he takes control of the majority of the active computers (users that are currently connected to the service) then he took over the system.

9. Conclusion

Considering that the system complexity is one of the first causes of security failure, it is not surprising that using a DHT brings a plethora of problems. Lack of secure ID, lack of good implementations, lack of secure routing, lack of access control and the existence of botnets are some of the problems we face. Secure DHTs do not exist and will probably never exist due to their complexity.

It is difficult to understand why so many researchers use them to describe their algorithms in the first place. Naivety, buzzwords and fashion come in mind. But those are not acceptable explanations. The truth is that none of the reviewed articles really need a DHT. They only need a secure and scalable database. Proposing a DHT as an example is one thing, making the algorithm dependent of it is another.

Continuing to follow this way is not serious and we recommend to use a modular design. Specifying that the algorithm needs a secure and scalable database instead of a particular technology gives you three advantages. Firstly, modular designs reduce the complexity of the system making it more understandable and so, it is easer to find its weakness. Secondly, if a particular implementation becomes weak, you can use another one without the need to completely review the algorithm. Finally, this allows the chance for database specialists to propose their best implementation without having to understand how trust and reputation management algorithm really works.

References

[1] chord-0.1-20060908.tar.bz2, utils/id_utils.C, line 358.

[2] The Chord/DHash project. `http://pdos.csail.mit.edu/chord/`.

[3] Digital ids for secure email. `http://www.verisign.com/products-services/security-services/pki/pki-app%lication/email-digital-id/index.html`.

[4] Pastry - a scalable, decentralized, self-organizing and fault-tolerant substrate for peer-to-peer applications. `http://freepastry.org/`.

[5] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, R. Schmidt, and J. Wu. Advanced peer-to-peer networking: The P-Grid system and its applications. *Praxis der Informationsverarbeitung und Kommunikation*, 26(3), 2003.

[6] K. Aberer and Z. Despotovic. P2P reputation management: Probabilistic estimation vs. social networks. *Journal of Computer Networks*, 2005.

[7] Karl Aberer and Zoran Despotovic. Managing trust in a peer-2-peer information system. In *Conference on Information and Knowledge Management*, pages 310–317, Atlanta, Georgia, USA, 2001. ACM press.

[8] Ross Anderson. *Security Engineering, a guide to building dependable distribut ed systems*, chapter 4, Access Control. Wiley, 2000.

[9] Ross Anderson. *Security Engineering, a guide to building dependable distributed systems*. Wiley, 2000.

[10] Yu Bin, Munindar P. Singh, and Katia Sycara. Developing trust in large-scale peer-to-peer systems. In *First IEEE Symposium on Multi-Agent Security and Survivability*, pages 1–10, 2004.

[11] Miguel Castro, Peter Druschel, Ayalvadi Ganesh, Antony Rowstron, and Dan S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *ACM SIGOPS Operating Systems Review, 5th symposium on Operating systems design and implementation*, volume 36, pages 299–314, 2002.

[12] daemon9, route, and infinity. IP-spoofing demystified (trust-relationship exploitation). *Phrack Magazine*, 1996.

[13] Prashant Dewan. Countering identity farms in reputation systems for P2P networks. Technical report, Arizona State University, 2004.

[14] Prashant Dewan and Partha Dasgupta. Securing P2P networks using peer reputations: Is there a silver bullet? In *IEEE Consumer Communications and Networking Conference (CCNC 2005)*, 2005.

[15] John R. Douceur. The Sybil attack. *Lecture Notes In Computer Science*, 2429:251–260, 2002.

[16] Noam Eppel. Security absurdity: The complete, unquestionable, and total failure of information security, 2006. `http://www.securityabsurdity.com/failure.php`.

[17] Niels Ferguson and Bruce Schneier. *Practical Cryptography*. Wiley Publishing, Inc., 2003.

[18] Dan Goodin. Calif. man pleads guilty to felony hacking. *Breitbart.com*, Jan. 23 2006.

[19] Kai Hwang and Runfang Zhou. PowerTrust: a robust and scalable reputation system for trusted peer-to-peer computing. Manuscript submitted to IEEE Transactions on Parallel and Distributed Systems.

[20] Gregg Keizer. Dutch botnet bigger than expected. *InformationWeek*, Oct 21 2005. `http://www.informationweek.com/story/showArticle.jhtml?articleID=172303%265`.

[21] Gregg Keizer. Dutch botnet susptects ran 1.5 milion machines. *TechWeb*, Oct 21 2005. `http://www.techweb.com/wire/security/172303160`.

[22] Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE*, pages 72–93, 2005.

[23] John Markoff. Attack of the zombie computers is growing threat. *The New York Times*, 7 Jan. 2007.

[24] S. Murphy. BGP security vulnerabilities analysis. Technical Report RFC4272, Internet Engineering Task Force, 2006.

[25] Thomas Reidemeister, Klemens Bohm, Paul A. S. Ward, and Erik Buchmann. Malicious behaviour in content-addressable peer-to-peer networks. In *3rd Annual Communication Networks and Services Research Conference (CNSR'05)*, pages 319–326, 2005.

[26] Atul Singh, Miguel Castro, Peter Druschel, and Antony Rowstron. Defending against eclipse attacks on overlay networks. In *ACM SIGOPS European Workshop, 11th workshop on ACM SIGOPS European workshop: beyond the PC*, Leuven, Belgium, 2004.

[27] Emil Sit and Robert Morris. Security considerations for peer-to-peer distributed hash tables. In *Lecture Notes in Computer Science, Revised Papers form the First International Workshop on Peer-to-Peer Systems*, pages 261–269, 2002.

[28] Mudhakar Srivatsa and Ling Liu. Vulnerabilities and security threats in structured overlay networks: A quantitative analysis. In *20th Annual Computer Security Applications Conference (ACSAC 2004)*, Hilton Tucson El Conquistador Tucson, Arizona, USA, December 2004.

[29] Mudhakar Srivatsa, Li Xiong, and Ling Liu. Trustguard: countering vulnerabilities in reputation management for decentralized overlay networks. In *14th international conference on World Wid Web*, pages 422–431, Chiba, Japan, 2005.

[30] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM 2001*, pages 149–160, San Deigo, CA, 2001.

[31] Dan S. Wallach. A survey of peer-to-peer security issues. In *International Symposium on Software Security*, Tokyo, Japan, 2002.

[32] Li Xiong and Ling Liu. PeerTrust: Supporting reputation-based trust in peer-to-peer communities. *IEEE Transactions on Knowledge and Data Engineering*, 6, 2004.