'peer to peer' Public Key Infrastructure

David Irvine, email: david.irvine@maidsafe.net, LifeStuff David

maidsafe.net limited (registered in Scotland Sc 297540)

September, 2010 ***Abstract*—This paper presents a system of validation that utilises asymmetrical encryption to create a Public Key Infrastructure (PKI) in a manner that requires no servers or centralised control. This method provides an extremely coherent and mathematically secure method of validation that can be employed in any modern network or system but is very well suited to distributed networks and in particular overlay networks such as Distributed Hash Tables (DHT's).**

***Index Terms*—security, freedom, privacy, DHT, encryption**

CONTENTS

I Introduction 1
- I-A The Issues Addressed by this Paper . . 1
- I-B Conventions Used 1
- I-C Asymmetric public key encryption . . . 1
- I-D Cryptographically secure hash 2
- I-E Return to asymmetric encryption to produce digital signatures 2

II Implementation 2
- II-A Methodology 2
- II-B Simple Configuration 2
- II-C Addition of a key revocation method . . 2
- II-D Passing ID between parties 2
- II-E Key validity checks 3
  - II-E1 Check validity by looking up key 3
  - II-E2 Check validity by message passing 3
- II-F Combined with DHT 3

III Improvements with security of private keys 4

IV Conclusions 4

References 4

Biographies 4
- David Irvine 4

LIST OF TABLES

I Message structure 3

LIST OF ALGORITHMS

1 Hash(Public key) == ID 2
2 Hash(Public key + Signature) == ID 2
3 if (Hash(Public key + Signature) == Unique) then Hash(Public key + Signature) == ID == Network address 3

I. INTRODUCTION

VALIDATION is a requirement for any connecting parties who are to exchange information to ensure uniqueness, proof of identity and generally any situation where the information requires clarification or authentication. This paper presents a system that uses many of the techniques used today, such as asymmetric encryption techniques.

Another significant differentiator in the system presented is the ability for such a system to create multiple key-pairs and restrict a key to identification only. This key can only identify the node or identity but cannot take any action aside from that, or the actions are limited. All other (and hopefully significant) actions are the responsibility of a non identification key, by this an identification key is the key used to validate a particular identity.

This type of system has been the subject of many a wish list[1] for some time now

A. The Issues Addressed by this Paper

The issue with today's PKI networks is the balance between trusting a chain of control or trusting known entities in a so called 'web of trust system'. The former has to exist in a manner to be secure at the root and more importantly trusted. The latter is open to abuse should enough untrustworthy entities manage to rate each other as trusted in an manner synonymous with a Sybil attack on such networks where groups surround good entities and effectively intercept or hide information.

Both of these systems are very much open to abuse and neither are able to provide a secure system to allow freedom of the creation of identifiable nodes.

B. Conventions Used

This paper does not require all the operations listed below, but lists these for example implementations, which are outlined later.

Hash = Hash function such as SHA or MD5 etc.

Symm = Symmetrical encryption such as AES, 3DES etc.

Asym = Asymmetric encryption such as RSA or ECC etc.

PBKDF2 = Password-Based Key Derivation Function or similar

Kpriv = private key (used to decrypt public key encrypted data or to sign data)

Kpub = public key (used to encrypt)

netput[K]/[V] = put a value (V) on the network with a key (K).

netget[K] = get value from network using the key (K).

C. Asymmetric public key encryption

This paper makes use of public key cryptography. This is a system of encryption that does not require passwords or keys

to be passed around, rather it allows the publication of a public key $Kpub$. This key can be thought of as the encryption key, where any data encrypted by this key can only[1] be decrypted by the holder of the private key $Kpriv$. Therefor the private key can be considered to be the key to unlock the data. It should be noted, however that the opposite is also true in that data encrypted with the private key can be decrypted by the public key. This seems strange and useless (as everyone has access to the public key), but is used to excellent effect and described shortly.

D. Cryptographically secure hash

A hash function can be thought of as a digital fingerprint. Just as a fingerprint of a person is supposed to be unique, then a digital hash function is also supposedly unique. We have all heard of two people with identical fingerprints (but perhaps have never met any !) and in the digital world it can be possible to get two peices of data with the same hash result. This is referred to as a collision and reduces the security of the hash algorithm. The more secure the alogirthm then the likelyhood of a collision (or two people having the same fingerprint) is reduced. It is very similar to taking more points of reference on an actual fingerprint to reduce collisions in that area of science also. This is an area where both systems share a similarity in the increasing complexity of measurement and recording of data points of reference.

In cryptographically secure hashing the data is analysed and a fixed length key is produced, this key is called the hash of the data. Again similarly with human fingerprinting a hash cannot reveal data just as a fingerprint cannot reveal a person (i.e. you cannot recreate the person from the print) and you cannot recreate the data from the hash, otherwise it would be a miraculous compression scheme indeed.

Early hash algorithms are considered broken, such as md4, md5 and even early SHA schemes, this is a bad representation as they are not broken, they simply can allow too many collisions and larger decriptors (keylengths) and more efficient algorithms are almost always required.

Therefor a hash is merely a best attempt to distinguish a piece of data by a fixed length string representation of the contents of the data and of only that data[2], it is really as simple as that.

E. Return to asymmetric encryption to produce digital signatures

In I-C we completed the section with a bit of a laugh at the thought of using a private key to encrypt data that the public key could decrypt. This is, as promised, used to great effect with the addition of hashing as described in I-D.

If we now take a peice of data and hash that data, to produce a fixed length key and encrypt that hash with our private key, anyone can decrypt the encrypted hash and then hash the data themselves and confirm it matches the hash that was decrypted. This confirms the piece of data passed to you is in fact cryptographically guaranteed to be the peice of data the signature refers to, otherwise the decrypted hash would be different in which case the data is not in fact signed (or hashed and encrypted) by the person with the private key that matches the ID of the ID who claimed to sign the data.

This is digital signing (albeit with some detail missing) as it's simplest. This is as detailed an understanding as is required to manage this paper.

[1]Here we assume the perfect algorithm and implementation of such, this is unlikely to exist in perfection.

[2]you can see where the problem exists as the logical conclusion is a key longer than any known peice of uncompressable data, to ensure no collisions

Algorithm 1 Hash(Public key) == ID

Algorithm 2 Hash(Public key + Signature) == ID

II. Implementation

A. Methodology

This validation system requires, as all cryptographically secure validations systems do today, manipulation of key pairs. These key pairs are themselves used to generate the identity rather than being later tied to an identity and this is one of the fundamental tenets of this paper. The system this paper proposes will operate equally well with databases, DHT's or any key adressable storage system. In some cases this system will work outwith such key adressable systems, although in a reduced fashion.

B. Simple Configuration

In a relatively simple method, identities are created as follows: $Hash(Kpub_1) = Identity$. This strictly ensures the identity is mathematically linked to an identity rather than being later tied to one.

C. Addition of a key revocation method

In another small step we can introduce the ability for a revocation system. this is done as follows: $Hash(Kpub_1 + SignatureKpriv_2) = Identity$. For key revocation to operate the identity packets should be stored in a way that retains all information as mutable unless the signer of the data requests amendment or deletion. In this case deletion is not recommended and replacement of the public key with a false key (all 0's for instance) would be identified quickly as a revoked key.

Alone this would not seem to make sense as we generally wish to have a chosen identity and then secure this with a validation system. Although in some cases the system would be a good enough method, systems such as telephone numbers or bar-codes on product etc. may be facilitated by the system in this simple state.

D. Passing ID between parties

Whereas the ID packet may be stored in any key adressable database or network as previously stated, there does exist a mathematically secure mechnism for doing so. This is possible as we have introduced a secure cryptographic hashing

Table I

$Kpub_1$	Sig_{Kpriv_2}	$Kpub_2$	Sig_{Kpriv_2}	Payload

mechanism which allows us to ensure that it is unliley that we can produce two peices of data that will hash collide.

With this in mind and the fact that the ID packet described in II-C is the hash of the content of the packet, in this case a public key and an (optional) signature, which we now know is another hash encrypted. Consider a message formatted as shown in Table 1.

In this scenario the message is recieved and is allegedly from an ID. The first 2 fields represent the ID packet that would be stored in the key addressable storage mechanism decribed earlier. Here though we can simply hash the first two fields and confirm the hash matches the ID we think the message is from. If the message is also signed then this is conformation this message is valid. If there is no signature, a valid response would be to to take the public key in field 1, encrypt a message or challenge back to the sender, or indeed simply encrypt the reply the sender requested from you using this key.

It is therefor possible to validate an identity even without a key adressable network as long as all parties send the header fields required to identify themselves in the first place. The addition of key adressable storage though allows better key revocation in cases werhe $Kpriv_1$may be compriomised, if it is stored on acessable media (such as a hard drive somewhere). The most efficient schemes would allow checking key validity on ocassion and this is possible in two ways.

E. Key validity checks

We can easily tell if a key is valid cryptographically using either of the schemes previously decribed, however, there are situations where a cryptographically valid key is not in fact valid from our perspective and this is when, for whatever reason, a key has been revoked. In such cases the system requires to be cryptographically secure in the revocation scheme being used and also has to ensure key validity is maintained and revoked keys no longer get used.

In the scheme described in II-D validity is chacked but currency of the key is not and it wil remian cryptographically valid forever!. Unless a check is somehow introduced into the system to tell if a key is currently valid as well as cryptographically valid, then there is room for abuse. Therefor a revocation scheme that can be bypassed would be next to useless.

There are two methods in this paper of ensuring a key is currently valid and these are as follows:

1) Check validity by looking up key: The valiaidy of a key can be confirmed by forcing a lookup of the key in the key addressable storage medium. A mid way measure may be that keys passed in messages are tested for vaidity at random. Either way would remove invalid credentials in a balance between speed and efficiency, with the forced lookup being the most efficient form a security perspective but perhaps too slow for some implementations.

Algorithm 3 if (Hash(Public key + Signature) == Unique) then Hash(Public key + Signature) == ID == Network address

2) Check validity by message passing: This section is a little involved and we need to step back a little and look again at the structure of the key pairs in the double checked message as shown in Table 1. It is notable in this case key pair 2 must be valid as they sign key pair 1, therefor it follows if key pair 2 is invalid then key pair 1 is automatically invalidated[3].

Taking this into account we can clearly see the chain of validity in this case. The chain being a set of two key pairs.

In the case of message passing the message should contain both ID packets as per the table. The validity of key 2 is then tested to confirm key 1 is valid. This does make sense in the design so far where key 2 is never available on a disk or anywhere that can be placed in a position of mistrust. To do so we simply encrypt a message back using ID2's public key and the sender should be able to answer and / or sign a response. This does though assume the request being made is a request that has come from an enitity that is at a higher level than the key1 holding node alone. So this mechanism is only valid for higher level messages and not the lower level messages or actions that are restricted to the node level entity.

Therefor in this implementation the message passing checking mechanism is only effective with a certian message type and therefor in a closed network or system where message types are in fact known and/or able to be identified.

The maidsafe[4] network can provide such a system as it depends on a node that can be identifeid and behave, building rank which a user (who has $Kpriv_2$) can use as the owner of $Kpriv_1$thereby getting the benefits of any quid pro quo relationship with a node and the persons rank on the system. In this case the person can easily revoke and recreate the key transferring any rank across to the new key.

F. Combined with DHT

DHT networks have a particular requirement in common and this is to create a network address that is unique. Many implementations use various techniques for achieving this such as chord[2] which uses the hash of the IP/PORT combination, Kademlia[3] uses a random hash and so on. Using the above method and actually storing the identification packet on the network can provide for uniqueness and at the same time a mechanism for finding a nodes public key to send information encrypted or to check a signature.

This is implemented as follows: $netget[Hash(Kpub_1)]$to check for uniqueness and if false then the true keys can be stored as follows: $netput[Hash(Kpub_1)/[Kpub_1]$ this is the simple case above without revocation, simply for brevity.

The hash size chosen should be of the same length as the network addressing scheme in place for completeness.

[3]This must be true otherwise we are saying key pair 1 is signed by an non existant or invalid key pair 2

III. Improvements with security of private keys

If such a system were implemented in a DHT or similar publically connected system (as are web servers), then the issue over the security of the private key is paramount. In today's networks this is achieved via some brute force techniques such as firewalls, secured hard drives, private key passwords (which make automatic reboot of a server require human intervention) and other custom approaches. There is no obvious improvement in this particular situation of this private key, however, the system presented does have flexibility that the others do not and this is in the ability to reduce the effectiveness or scope of the private key, $Kpriv_1$.

Here, it is assumed that $Kpriv_2$ is a key that is used to sign the identification packet for $Kpub_1$ and that this key is not located on the machine or node as $Kpriv_1$ has to be. The intention now would be to ensure that the node can only identify itself, but be limited in it's possible actions. In such a case the node acts for some client or person and this person takes action on the network as that node, perhaps as the node is a node that succumbs to some kind of ranking mechanism which allows the person using this system to operate in a particular sphere of reference as laid out by the rank or effectiveness of the node. In such cases the person would usually maintain the identity of the node and sign/decrypt messages with $Kpriv_1$ acting as the ID that $Hash(Kpub_1 + SignatureKpriv_2)$ provides.

A helpfull addition in this case is to consider the singnature key of this identifier packet. This can be identified by the person quite easily as shown in II-D.

IV. Conclusions

This paper introduces a particular case for 'non rooted' yet cryptographically secure PKI networks to be created, it is envisaged that this method will extend with many more capabilities. Such measures may include:

Idnetification of credit card data linking the ID to a known name in another secure location. People could have a card and a revocation card or perhaps even better all done in software using a keying approach as described in this paper.

Single continuous validation systems where a known ID can be used across multiple web sites or on-line systems that require history to operate effectivly.

References

[1] As described by Van Jacobson in this link below, August 30, 2006 HTTP://video.Google.com/videoplay?docid=-6972678839686672840

[2] Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan,Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, .

[3] Petar Maymounkov and David Mazirese Kademlia: A Peer-to-peer Information System Based on the XOR Metric {petar,dm}@cs.nyu.edu http://Kademlia.scs.cs.nyu.edu

[4] David Irvine, maidsafe - a new network paradigm. david.irvine@maidsafe.net

David Irvine is a Scottish Engineer and innovator who has spent the last 12 years researching ways to make computers function in a more efficient manner.

He is an Inventor listed on more than 20 patent submissions and was Designer / Project Manager of one of the World's largest private networks (Saudi Aramco, over $300M). He is an experienced Project Manager and has been involved in start up businesses since 1995 and has provided business consultancy to corporates and SMEs in many sectors.

He has presented technology at Google (Seattle), British Computer Society (Christmas Lecture) and many others.

He has spent many years as a lifeboat Helmsman and is a keen sailor when time permits.