Orthus Authentication Protocol: Background Services

Dean Rogers
Edinburgh Napier University
Edinburgh, United Kingdom

Abstract— **This paper discusses background security supporting services compatible with computer systems protected by the Orthus authentication protocol. A secure password change protocol for Orthus along with implications for User session duration time-outs and long running services is also outlined. Reference is given to the protocols adaptation to encompass spontaneous randomly generated elements by the use of hard or possibly software Tokens when combined with the static User defined password element. Further, a unique method of Identity Management based on Version 4 generated UUID, and a secure storage and retrieval scheme for Orthus keys are also described. Additionally discussion is given to an authorisation system which integrates with Orthus thus helping to provide a complete security system.**

Keywords—, authentication, identity management, key management, password change, Token, authorization.

I. INTRODUCTION

In the contemporary Internet of Things, it has become increasingly necessary to be concerned with the security implications accompanying connecting 'everything' to the Internet. No longer does involve just protecting confidentiality of records, but also preventing misuse of Internet enabled devises. And in an era of Big Data, there is the further risk that information leaked and collated from these devises could readily be used to compile a profile of a targeted person, and be subsequently used for malicious purposes.

To a degree precautions have already been taken e.g. the heart pacemaker of former US vice president Dick Cheney had the wireless capability disabled hindering potential abuse. Other devises such as insulin pumps may potentially also be open to similar abuse. In cases such as this the alarming prospect is that the sabotage could be indistinguishable from malfunction, and so remain undetected. The list goes on; wirelessly enabled emergency service vehicles could be prevented from fulfilling their duties, hacked security cameras or television sets providing burglars with vital information, electrical power systems maliciously shut down, or driverless vehicles hijacked. Small devises with embedded systems consequently do not have the capacity to incorporate large software systems such as anti-virus and firewalls for protection.

Computerised systems located within institutional boundaries have a degree of protection afforded by the usual corporate grade firewalls, Intruder Detection Systems, and anti-virus and malware prevention systems. The degree of sophistication employed by hackers is ever increasing, organised crime and malicious national governments have thrown their resources into the illicit penetration of targeted systems, for example the December 2016 closing down of the Ukrainian Power Grid system was the first confirmed attack of this kind. It proceeded in several stages, the first being reconnaissance of the corporate network, secondly executives were spear phishing targeted to insert malware, thirdly credential theft via the malware, fourthly malicious firmware was implanted in the control systems of the power grid, etc.[1] Taking control of computer systems commonly involves logging in as an authorised user, perhaps by finding a backdoor, compromising the authentication system, facilitated by malware aided password theft or cracking.

Enterprise wide imposed password complexity and renewal policies greatly enhance the prevention of infiltration, however the end result only encourages users to write down their passwords, and does little to prevent identity, login and password, sharing. Ideally the User entered password should be unique upon each entry. Naturally the typical User would not accept the inconvenience of changing their password upon each login, after all systems must remain usable.

II. ISSUES ADDRESSED

A seldom used solution involves simultaneously combining the relatively static User generated password together with a spontaneously generated random element, effectively meaning that each login consists of a unique identifier. This may be accomplished via the use of hard or software Tokens. Secure registration of a token to an individually identifiable user is integral to successful security; however that process is not the concern here, just facilitation of the use of Tokens, and that the implementation be Orthus compliant.

This paper essentially presents the design and concept of various background services necessary for the functioning of the Authentication within Realm environments. Consideration is given to how static and dynamic elements mentioned above can be incorporated into the Orthus protocol authentication process, but the principle could be readily adapted to other authentication systems such as Kerberos [2]. This paper continues by discussing related issues such as a password change protocol, and key management issues relating to the Orthus protocol, including secure storage and retrieval. It is not the intention here to repeat previously published material, a full working description of the message flow in Orthus may be found in earlier publications [3], and [4]. Further, the

management and transition of identities over time, while preserving authorisation and subsequent access rights, compatible with Orthus is considered. As is an authorisation system capable of integrating with Orthus.

III. The Orthus Authentication Protocol

Internally the Orthus authentication protocol operates differently to Kerberos, following successful authentication by Kerberos the end result is the issue of a security key to the client for their selected service, see previous publications. Having been based on the earlier Needham Schroeder [5] protocol Kerberos has been updated [6] but retains the fundamental architecture.

Essentially after authentication with Orthus the result is a security ticket verifying realm membership only, without access to any services. Orthus, like Kerberos relies on a DNS [7] infrastructure. Further, Orthus requires integration with a compatible service-side authorisation system [8]. The internal messaging mechanisms of Orthus are depicted in Fig. 1.

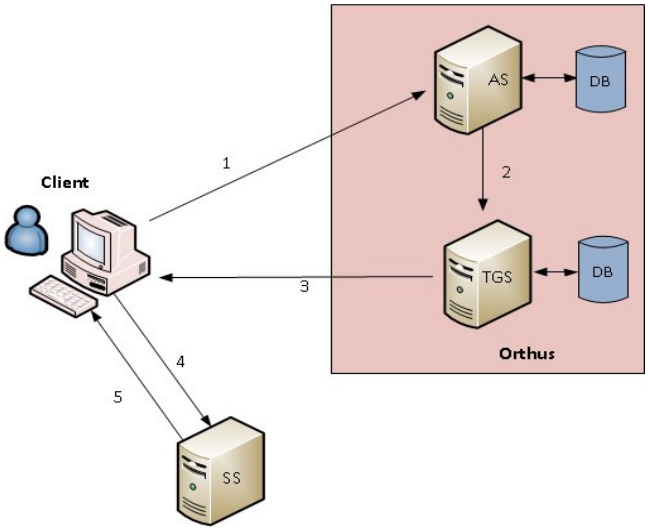

Fig. 1 Orthus Architecture

IV. Adaptation of Orthus for use with Tokens

Once we consider incorporating spontaneous random elements into the "entered" password to assist in preventing the re-use of captured or cracked passwords, methods of combining these static and dynamic elements must be included in the design. Usage of synchronised Tokens is not novel and they can greatly enhance security. At present commonly used are software based tokens for iOS or Android Apps, but hardware Tokens could also be included, discussion of the variation in implementation would constitute a different topic. The fundamental architecture mandates the random number generated by the Token is identically synchronised with the server side of operations, the mechanism of which does not concern us here.

Typically in use, an encryption key based on the users entered password is generated, together with a separate one for the spontaneous random element. Although the user enters one concatenated string (by convention in the order PWD + Random), a character count of the previously entered password is retrieved from the local store (Stored in itself in encrypted form, the key to which is freshly generated upon each password change, and only accessible with 'system' privileges), and used as a subtraction template to separate the elements. A Users initial realm login, or login on a new client, must be with their password only; thus generating the character count on the new client. The User will receive an error message, and on re-try enters the combined string. Off-line login, i.e. locally authenticated when realm authentication is achieved over a VPN connection, is facilitated with the User logging in as usual with both password elements, and performs this again over the VPN, because the spontaneous element would no longer be synchronised were it to be buffered and passed through to the VPN software. The server-side encryption key created from the generated synchronous value which was made spontaneously, and is discarded after use. It should be noted that attempting to concatenate the user entered password and spontaneous element to form a new key would not be successful, as the AS decrypting the users' stored key cannot recover their password for server side concatenation with the synchronous random element: meaning no comparison of transmitted elements can be achieved in this case.

Orthus Message 1 transforms from the original

$$A \rightarrow AS\text{: } A+R_A+TGS+N_1+Tf$$

Where: *A is the clients' realm name*
AS is the Authentication Service Realm name
R_A is the Realm name the client wishes to join
R_{TGS} is the realm name of the Ticket Granting service, N_1 is a nonce
Tf are client options, from(start) till(expiration, or time-to-live) rtime (renew till time request)

To become

$$A \rightarrow AS\text{: } A+R_A+K_R(R_{TGS}+N_1+Tf)$$

Where: *K_R is an encryption key from the random element*

Message 2 becomes

$$AS \rightarrow TGS: A+R_A+K_{TGS}(K_A+A+R_A+Tf)+K_A(R_{TGS}+TGS+N_1+Tf)$$

Where: *K_{TGS} is the encryption key of TGS*
K_A is the clients' encryption key derived from their password as previously described
R_{TGS} is the realm of TGS

Note the AS can spontaneously compute K_R from the synchronized value; and that K_{TGS} has been pre-shared out of

band between AS and TGS. The value of K_A has been retrieved from the AS database (see below).

Message 3 becomes

$$TGS \rightarrow A: A+R_A+K_R(K_{ASS}+A+R_A+R_{TGS}+TGS+N_1+Tf+K_{SSU})$$

Where: *K_{ASS} is a session key generated by the TGS for the sole use of A and SS*
K_{SSU} is the encryption key for realm authentication

The essential difference of the Token enabled version of Message 3 is that the payload is now encrypted with K_R instead of K_A. The clients' version of K_R should match that generated at the AS and it can successfully decrypt the payload. These amendments to the message architecture do not affect the structure of the Password Change Protocol, PCP, (see below) but do have implications for the authentication indicated protocol below, meaning that the authentication system needs to be programmed to be switchable at the Realm level, between 'plain' or 'Token enabled'. While this may appear straightforward for the Authentication Services, implicitly a trigger needs to be cascaded to all realm clients informing them of which key to use a key for encrypting message 1, and when decrypting the payload of message 3.

This trigger item is effective upon boot when the host machine account joins the realm as a down loadable system update, and provides for clients that have not connected in while to still be able to connect successfully. Implementing a 'mixed mode' is not be recommended as it would contravene security provisions.

The rest of the Orthus Protocol, messages 4 and 5, remain unchanged to the previously published

Message 4

$$A \rightarrow SS: A+O+K_{SSU}(K_{ASS}+R_A+A+Tf_1)$$

Where: *$Tf1$ represents a new set of client options relative to SS channel*
O represents options that can be requested by A

Message 5

$$SS \rightarrow A: K_{ASS}(T_{S2}+S_K+SQ_{NO})$$

Where: *T_{S2} is a new timestamp*
S_K is a flag requesting a new sub-key
SQ_{NO} is a sequence number generated by SS

V. Password Change Protocol

The availability of a universal password change protocol for e.g. the widespread Kerberos Authentication system remains in the balance at the time of writing. From 1998 onward there is a history of draft protocols within the IEFT [9], and [10], and subsequently various Microsoft collaborative Drafts, the latest of which without formally proceeding to full IETF RFC status [11], expired on May 7 2009 [12] and there is also a Microsoft Windows specific RFC, the last of which dates from 2002 [13].

Clearly within the Orthus Protocol family suite there is a perceived need to establish a method of enabling the secure change their passwords by their User owners, either because of enforced realm policies, perceived threats or personal preference. In the nature of Orthus a User may change their password mid-session or during the login process without undue consequences. Once admitted to the realm, access to services is no longer password dependent, but relies on an authorisation negotiation between client and service. This implies that access to e.g. the email server would not be interrupted mid-session due to password expiration, but only be effective on the next login when User would expect an appropriate prompt. The alternative of automatically ending the session and automatically logging the User back in with the new password would be of little benefit.

This has consequent implications for **long running service** accounts needing to remain logged in for extended periods and be exempt from enforced password changing policies. This may be achieved with an entity-type flag exempting that entity from password policies except those concerning complexity. The implication here is that for security reasons a User **session time-out** must be implemented, a compromise of reasonable duration allowing a User's normal work but which prevents leaving a session running to avoid enforced password re-sets. Ten hours may be sufficient, but the standard should be realm administrator configurable.

1. $A \rightarrow AS: A+R_A+AS+N_1+Tf$

This differs from Orthus' first message by the inclusion of the AS identity rather than TGS, acting as a trigger interpreted by the AS as a password change request.

2. $AS \rightarrow A: A+R_A+K_A(K_{ASS}+A+R_A+N_1+Tf)$

This is a different K_{ASS} session key as issued during the authentication process.

3. $A \rightarrow AS: K_{ASS}(K_{AO}+K_{AN}+N_2+Tf_2)$

The session retrieved is used to encrypt the old and new client keys together with new session information.

4. $AS \rightarrow A: K_{ASS}(R_A+N_2+Tf_2)$

This is the AS confirmation to the client of password change.

The client has already provided all information required and the next message in sequence is analogous to Orthus authentication message 2 except the clients' new encryption key is used.

VI. Key Management

Since the realm administrator has neither control nor prediction of the names of Users engaged within the enterprise, and the User name is fundamental to the Login process, even though as indicated above here it is the UUID that is the core

entity, a policy still needs to be implemented dealing with name clashes such as the legitimate existence of e.g. three Gloria Jones's within the enterprise, but they are now not so critical.

This paper is not concerned with the management of asymmetric keys such as used in PKI [15] systems, rather the storage and retrieval of symmetric keys, as preferred for their performance, and employed during Orthus authentication. The key generated from a user's password being presumed unique to that user is the basis for the security paradigm. For reconstruction purposes the key is stored, not a hash of it. As in the event of the AS being breached, this constitutes a potential security risk, security is enhanced with these keys in turn being encrypted with a 'master' AS key. This master key must be stored in a separate but readily accessible location, which is set, and only accessed, by Orthus with 'system' level permissions.

Relevant information is stored in a dual table database. Example values of UUID and key are given for illustrative purposes and should not be taken as representative.

Logon	UUID	TS (*ms*)
Joan Smith	xLighjk	Yyyy:mm:dd:mm:ss:ms
John Jones	3Qwert	Yyyy:mm:dd:mm:ss:ms
Joan Evans	xLighjk	Yyyy:mm:dd:mm:ss:ms

Table 1. Table A

Table A allows for changes of Login e.g. from Joan Smith to Joan Evans, and therefore for multiple entries of the corresponding UUID associated with this person. Should company policy dictate that this only be allowed to persist for a predetermined transition period, then it is a trivial matter to remove the older duplicate at a later date. Identity management is preserved, because although a many to one mapping for Login to UUID is possible, it is not possible for the inverse to occur because of the uniqueness of the UUID, enforced independently for this table. E.g. after the original Joan Smith login is removed the company may employ a second person of the same name at a later date, but that person would automatically receive a different UUID. Denotation of year by four digits avoids potential turn of centaury problems.

VII. Identity transition management

Hostnames have to be unique within a realm, however it is not unreasonable that for various reasons a name collision could occur, e.g. by a host transfer between realms, thus necessitating inconveniently re-naming either of the Hosts. Users get married or may change their name by other Legal registration means. The use of Universally Unique Identification numbers, UUIDs, is recommended, to accommodate such scenarios', as an attribute of the named entity. However, the intention here is that an individual UUID 'represents' the entity, and further aspects are added as attributes of the entity. Therefore, the UUID should not be based on properties of the named object, but rather derived from a Version 4 random generation method [14]. In this way, changes to a User or Host name can be readily facilitated, with the UUID retaining attributes of both old and new names (including for audit purposes an incontrovertible history function detailing name changes associated with the UUID), and maintaining other access permissions or privileges.

The possibility must remain for Joan to revert to her former name and pickup her previous access at a later date. A method that prevents a malicious administrator from assigning the old account to a completely different Joan (which would of course involve legal consequences), or even to a completely different person (possibly for malicious reasons) would be difficult to devise. To provide a degree of accountability all entries need to be time-stamped and recorded in a 'system' privilege write only field, with read access for other services.

UUID	Name	Key
xLighjk	Joan L Evans	0x678bnmPas
3Qwert	John A Jones	1x987Dyxcvb

Table 2. Table B

Table B stores each field of UUID, actual User name, security key are unique values, but for these purposes, the Name and Key fields are re-writable and not suitable for use as Table Primary Keys. Only the UUID is time independent, in the sense of being non re-writable, and subsequently suitable for the enforcement of referential integrity and use as a primary key. It remains possible to remove redundant records from the 'live' system to an archive.

The system functions because referential integrity is enforced on the Login field of Table A, when John Jones logs in his UUID is found from Table A, and then used in Table B to retrieve his key.

In the event that either the login name supplied or the key generated from the password not match that retrieved from the store, a suitable error message is returned to the User. A timed read access counter enforces a login re-try lockout policy; several bad attempts resulting in temporary disablement of the login procedure for the entered User name. Upon password reset the previously associated key would be removed to a history field, the number of fields depending on the history policy – these fields are checked to enforce that policy.

Where for legal reasons a User's previous names need to be preserved it is a trivial matter to provide similar history functionality derived from Table B, which for example in either case could be implemented as a separate archive table, thus providing an audit trail of those Identities associated with the UUID over time. Thus, a person could change their Sur/Family name several times during their employment history while seamlessly retaining their assigned access rights.

VIII. Authorisation in the Orthus Environment

Processing the appropriate authorisation level for an authenticated member begins at the SS with receipt of Orthus Message 4.

$$A \rightarrow SS: \; A+O+K_{SSU}(K_{ASS}+R_A+A+Tf_1)$$

Where: Tf_1 represents a new set of client options relative to SS channel
O represents options that can be requested by A

Upon receipt of this message the SS can determine the client identity, A, and that it resides in the same realm as itself R_A – this it has to determine by using its copy of K_{SSU} for the encrypted components. The identity "A" which is transmitted is the UUID, in other words the fundamental entity to which attributes are added such as a User Name, fig. 2.

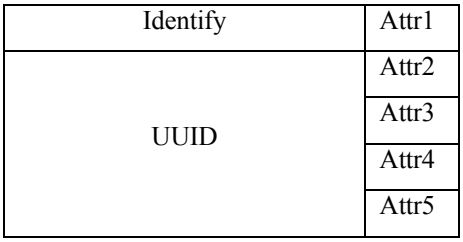

Fig 2. Identity Item

Attribute 1 would be the Object Name which cannot be null, Attribute 2 its email address where null is permissible, Attribute 3 a binary on/off flag for 'services'. The number of attributes is not fixed, and may be added to as appropriate. The user entity cannot carry with it all of its permissions at different levels, this would be impractical and would lead to extremely long access times while the attributes were read.

The onus is on the article "J" to maintain a customizable access control list, ACL. Each file carries an ACL, for non-executable files that part is mute. When a User attempts access to a Service the relevant executable file reads their ID and after lookup the Service knows what the User is allowed to do after reference to its locally maintained ACL

Table 3 below shows a typical Access Level List

Access level	Server Level	Description
0	No access	Access Denied
1	Minimal	List, files not services
2	Log reader	Read files, folders, list services
3	User	Modify Files
4	Advanced User	Create, Delete, Modify files
5	Backup Ops	Execute, List, Copy
6	Junior Admin	User accounts, Printers (not servers)
7	Server Admin	Full access individual server
8	Realm Admin	Junior Admin + Realm controller + add remove servers
9	Customisable	As needed

Table 3. Access Level List

Realm administrators who need access to several machines can be accommodated via groups, whereby an attribute can be set specifying directly the access level. Likewise a Server administrator, without Realm privileges, can be accommodated via customisable groups.

Below, Table 4. Illustrates a typical Access Control List

Entity	Access Level
Xxy01	0
Y02zz	4
0zy2xx	8

Table 4. Access Control List

In this way the appropriate level of access for an individual or service can be determined without causing 'token bloat' as is often witnessed in other authentication systems. The interesting feature here required for integration with Orthus is the nominal access authorisation level of 'zero'. Every Service requires the authenticating entity to present its access level, and it is possible that upon successful authentication that nothing can be accessed. This is a feature, not a bug.

IX. Summary

Many security breaches have been publicised recently, and it has become clear that it has long been time that greater use of two factor protection was made, instead of traditional 'simple' password authentication protocols. This paper presented the design for integrating two-factor authentication via the use of randomly generated numbers from hard or software Tokens, into the Orthus protocol. This greatly enhances the use of password technology as the access challenge method to computer systems, and in the process impedes identity/password sharing among Users.

A secure password change protocol for use with Orthus has been outlined. The need for a systematic means of organizing key storage and retrieval is integral to the whole secure network ideology, and a secure key storage management scheme was also outlined. This topic led quite naturally onto managing the changing identity of user/entities over time which and is problematic with other authentication schemes and can be a cause of great concern to enterprises administrators. Here, a novel means of Identity Stipulation leveraging Class 4 UUID was introduced to facilitate enhanced User/entity tracking. This concept can potentially find application in other authentication systems.

Finally, rounding out the Orthus landscape, with the objective of secure access to files or services within the enterprise, an Orthus compliant Authorisation system was described.

X. SUGGESTIONS FOR FURTHER RESEARCH

The Orthus landscape presents an integrated map of compatible protocols for ensuring secure Authentication and authorisation. Recent developments in biometric authentication have rekindled interest in that topic. The design of a biometric frontend for Orthus remains on the drawing board. It is further anticipated that research could reveal the adaptation of the protocol to other authentication uses.

XI. CONCLUSION

There are a multiplicity of security protocols deployed and/or under development to help protect computerised information networks and systems, e.g Secure DNS [16] and Host Identity Protocol [17] etc. All of these add depth to security protection, but many attacks are perpetrated via the capture of identity information through malware. Virus and malware filters are always playing catch-up with the newest released variations. Once unwittingly installed in computers Malware often contrives to elevate its privileges to administrative level, and can be programmed to disable incumbent anti-virus systems. The design rational behind Orthus was to provide a simpler authentication system making it easier to maintain, and consequently because of the comparatively fewer message exchanges involved better performing.

Heightened vigilance via user education remains a necessary defence against such attack. Users have to be constantly aware of the risks involved in opening links in unsolicited emails, and that emails can be forged to appear legitimate. This also applies to web-sites and web-applications users may visit, internal or external. However even the most attentive users are human, and vulnerable to error. Relying on user care amounts to a numbers game; at some point, an organisation may become infiltrated. Isolation from the general Internet, as with some Government and Military networks, is one answer. For other organisations spam and virus filters on email, together with both white & black Internet access lists are facts of daily life.

Password protection is a vital aspect of security in depth which must be integral to our safety. The usage of passwords shows little sign of diminishing in the near future. Together with the previously published Orthus supporting services, this family of Orthus compliant protocols has the objective of making such malicious penetration more difficult. A further part of the focus in devising the Orthus landscape was to alleviate some of the common problems encountered in Realm maintenance and administration providing secure yet simpler and easier to maintain systems, as was the objective of the identity management scheme presented.

APPENDIX A - RELATED RESEARCH

Various alternative authentication protocols exist, and are in continual development. One such is known as Central Authentication Service [18], CAS, and was developed at Yale University by Shawn Bayern in 2004, the specification being published in 2005. This, however, is browser based and relies on use of the https protocol to secure message transfers. Another is RADIUS, which is undergoing continual development by an IETF (Internet engineering Task Force) working group [19].

REFERENCES

[1] P. Fairley, "Cybersecurity at US Utilities due for an Upgrade", IEEE Spectrum, pp 07 – 09, 05/2016.

[2] J. Steiner, C. Neuman, and J. Schiller, "Kerberos: An Authentication Service for Open Network Systems," *Usenix Conference Proceedings*, Dallas, Texas, pp. 191-202, February 1988.

[3] D. Rogers, "Introducing Orthus, a Dual Headed Authentication Protocol," *International Journal of Signal Processing Systems*, pp. 153-158, December 2015.

[4] D. Rogers, "Orthus v2 Authentication Protocol Enhancement, and supporting Enterprise Architecture," *Procedia Computer Science*, Volume 63, pp. 581-588, October 2015.

[5] R. M. Needham, M. D. Schroeder, "Authentication in Large Networks of Computers," ACM Vol 21 No 12, December 1978.

[6] B. C. Neuman, T. Yu, S. Hartman, and K. Raeburn, "The Kerberos Network Authentication Service (V5)," RFC 4120, July 2005. http://tools.ietf.org/html/rfc4120.

[7] J. Postel, "Domain Names Plan and schedule", RFC 881, November 1983, http://tools.ietf.org/html/rfc7230.

[8] R. S. Sandhu, and P. Samarati, "Access Control: Principles and practice," *IEEE Communication Magazine*, pp. 40-48, September 1994.

[9] M. Horowitz, "Kerberos Change Password Protocol", Internet Draft, August 1998. https://tools.ietf.org/html/draft-ietf-cat-kerb-chg-password-02

[10] M. Swift, J. Trostle, J. Brezak, B. Gossman, "Kerberos Set/Change Password Version 2" Internet Draft, March 2001. https://www.ietf.org/proceedings/50/I-D/cat-kerberos-set-passwd-04.txt

[11] J. Trostle, M. Swift, J. Brezak, B. Gossman, "Kerberos Set/Change Password: version 2", Internet draft. https://www.ietf.org/proceedings/53/I-D/draft-ietf-cat-kerberos-set-passwd-06.txt

[12] N. Williams, "Kerberos Set/Change Key/Password Protocol Version 2", Internet Draft, November 2008. https://tools.ietf.org/html/draft-ietf-krb-wg-kerberos-set-passwd-08

[13] M. Swift, J. Trostle, J. Brezak, "Microsoft Windows 2000 Kerberos Change Password and Set Password Protocols", February 2002, http://tools.ietf.org/html/rfc3244.html

[14] P. Leach, M. Mealling, R. Saltz, "A Universally Unique Identifier", RFC 4122, July 2005, http://tools.ietf.org/html/rfc4122

[15] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM* 21 volume 2, pp. 120–126., February 1978

[16] A. Arends, et al, "DNS Security Introduction", RFC 4033, March 2005, http://tools.ietf.org/html/rfc4033

[17] R. Moskowitz, P. Nikander, "Host Identity Protocol Architecture", RFC 4423, May 2006, http://tools.ietf.org/html/rfc4423

[18] D. Mazurek, et al, "CAS Protocol," May 2005, http://www.jasig.org/cas/protocol, accessed 26 January 2014.

[19] RFC 6614, "Transport Layer Transport Layer Security (TLS) Encryption for RADIUS," S. Winter, M. McCauley, S. Venaas, K. Wierenga, 14 May 2012 (Experimental).