ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αρχιτεκτονική Ασφαλείας για Κατανεμημένες Εφαρμογές Πλέγματος Ευαίσθητες σε Χρόνο Απόκρισης

A Security Architecture for Distributed Grid Applications with Sensitive Delay Requirements

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ
DOCTORAL DISSERTATION

Αθανάσιος Μώραλης του Μιχαήλ
Athanasios Moralis

Αθήνα, Μάρτιος 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αρχιτεκτονική Ασφαλείας για Κατανεμημένες Εφαρμογές Πλέγματος Ευαίσθητες σε Χρόνο Απόκρισης

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Αθανασίου Μώραλη

Συμβουλευτική Επιτροπή: **Β. Μάγκλαρης**
Ε. Συκάς
Μ. Θεολόγου

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την

Β. Μάγκλαρης Ε. Συκάς Μ. Θεολόγου

Ι. Βασιλείου Σ. Παπαβασιλείου Ε. Χατζηευθυμιάδης

Δ. Καλογεράς

Αθήνα, Μάρτιος 2011

.....

Αθανάσιος Μώραλης του Μιχαήλ
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η αρχιτεκτονική ασφαλείας που προτείνεται στην διατριβή είναι προσανατολισμένη στο να δώσει λύσεις σε κατανεμημένες εφαρμογές Πλέγματος (Grid) ευαίσθητες σε χρόνο απόκρισης. Πεδίο εφαρμογής της είναι τα Πλέγματα Μετρήσεων και Ελέγχου (Instrumentation Grids), επέκταση των Συστημάτων Υπολογιστικού Πλέγματος (Computational Grids). Τα Instrumentation Grids είναι κατανεμημένα συστήματα τεχνολογιών πλέγματος, τα οποία προσφέρουν απομακρυσμένη πρόσβαση σε όργανα «μετρήσεων και ελέγχου». Βασική προδιαγραφή στα περιβάλλοντα αυτά αφορά στον περιορισμένο χρόνο απόκρισης των κατανεμημένων λειτουργιών των οργάνων. Η αρχιτεκτονική ασφαλείας που υιοθετείται στα παραδοσιακά Grids βασίζεται σε Υποδομή Δημοσίου Κλειδιού (PKI) και δημιουργεί συνήθως μεγάλες καθυστερήσεις. Για τον λόγο αυτό προτείνουμε μια νέα αρχιτεκτονική ασφαλείας βασισμένη στο πρωτόκολλο Kerberos. Η αρχιτεκτονική μας αποτελείται τέσσερα υποσυστήματα: Το Kerberos KDC που αποτελεί την Τρίτη Έμπιστη Οντότητα του συστήματος, το KrbClient που αλληλεπιδρά με την αρχιτεκτονική εκ μέρους του λογισμικού πελάτη, ο Access Control Manager (ACM) που προστατεύει τις υπηρεσίες ελέγχου του Instrumentation Grid και το Policy Repository που αποθηκεύει τους κανόνες πρόσβασης (authorization) των κατανεμημένων πόρων. Στα πλαίσια της διατριβής υλοποιήθηκε πρότυπη αρχιτεκτονική ασφαλείας βασισμένη σε ενδιάμεσο λογισμικό Υπηρεσιών Ιστού (Web Services). Οι μετρήσεις που πραγματοποιήσαμε επιβεβαιώνουν πως η προτεινόμενη αρχιτεκτονική βελτιώνει σημαντικά την απόδοση στις λειτουργίες ασφαλείας (Ταυτοποίηση, Εξουσιοδότηση και Ακεραιότητα Μηνύματος) αντικαθιστώντας την Κρυπτογραφία Δημοσίου Κλειδιού των Computational Grids με Συμμετρική Κρυπτογραφία. Η βελτίωση γίνεται εντονότερη κάτω από υψηλό υπολογιστικό φορτίο στις υπηρεσίες μετρήσεων και ελέγχου των οργάνων.

Abstract

In this thesis we propose a security architecture that provides a solution to distributed Grid applications with delay sensitive requirements. These applications are usually encountered within an Instrumentation Grid, an extension of a Computational Grid. Instrumentation Grids are distributed systems that follow Grid technologies to interact with remote distributed instruments. One of their main requirements is low latency times. This may not be satisfied with security architectures adopted in Computational Grids based on Public Key Infrastructure (PKI). For that reason we propose an alternate security architecture based on the Kerberos protocol. Our architecture is composed of four components: The Kerberos KDC that acts as a Third Trusted Party, the KrbClient that represents the client, the Access Control Manager (ACM) that protects the services of the Instrumentation Grid and the Policy Repository which stores the access rules (authorization) of the distributed resources. We validated our architecture by implementing a prototype built on Web Service middleware. Our experiments showed a significant performance improvement in authentication, authorization and message integrity processes over legacy Grid security architectures. The improvement becomes substantial under heavy processing load of the remote instrumentation services.

Πίνακας Περιεχομένων

1. Εισαγωγή....12
   1.1. Περιγραφή του Προβλήματος....12
   1.2. Προτεινόμενη Λύση....13
   1.3. Παρουσίαση Περιεχομένων....14
2. Computational και Instrumentation Grids....16
   2.1. Εισαγωγή....16
   2.2. Computational Grids....16
      2.2.1. Ορισμός Grid Computing....16
      2.2.2. Εισαγωγή στα Computational Grids....17
      2.2.3. Δομή Computational Grid....19
   2.3. Instrumentation Grid....21
   2.4. Αρχιτεκτονικές SOA και Web Services....24
      2.4.1. Ορισμός των SOA....24
      2.4.2. Ορισμός της Υπηρεσίας σύμφωνα με τα SOA....25
      2.4.3. Πρότυπες Αρχιτεκτονικές SOA....27
   2.5. Web Services....27
      2.5.1. Τα Είδη και η βασική Στοίβα Πρωτοκόλλων Web Services....28
      2.5.2. Web Services και Grids....31
3. Αρχιτεκτονικές Ασφαλείας....33
   3.1. Εισαγωγή....33
   3.2. Λειτουργίες Ασφαλείας....34
   3.3. Είδη κρυπτογραφίας....35
      3.3.1. Κρυπτογραφία Συμμετρικού Κλειδιού....36
      3.3.2. Κρυπτογραφία Δημοσίου Κλειδιού....38
   3.4. Πρωτόκολλα και Συστήματα Ταυτοποίησης....41
      3.4.1. Υποδομή Δημοσίου Κλειδιού (Public Key Infrastructure - PKI)....42
      3.4.2. Kerberos....45
      3.4.3. Ασφάλεια σε Computational Grids....49
         3.4.3.1. Ταυτοποίηση (Authentication) στα Computational Grids....49
         3.4.3.2. Εξουσιοδότηση (Authorization) στα Computational Grids....51
   3.5. Πρωτόκολλα Ασφαλείας των Web Services....52
      3.5.1. Web Services' end-to-end Security....53
      3.5.2. WS-Security....54
         3.5.2.1. WS-Security Core....55
         3.5.2.2. WS-Security X.509 και Kerberos Token Profiles....56
      3.5.3. XML Encryption....58
      3.5.4. XML Digital Signature....59
      3.5.5. WS-SecureConversation....60
      3.5.6. WS-Policy και WS Trust....61
4. Προτεινόμενη Αρχιτεκτονική....63
   4.1. Εισαγωγή....63
   4.2. Απαιτήσεις της Προτεινόμενης Αρχιτεκτονικές....63
   4.3. Παρουσίαση Αρχιτεκτονικής Ασφαλείας....67
   4.4. Ανάλυση Αρχιτεκτονικής....72
      4.4.1. Η λειτουργία σε βήματα....72
      4.4.2. Παρουσίαση των τμημάτων της αρχιτεκτονικής....75
         4.4.2.1. KrbClient....76
         4.4.2.2. Kerberos Key Distribution Center – KDC....77

4.4.2.3. Access Control Management - ACM....77
4.4.2.4. Policy Repository....80
4.4.3. Ταυτότητες και Έλεγχος Πρόσβασης (Authorization)....81
4.4.4. Μηχανισμοί Αντιστοίχισης (Mapping Mechanisms)....83
4.4.5. Υποστήριξη Αντιπροσώπευσης Διαπιστευτηρίων....85
4.5. Ένταξη προτεινόμενης αρχιτεκτονικής ασφαλείας σε ένα γενικό πλαίσιο διαλειτουργικότητας....87
5. Υλοποίηση Αρχιτεκτονικής....92
5.1. Τεχνολογίες που χρησιμοποιήσαμε....92
5.2. Handlers....94
5.3. Υλοποίηση του ACM....95
5.4. Υλοποίηση του Kerberos KrbClient API....99
5.5. Υλοποίηση του Policy Repository....102
6. Πειραματική Επαλήθευση και Αξιολόγηση της Προτεινόμενης Αρχιτεκτονικής Ασφαλείας....104
6.1. Εισαγωγή....104
6.2. Δοκιμαστικό Περιβάλλον και Σενάρια....104
6.2.1. Δοκιμαστικό Περιβάλλον....105
6.2.2. Σενάρια Μετρήσεων....106
6.2.3. X.509 Certificates....109
6.2.4. Kerberos Tokens....112
6.3. Παράμετροι Αξιολόγησης....114
6.4. Μετρήσεις και Αποτελέσματα....116
6.4.1. Σενάριο SC1 – Εμπιστευτικότητα μηνύματος υπό αυξανόμενο αριθμό πελατών....116
6.4.2. Σενάριο SC2 – Ταυτοποίηση, Ακεραιότητα και Εμπιστευτικότητα Μηνύματος υπό αυξανόμενο μέγεθος μηνυμάτων SOAP σε υψηλό υπολογιστικό φορτίο....120
6.4.3. Σενάριο SC3 – Ταυτοποίηση και Ακεραιότητα Μηνύματος υπό αυξανόμενο μέγεθος μηνυμάτων SOAP σε υψηλό υπολογιστικό φορτίο....124
7. Συμπεράσματα....128
7.1. Σύνοψη....128
7.2. Ανοιχτά Θέματα – Μελλοντικές Κατευθύνσεις....130
8. Βιβλιογραφία....133

Λίστα Σχημάτων

Σχήμα 1: Η ιστορία του EGEE μέχρι το 2004 [Mite04].....19
Σχήμα 2: Τυπική Δομή ενός Computational Grid 20
Σχήμα 3: Αρχιτεκτονική ενός Instrumentation Grid.....22
Σχήμα 4: Ενδεικτική Αρχιτεκτονική Instrumentation Grid με την χρήση Instrument Elements (IEs) [Linz09].....23
Σχήμα 5: Αρχιτεκτονική Web Services [Boot04].....30
Σχήμα 6: Εμπιστευτικότητα (Confidentiality) με την χρήση Κρυπτογραφίας Δημοσίου Κλειδιού.....39
Σχήμα 7: Ταυτοποίηση και Ακεραιότητα Μηνύματος με την χρήση Ψηφιακών Υπογραφών.....39
Σχήμα 8: Δομή Πιστοποιητικού X.509 v3.....43
Σχήμα 9: Η στοίβα πρωτοκόλλων WS-* που συμπεριλαμβάνει το Security [Sing07]52
Σχήμα 10: Γενική Αρχιτεκτονική Διατριβής 67
Σχήμα 11: Προτεινόμενη Αρχιτεκτονική Ασφαλείας.....70
Σχήμα 12: Αλληλουχία Μηνυμάτων (Message Sequence Flow) 73
Σχήμα 13: Διάγραμμα Καταστάσεων του ACM 79
Σχήμα 14: Μεταβίβαση Διαπιστευτηρίων (Delegate Proxy Certificate).....86
Σχήμα 15: Διαστρωμάτωση των πρωτοκόλλων Ασφάλειας της προτεινόμενης Αρχιτεκτονικής 87
Σχήμα 16: Γενικά Στοιχεία ενός Μοντέλου Ασφάλειας Υπολογιστικού Πλέγματος [Fost06b] 90
Σχήμα 17: Παράδειγμα ενός πελάτη με μια Υπηρεσία με την χρήση των handlers της Αρχιτεκτονικής Ασφαλείας 95
Σχήμα 18: Αρχιτεκτονική του ACM.....96
Σχήμα 19: Μέση Ρυθμαπόδοση για αυξανόμενο αριθμό πελατών.....117
Σχήμα 20: Μέσος χρόνος εξυπηρέτησης στον handler μια αίτησης (Total Handler Processing).....118
Σχήμα 21: Μέσος Χρόνος Επεξεργασίας Λειτουργιών Ασφαλείας Αίτησης (Security Request Processing).....118
Σχήμα 22: Μέσες τιμές Χρόνων Επεξεργασίας του ACM Kerberos Token handler 119
Σχήμα 23: Μέσες τιμές Χρόνων Επεξεργασίας του WSS4J X.509 Token handler ..120
Σχήμα 24: Μέση Ρυθμαπόδοση για ACM Kerberos Token Profile και WSS4J X.509 Certificate Token Profile handlers για διαφορετικά μεγέθη του σώματος του SOAP μηνύματος. 121
Σχήμα 25: Effective Throughput για διαφορετικά μεγέθη Σώματος SOAP μηνύματος 122
Σχήμα 26: Υπολογιστικοί Χρόνοι χρησιμοποιώντας Kerberos Security ταυτοποίηση-ακεραιότητα-εμπιστευτικότητα μηνύματος.....123
Σχήμα 27: Υπολογιστικοί Χρόνοι χρησιμοποιώντας τον WSS4J X.509 Token handler για ταυτοποίηση-ακεραιότητα-εμπιστευτικότητα μηνύματος.....124
Σχήμα 28: Μέση Ρυθμαπόδοση για τους WSS4J X.509 Token και ACM Kerberos Token handlers με χρήση ψηφιακών υπογραφών στο μήνυμα (ταυτοποίηση-ακεραιότητα μηνύματος) 125
Σχήμα 29: Effective Throughput για τα δύο handlers με την χρήση ηλεκτρονικών υπογραφών για ταυτοποίηση-ακεραιότητα μηνύματος 126
Σχήμα 30: Χρόνοι Επεξεργασία του ACM Kerberos Token handler με χρήση ηλεκτρονικών υπογραφών (HMAC).....127

Σχήμα 31: Χρόνοι Επεξεργασίας του WSS4J X.509 Certificate handler με την χρήση ηλεκτρονικών υπογραφών (Digital Signature) 127

Λίστα Πινάκων

Πίνακας 1: Λειτουργίες Ασφάλειας Πληροφοριακών Συστημάτων [Mene97].....34
Πίνακας 2: Μοντέλα Οργάνωσης Εμπιστοσύνης μεταξύ Υποδομών Δημοσίου Κλειδιού 44
Πίνακας 3:Security Tokens του πρωτοκόλλου WS-Security (WSS Core).....56
Πίνακας 4: Απλό Παράδειγμα χρήσης του KrbClient API.....101
Πίνακας 5: Ορισμός του RuleTuple σε XML Schema 102
Πίνακας 6: Μεθοδολογία σεναρίων υπό μεταβαλλόμενο υπολογιστικό φορτίο 108
Πίνακας 7: Μέση Ρυθμαπόδοση με χρήση κρυπτογράφησης και ηλεκτρονικών υπογραφών για Kerberos και X.509 Token Profiles.....121

1. Εισαγωγή

1.1. Περιγραφή του Προβλήματος

Στην παρούσα διδακτορική διατριβή προτείνεται μια Αρχιτεκτονική Ασφαλείας που έχει ως σκοπό την βελτίωση της απόδοσης των λειτουργιών ασφαλείας σε κατανεμημένες εφαρμογές Πλέγματος (Grid). Ο κύριος στόχος είναι τα Instrumentation Grids, κατανεμημένα συστήματα μετρήσεων και ελέγχου οργάνων που ακολουθούν την γενική αρχιτεκτονική Συστημάτων Υπολογιστικού Πλέγματος (Computational Grid).

Ειδοποιός διαφορά των Instrumentation Grids με τα Computational Grid είναι συνήθως ο κατά τάξη μεγέθους μεγαλύτερος αριθμός οντοτήτων (instruments) σε σχέση με υπολογιστικούς και αποθηκευτικούς πόρους (computing and storage resources), καθώς και η απαίτηση για αυστηρούς περιορισμούς στους χρόνους απόκρισης που εξαρτώνται από τις προδιαγραφές των συγκεκριμένων εφαρμογών μετρήσεων και ελέγχου. Οι απαιτήσεις ασφαλείας σε ένα Instrumentation Grid μπορεί να είναι ιδιαίτερα σημαντικές λόγω της κρισιμότητας των εφαρμογών παρακολούθησης και ελέγχου κρισίμων υποδομών μέσω δικτύων τύπου Internet (π.χ. απομακρυσμένος μετρήσεις και έλεγχος επιταχυντών, συστημάτων παραγωγής ηλεκτρικής ενέργειας κτλ.).

Στα σύγχρονα Grids η αλληλεπίδραση μεταξύ χρηστών, πόρων και συστημάτων γίνεται μέσω Υπηρεσιών (Services). Η έννοια της Υπηρεσίας είναι βασική και επιτρέπει προτυποποιημένες διαπροσωπίες κατασκευάζοντας ένα ανοιχτό Grid, όπου εύκολα νέοι χρήστες και εφαρμογές μπορούν να προστεθούν.

Η ασφάλεια παίζει κύριο ρόλο σε ένα κατανεμημένο περιβάλλον όπως τα Grids (Computational & Instrumentation) όπου οι διαμεριζόμενοι πόροι (υπολογιστές, αποθηκευτικός χώρος και όργανα) ανήκουν σε διαφορετικούς οργανισμούς μιας Ομοσπονδιακής δομής (Federation). Τα συστήματα ασφαλείας ταυτοποιούν (authenticate) αμοιβαία τους χρήστες και τους κατανεμημένους πόρους (μέσω υπηρεσιών), εξουσιοδοτούν (authorize) τους χρήστες με συγκεκριμένα δικαιώματα στους πόρους και διασφαλίζουν την εμπιστευτικότητα της ανταλλαγής μηνυμάτων.

Τα Computational Grids βασίστηκαν σε κρυπτογραφία Δημοσίου Κλειδιού, με την χρήση ηλεκτρονικών πιστοποιητικών (X.509 Certificates) και πληρεξούσιων πιστοποιητικών (proxy certificates) για την πιστοποίηση των χρηστών και πόρων. Οι αντίστοιχες υλοποιήσεις των αρχιτεκτονικών ασφαλείας επιτρέπουν την ταυτόχρονη ταυτοποίηση σε πολλούς πόρους και υπηρεσίες (single sign-on), την εκπροσώπηση του χρήστη (delegation) με την χρήση προσωρινών πληρεξούσιων πιστοποιητικών (proxy certificates) και την εγκατάσταση ασφαλών κρυπτογραφημένων συνδέσεων.

Σε εφαρμογές Grid ευαίσθητες σε χρόνο απόκρισης, όπως είναι τα Instrumentation Grids, η χρήση κρυπτογραφίας Δημοσίου Κλειδιού μπορεί να αποδειχτεί απαγορευτική. Η κρυπτογραφία Δημοσίου Κλειδιού βασίζεται σε δύσκολα υπολογιστικά προβλήματα (NP-Hard) και παρουσιάζει μεγάλη υπολογιστική πολυπλοκότητα σε σχέση με την Συμμετρική Κρυπτογραφία. Το γεγονός αυτό την καθιστά ακατάλληλη για λειτουργίες ασφαλείας που υπόκεινται σε αυστηρές απαιτήσεις απόδοσης. Στα Instrumentation Grids, ο έλεγχος μια συσκευής ή ενός οργάνου θέτει συγκεκριμένους περιορισμούς στους χρόνους απόκρισης που εξαρτώνται από το όργανο που είναι υπό έλεγχο. Τέτοια όργανα μπορεί να είναι ένα επιταχυντής σωματιδίων, ένα τηλεσκόπιο, μετεωρολογικά όργανα κτλ.

1.2. Προτεινόμενη Λύση

Στην παρούσα διατριβή προτείνεται μια Αρχιτεκτονική Ασφαλείας που καλύπτει όλες τις απαιτήσεις ασφαλείας ενός Instrumentation Grid (συμπεριλαμβανομένης της διαλειτουργικότητας με Computational Grids) λαμβάνοντας υπόψη τις αυξημένες απαιτήσεις απόδοσης (μικροί χρόνοι απόκρισης και αυξημένο φορτίο ανταλλαγής μηνυμάτων μετρήσεων και ελέγχου).

Η αρχιτεκτονική μας ορίζει όλα τα υποσυστήματα (components) ασφαλείας και προδιαγράφει την αλληλεπίδρασή τους με το Instrumentation Grid. Η λύση που προτείνουμε βασίζεται στο πρωτόκολλο Kerberos για την αμοιβαία ταυτοποίηση χρηστών και υπηρεσιών. Ορίζει μια περιοχή ασφαλείας (security domain), όπου όλες οι κρυπτογραφικές λειτουργίες βασίζονται σε κρυπτογραφία Συμμετρικού Κλειδιού. Με τον τρόπο αυτό αποφεύγεται πλήρως η κρυπτογραφία δημοσίου κλειδιού κατά την αλληλεπίδραση ενός χρήστη με ένα όργανο, βελτιώνοντας την απόδοση των λειτουργιών ασφαλείας σε σημαντικό βαθμό.

Η προτεινόμενη αρχιτεκτονική ακολουθεί την προσέγγιση πελάτη/εξυπηρετητή και λειτουργεί σε επίπεδο μηνύματος, μέσω των οποίων επικοινωνούν οι υπηρεσίες ενός Grid. Ορίζει ένα υποσύστημα, τον Access Control Manager (ACM) που επιτρέπει:

- Ταυτοποίηση (Authentication), Εξουσιοδότηση (Authorization) και Ακεραιότητα (Integrity) μηνυμάτων που ανταλλάσσουν τα διαφορετικά συστήματα και χρήστες του Instrumentation Grid.
- Παρέχει έλεγχο πρόσβασης (Authorization) των αιτήσεων των πελατών στις υπηρεσίες ου Instrumentation Grid.

Η αρχιτεκτονική συμπληρώνεται από το υποσύστημα KDC του εξυπηρετητή Kerberos που αποτελεί η Τρίτη Έμπιστη Αρχή (Third Trusted Party), το λογισμικό του πελάτη KrbClient που αναλαμβάνει και υλοποιεί όλες τις αλληλεπιδράσεις του λογισμικού πελάτη με την αρχιτεκτονική και τέλος την υπηρεσία εύρεσης της πολιτικής πρόσβασης σε υπηρεσίες που προστατεύονται από την προτεινόμενη αρχιτεκτονική.

Η απαίτηση διαλειτουργικότητας με τα Computational Grids, τα οποία βασίζονται σε κρυπτογραφία Δημοσίου Κλειδιού και ανήκουν σε διαφορετικές περιοχές ασφαλείας, υλοποιείται με αυτόματο μηχανισμό που ορίζουμε στην αρχιτεκτονική ασφαλείας. Πραγματοποιείται αντιστοιχίζοντας την ταυτότητα του χρήστη/εφαρμογής σε ένα ηλεκτρονικό πιστοποιητικό X.509 με την ταυτότητα του πρωτοκόλλου Kerberos που ακολουθεί η διατριβή.

Η Αρχιτεκτονική Ασφαλείας υλοποιήθηκε σε περιβάλλον Java. Η απόδοσή της επιβεβαιώθηκε πειραματικά με εξομοιώσεις (emulations) και ανάλυση των αποτελεσμάτων ανταλλαγής μηνυμάτων μεταξύ πελατών και εξυπηρετητή σε τοπικό δίκτυο. Η κίνηση μηνυμάτων προέκυψε από σενάρια προσομοίωσης υψηλού και χαμηλού φορτίου ανά πελάτη. Η προτεινόμενη αρχιτεκτονική, συγκρινόμενη με αντίστοιχη παραδοσιακή υλοποίηση ασφαλούς μεταφοράς μηνυμάτων σε Grid, παρουσίασε σημαντική βελτίωση επεκτείνοντας κατά ~50% την μέγιστη ρυθμαπόδοση σε μηνύματα το δευτερόλεπτο.

1.3. Παρουσίαση Περιεχομένων

Η παρούσα διατριβή διαρθρώνεται ως εξής:

Το Κεφάλαιο 2 παρουσιάζει το γενικό περιβάλλον εφαρμογής της αρχιτεκτονικής ασφαλείας που αναπτύξαμε (Computational & Instrumentation Grids, Service Oriented Architecture – SOA και Τεχνολογίες Web Services).

Το Κεφάλαιο 3 περιγράφει τις λειτουργίες ασφαλείας που άπτονται της διατριβής (Κρυπτογραφία Συμμετρικού και Δημοσίου Κλειδιού, Authentication και Authorization σε Grids, το πρωτόκολλο Kerberos, τεχνολογίες ασφαλείας των Web Services).

Το Κεφάλαιο 4 αναλύεται το πρόβλημα με μεγαλύτερη λεπτομέρεια και παρουσιάζει την αρχιτεκτονική ασφαλείας, καθορίζοντας και περιγράφοντας τα υποσυστήματα της και ορίζοντας τους κανόνες και την λειτουργία τους.

Στο Κεφάλαιο 5 παρουσιάζεται η υλοποίηση της αρχιτεκτονικής (βασικοί μηχανισμοί, τεχνολογίες και εργαλεία λογισμικού που χρησιμοποιήσαμε στην υλοποίηση των τμημάτων της αρχιτεκτονικής μας).

Στο Κεφάλαιο 6 παρουσιάζονται τα αποτελέσματα της αρχιτεκτονικής ασφαλείας κατά την λειτουργία εξομοίωσης. Εκτίθενται σενάρια προσομοίωσης κίνησης μηνυμάτων, μετρήσεις των επιδόσεων (ρυθμαπόδοση, καθυστέρηση) και συγκρίσεις με αντίστοιχη παραδοσιακή υλοποίηση ασφαλείας σε Grids.

Τέλος στο Κεφάλαιο 7 παρουσιάζουμε τα συμπεράσματα και τις μελλοντικές επεκτάσεις της αρχιτεκτονικής.

2. Computational και Instrumentation Grids

2.1. Εισαγωγή

Στο παρόν κεφάλαιο θα γίνει η απαραίτητη παρουσίαση του περιβάλλοντος κατανεμημένης Αρχιτεκτονικής Πλέγματος (Grid) που αποτελεί το γενικότερο πλαίσιο της διατριβής. Συγκεκριμένα σε αυτό το κεφάλαιο θα καλύψουμε τα εξής θέματα:

- **Computational Grids**: Στην περίπτωσή μας αποτελεί το περιβάλλον που υποστηρίζει batch λειτουργίες, βοηθητικές ενός real time Instrumentation Grid, π.χ. μαζικής αποθήκευσης παραγόμενων δεδομένων.
- **Instrumentation Grids:** Το κυρίως περιβάλλον που δραστηριοποιείται η αρχιτεκτονική ασφαλείας.
- **SOA και Τεχνολογίες Web Services**: Η επικρατούσα εξέλιξη στις Αρχιτεκτονικές Grid που μεταξύ άλλων παρέχουν το τεχνικό υπόστρωμα της αρχιτεκτονικής ασφαλείας.

Δεν γίνεται παρουσίαση των θεμάτων ασφάλειας σχετικών με την διατριβή. Σε αυτά θα αφιερώσουμε το επόμενο κεφάλαιο.

2.2. Computational Grids

2.2.1. Ορισμός Grid Computing

Ο όρος Υπολογιστικό Πλέγμα (Grid Computing) εφευρέθηκε στο βιβλίο "*The Grid: Blueprint for a New Computing Infrastructure*" [Fost99], όπου χρησιμοποιήθηκε η εξής μεταφορά: η ευκολία πρόσβασης του καθένα στην κατανάλωσης της υπολογιστικής ισχύς όπως συμβαίνει και με την κατανάλωση ηλεκτρικής ενέργειας σε ένα δίκτυο ηλεκτρισμού (Electric Power Grid).

Τα Συστήματα Τεχνολογιών Πλέγματος όπως ορίζει ο Ian Foster στο [Fost02c] είναι ένα σύστημα που τηρεί **τρεις αρχές**:

1. Οι πόροι που συμμετέχουν στο Grid δεν βρίσκονται κάτω από κεντρική διαχείριση
2. Χρησιμοποιούνται ανοιχτά πρότυπα, πρωτόκολλα και διαπροσωπίες
3. Οι συνδυασμένοι πόροι προσφέρουν αυξημένη ποιότητα υπηρεσίας ώστε το άθροισμα των πόρων που συμμετέχουν στην υπηρεσία να είναι μεγαλύτερο από το άθροισμα της αξίας του κάθε πόρου ξεχωριστά.

2.2.2. Εισαγωγή στα Computational Grids

Τα **Συστήματα Τεχνολογιών Πλέγματος** (Computational Grids) είναι συστήματα που ακολουθούν τις αρχές των κατανεμημένων συστημάτων, δημιουργώντας εφαρμογές που επεκτείνονται από τα στενά όρια ενός οργανισμού. Συνδυάζουν υπολογιστικούς πόρους (computing resources) που προέρχονται από διαφορετικούς οργανισμούς, τόσο σε επιχειρησιακό όσο και σε διαχειριστικό επίπεδο. Επιτυγχάνεται με την δικτυακή σύνδεση «χαλαρά» συνδεδεμένων (loosely coupled) υπολογιστών.

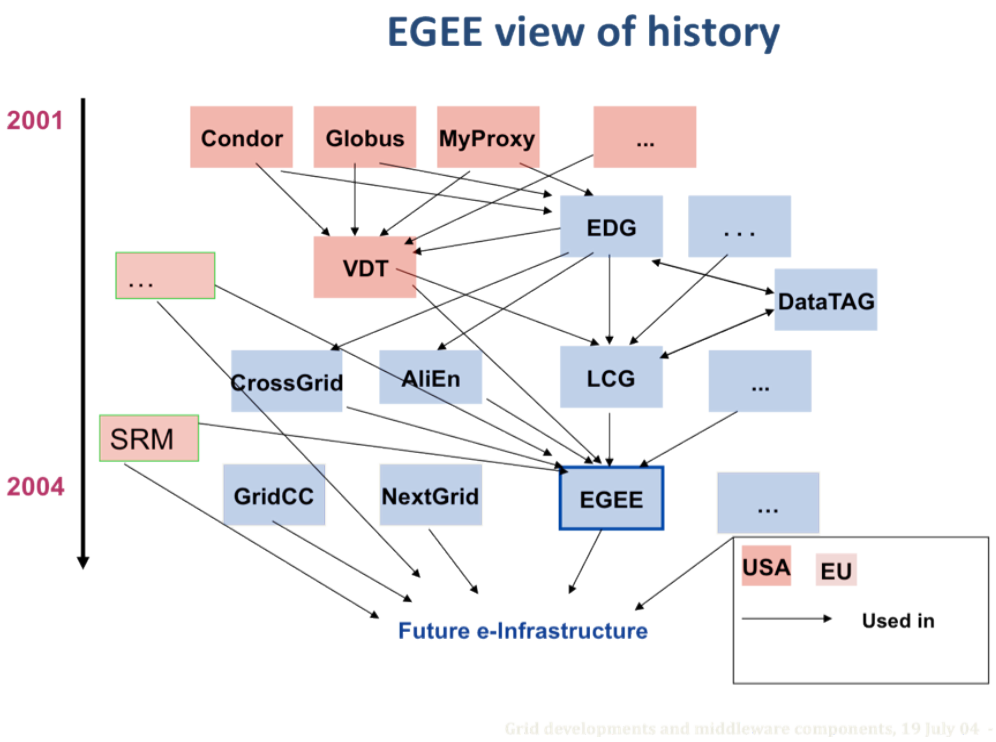
Βασικό συστατικό των Grids είναι το ενδιάμεσο λογισμικό – **middleware** πάνω στο οποίο χτίζεται ένα τέτοιο σύστημα. Το λογισμικό αυτό παρέχει την αναγκαία λειτουργικότητα ώστε να συνεργάζονται τα κατανεμημένα συστήματα σε ένα ομοσπονδιακό περιβάλλον (federated environment). Πρότυπη υλοποίηση middleware είναι το **Globus Toolkit** [GLOBUS] [Fost97], το οποίο βασίζεται σε αυτόνομα συστήματα λογισμικού (components). Κάθε component εκτελεί συγκεκριμένες λειτουργίες, όπως εύρεση υπολογιστικής ισχύος, αποθηκευτικού χώρου, πληροφορίες για το φόρτο του κάθε κόμβου/πόρου κ.α. Το Globus Toolkit προσφέρει όλες τις εφαρμογές, τα εργαλεία και τις υπηρεσίες για την κατασκευή ενός Computational Grid και υποστηρίζεται από την κοινότητα Globus Alliance [GlobusAlliance]. Από έκδοση 4 [Fost05], [Fost06a] όπως και το Globus Crux Toolkit [CruxToolkit], που αποτελεί την επόμενη γενιά, ακολουθούν τις αρχές των Service Oriented Architecture - SOA. Σήμερα βρίσκεται στην έκδοση 5 [GLOBUS].

Βασική επίσης είναι η ιδέα των **Εικονικών Οργανισμών (Virtual Organizations -VOs),** βάση των οποίων δίνονται οι ρόλοι πρόσβασης μέσα στο Grid. Εκτείνονται έξω από τα στενά όρια του οργανισμού και επιτρέπουν την δημιουργία συνόλων (VOs) που περιλαμβάνουν χρήστες και πόρους (υπολογιστές, αποθηκευτικός χώρος, βάσεις δεδομένων κτλ.), τα οποία είναι μέρος των οργανισμών

που συμμετέχουν στο Grid. Επιτρέπουν στους χρήστες τους να μοιράζονται τους πόρους με βάση συγκεκριμένες πολιτικές ανά VO, μέσω των οποίων επιτρέπεται η δημιουργία εικονικών επιστημονικών κοινοτήτων, οι οποίες χρησιμοποιούν υπολογιστικούς πόρους κατανεμημένους σε όλο τον κόσμο και έχουν κοινούς επιστημονικούς σκοπούς. Περισσότερα για τα VOs θα δούμε στην ενότητα 3.4.3.2.

Τα Computational Grids προτάθηκαν κυρίως από την επιστημονική κοινότητα για να δώσουν λύση στο πρόβλημα της εξεύρεσης πόρων υψηλής επεξεργαστικής ισχύος, οι οποίοι είναι απαραίτητοι για την επεξεργασία τεράστιου όγκου δεδομένων που προκύπτουν από επιστημονικά πειράματα. Σε αυτά συγκαταλέγονται πειράματα Φυσικής Υψηλών Ενεργειών όπως αυτά του **Large Hadron Collider - LHC** [LHC] του CERN που αρχικά χρησιμοποιούσε το **European Data Grid** [EDG], το **Grid Physics Network - GriPhyN** [GriPhyN] [Nefe06], το **Network for Earthquake Engineering and Simulation – NEES** [NEES]. Το γενικευμένο ευρείας κλίμακας Ευρωπαϊκό Grid υλοποιήθηκε μέσω του έργου **EGEE – Enabling Grids for E-sciencE** [EGEE10], [EGEE]. Για να έχουμε μια αίσθηση των απαιτήσεων, αναφέρουμε πως ο επιταχυντής **Large Hadron Collider** (LHC), ο οποίος έχει κατασκευαστεί στο CERN, υπολογίζεται ότι κατά την λειτουργία του θα παράγει περίπου 15 Petabytes (15 εκατομμύρια Gigabytes) δεδομένων ετησίως [LHCD]. Σε αυτά τα δεδομένα έχουν πρόσβαση χιλιάδες επιστήμονες από όλο τον κόσμο, οι οποίοι τα αναλύουν χρησιμοποιώντας το Computational Grid που έχει χτιστεί για αυτό το σκοπό.

Η εξέλιξη των Grids βασίζεται στην εξέλιξη του middleware. Στο Σχήμα 1 βλέπουμε την εξέλιξη του EGEE. Αποτέλεσμά του, όσον αφορά το λογισμικό, είναι το **gLite** [gLite]. Η προσπάθεια του EGEE συνεχίζει ως έργο EGI [EGI] το οποίο ως φιλοσοφία ακολουθεί το αυτόνομο μοντέλο συντονισμού εθνικών προσπαθειών. Σημαντικό χαρακτηριστικό των Grids, είναι η δημιουργία λογισμικού και συστημάτων μέσα από την διαδικασία του **ελεύθερου λογισμικού – λογισμικού ανοιχτού κώδικα (free and open-source software)** και των **ανοιχτών προτύπων**, επιτρέποντας στα νεότερα έργα να εξελιχθούν χρησιμοποιώντας τα αποτελέσματα των προηγούμενων.



Σχήμα 1: Η ιστορία του EGEE μέχρι το 2004 [Mite04]

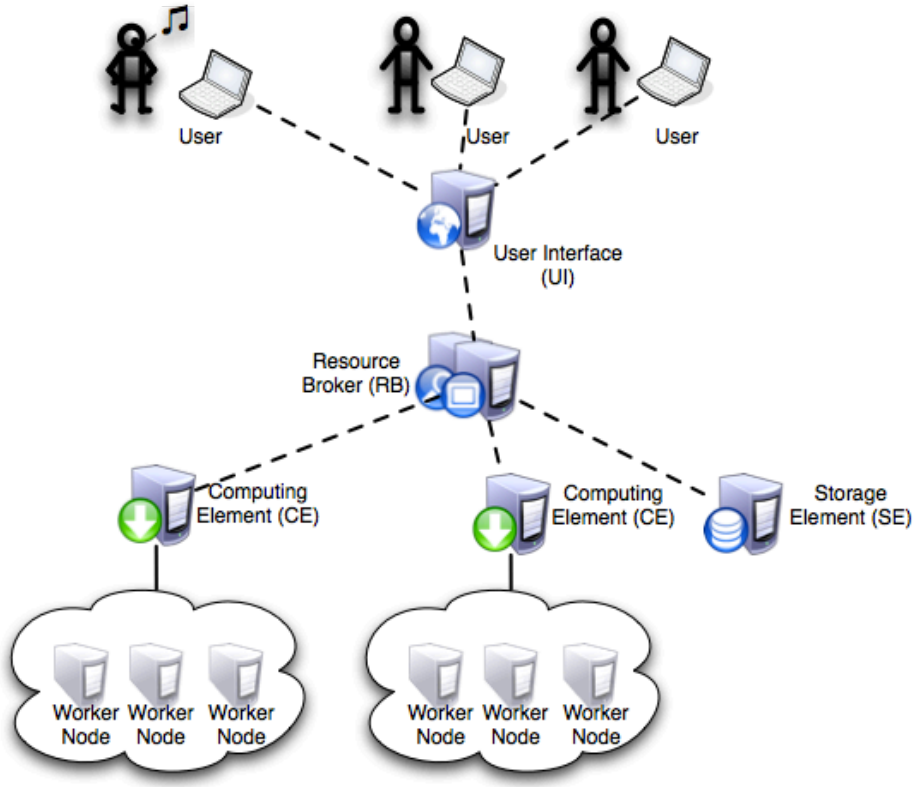
Στη συνέχεια του ενότητας θα δούμε τα δομικά συστατικά ενός Grid, θα αναφερθούμε στα Instrumentation Grids που αποτελούν το κυρίως περιβάλλον της διατριβής και θα παρουσιάσουμε εν συντομία τις SOA και τις τεχνολογίες Web Services.

2.2.3. Δομή Computational Grid

Ένα ενδεικτικό Computational Grid (βλέπε και Σχήμα 2) όπως το gLite [gLite] αποτελείται από:

- **User Interface (UI)**: Προσφέρει στον τελικό χρήστη την πρόσβαση στο Grid. Συνήθως είναι ένας υπολογιστής στον οποίο έχουν εγκατασταθεί τα εργαλεία πρόσβασης ως *Command Line Interface – CLI*. Εναλλακτικά το UI μπορεί να είναι μια web εφαρμογή, όπως το Ganga [GANG] και το GridSphere [GSPH]. Επιτρέπει την ταυτοποίηση του χρήστη στο Grid, την προβολή των πόρων που είναι κατάλληλοι για την εκτέλεση μιας συγκεκριμένης εργασίας (job), την υποβολή εργασιών προς εκτέλεση, την ακύρωση εργασιών, την ανάκτηση του αποτελέσματος ολοκληρωμένων εργασιών κτλ.

- **Computing Element (CE)**: Αντιπροσωπεύει ένα υπολογιστικό πόρο (resource) στο Grid, ο οποίος μπορεί να είναι ένα cluster ή ένα σύνολο από τοπικούς υπολογιστικούς κόμβους που ονομάζονται Worker Nodes (WNs). Το CE προσφέρει τις υπηρεσίες (services)**: Grid Gate (GG)** που είναι το γενικό interface πρόσβασης στους WNs ή στο cluster, Information Service που ενημερώνει το Grid για την κατάσταση του CE και το **Local Resource Management System (LRMS)** το οποίο είναι το batch system που φροντίζει την εκτέλεση των εργασιών που υποβάλλονται στο CE. Παραδείγματα του LRMS είναι το Condor [Litz88] [Condor], το OpenPBS [PBS] και το Torque [Stap06].



Σχήμα 2: Τυπική Δομή ενός Computational Grid

- **Worker Node (WN)**: Οι Worker Nodes είναι οι κόμβοι του Grid όπου εκτελούνται οι εργασίες των χρηστών. Τρέχουν ειδικό λογισμικό που τους επιτρέπει να συνεργάζονται με το CE που ανήκουν.
- **Storage Element (SE)**: Προσφέρει μια ομογενοποιημένη διαπροσωπία προς αποθηκευτικούς πόρους. Οι αποθηκευτικοί πόροι σε ένα Grid είναι συνήθως σειρές από δίσκους (disk arrays) ή ένα Mass Storage Systems

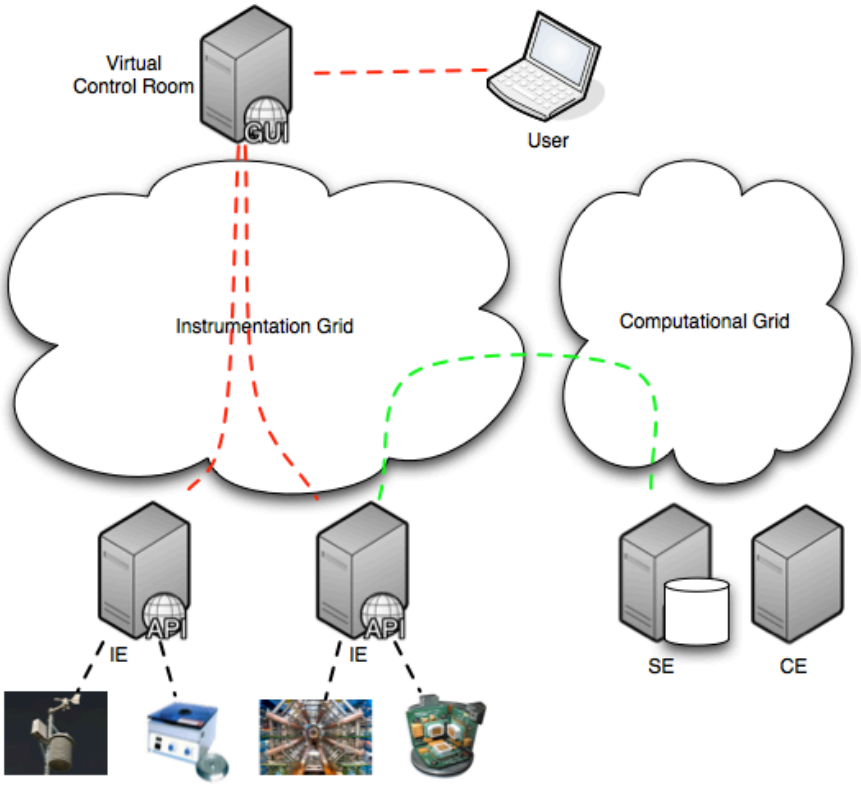
(MSS) που συνδυάζει δίσκους και αποθηκευτικές ταινίες. Στο SE προσφέρεται η υπηρεσία (service) **Storage Resource Manager (SRM)**, η οποία διαχειρίζεται τους αποθηκευτικούς πόρους προσφέροντας δυνατότητες όπως κράτηση (reservation) αποθηκευτικού χώρου, μεταφορά από δίσκο σε ταινία, μεταφορά από SE σε SE κτλ. Υλοποιήσεις του SRM είναι το gLite Disk Pool Manger [Stew07] και το CERN Advanced STORage manager (CASTOR) [Baud03]. Τα πρωτόκολλα μεταφοράς που χρησιμοποιεί το SRM είναι το GridFTP [Allc03] και το Remote File I/O (RFIO) του CASTOR.

- **Resource Broker (RB)**: Είναι ο κόμβος ο οποίος φιλοξενεί την υπηρεσία **Workload Management System (WMS)** [Avel04]. Το WMS δέχεται τις εργασίες των χρηστών (jobs), αναλαμβάνει να τις στέλνει σε κατάλληλα CEs και να ανακτά το αποτέλεσμα εκτέλεσής τους. Οι εργασίες περιγράφονται με την γλώσσα Job Description Language – JDL και υποβάλλονται μέσω του UI στο WMS. Το WMS χρησιμοποιεί την υπηρεσία MatchMaker για ανακαλύψει που σε ποιο CE και WNs θα στείλει την κάθε εργασία ώστε να τηρηθούν οι προδιαγραφές όπως ορίζονται στην περιγραφή της εργασίας (στο JDL).
- **Information Service**: Είναι μια υπηρεσία που υπάρχει σε όλους τους κόμβους του Grid και επιτρέπει την ανακάλυψη των πόρων και την ανάκτηση πληροφοριών σχετικές με την κατάστασή τους. Η δομή των δεδομένων για την παρακολούθηση και ανακάλυψη των πόρων του Grid ακολουθεί το GLUE Schema [Andr09]. Συγκεκριμένη υλοποίησή του είναι το Monitoring and Discovery Service (MDS) [MDS] του Globus Toolkit.

2.3. Instrumentation Grid

Ένα Instrumentation Grid αφορά στον έλεγχο οργάνων μετρήσεων και ελέγχου. Στο [Wang04c], στο **GridCC** [GRIDCC], στο **DORII** [DORI] και στο [Iosu04] προτείνεται η χρήση τεχνολογιών Grid για το έλεγχο κατανεμημένων συσκευών μετρήσεων και ελέγχου (***Instrumentation Grid*** ή ***Sensor Grid***). Ενδεικτικές εφαρμογές που μελετήθηκαν στο GridCC, όπου ο έλεγχος συσκευών και οργάνων αποτελεί βασική λειτουργία είναι: (i) το **Synchrotron Radiation Sotorage Ring** στην Elletra [ELLETRA], (ii) το Run Control System of Data Acquisition System που

ανήκει ***στο Compact Muon Solenoid (CMS)*** detector του Large Hadron Collider (LHC) [CMS08] του CERN (iii)_ έλεγχος ενός Δικτύου Κατανομής Ηλεκτρικής Ενέργειας (***Power Grid***) [Irvi04].

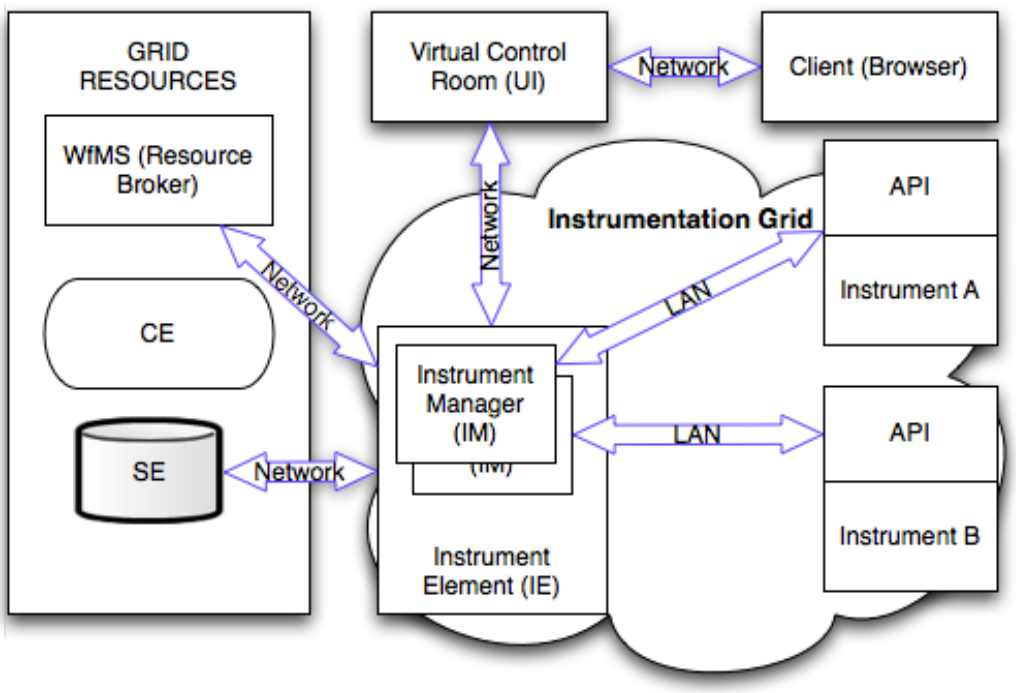


Σχήμα 3: Αρχιτεκτονική ενός Instrumentation Grid

Τέτοια συστήματα Instrumentation Grids είναι πολύπλοκα και ανομοιογενή. Περιλαμβάνουν ένα τεράστιο αριθμό από όργανα, αισθητήρες (sensors) ($O(10^4)$) και επιστημονικό εξοπλισμό που συμμετέχουν σε ένα πείραμα. Ο εξοπλισμός αυτός αποτελείται από διαφορετικά υποσυστήματα και λογισμικά που πιθανόν ανήκουν σε διαφορετικούς προμηθευτές, ακολουθούν διαφορετικές τεχνολογίες. Ο κεντρικός έλεγχος τους, γίνεται από εφαρμογές που χρησιμοποιούν τα εγγενή πρωτόκολλα των συστημάτων (π.χ. συστήματα SCADA [SCAD]) και γίνεται σε ένα κλειστό ελεγχόμενο περιβάλλον. Ωστόσο, παρουσιάζει ενδιαφέρον η δυνατότητα απομακρυσμένου ελέγχου με ασφάλεια από **Virtual Control Rooms** μέσω κατανεμημένων Instrumentation Grids, όπως φαίνεται στο Σχήμα 3.

Στο GridCC που είναι μια υλοποίηση ενός Instrumentation Grid προτάθηκε το **Instrument Element (IE)** [Friz06] [Lell07], το οποίο (κατ αναλογία του CE στο Computational Grid) είναι ένα σύνολο υπηρεσιών που προσφέρουν την απαιτούμενη

διαπροσωπία ώστε να ενεργοποιήσουν τον απομακρυσμένο έλεγχο και επίβλεψη φυσικών οργάνων και συσκευών. Μαζί με πόρους (resources) και υπηρεσίες (services) που υπάρχουν σε συνεργαζόμενα Computational Grids, (**Computing Element (CE),** το **Storage Element (SE)**, ο **Resource Broker (RB)** κτλ.) προσφέρουν ολοκληρωμένες υπηρεσίες στην διεξαγωγή ενός πειράματος, χρησιμοποιώντας το Grid. Στο Σχήμα 3 απεικονίζεται η σχέση ενός Instrumentation Grid με ένα Computational Grid.



Σχήμα 4: Ενδεικτική Αρχιτεκτονική Instrumentation Grid με την χρήση Instrument Elements (IEs) [Linz09]

Στο Σχήμα 4 παρουσιάζεται με μεγαλύτερη λεπτομέρεια η δομή ενός Instrumentation Grid με χρήση IEs. Κεντρικό component αποτελεί το IE. Αυτό συμπεριλαμβάνει ένα σύνολο από Instrument Managers (IMs). Κάθε IM μπορεί μέσω τοπικού δικτύου να έχει πρόσβαση σε ένα όργανο μετρήσεων και ελέγχου, μέσω του API που το όργανο προσφέρει. Το IM είναι ο οδηγός για το είδος του οργάνου που ελέγχεται και είναι συγκεκριμένος για κάθε όργανο, αφού σχετίζεται με το API του. Το IE ορίζει ένα κοινό interface για όλα τα IMs, ομογενοποιώντας την πρόσβαση στα όργανα και προσφέροντας επιπλέον υπηρεσίες συνεργαζομένων Computational Grids. Τέτοιες υπηρεσίες είναι: Η ασφαλής πρόσβαση (Ταυτοποίηση - Authentication

& Εξουσιοδότηση - Authorization) μέσω γενικών υποδομών ασφαλείας που λειτουργούν στα πλαίσια των Computational Grids και η χρήση της τεράστιας αποθηκευτικής ικανότητα των SEs για την αποθήκευση δεδομένων που παράγει το όργανο.

Τα Instrumentation Grids αποτελεί το περιβάλλον που δραστηριοποιείται η αρχιτεκτονική ασφαλείας που προτείνουμε. Έρχεται να συνεισφέρει στην απαιτούμενη αύξηση της απόδοσης μεταφρασμένη σε μικρότερους χρόνους καθυστέρησης (delay times) για τον έλεγχο του οργάνου και σε αυξημένη ικανότητα επεξεργασίας ασφαλών μηνυμάτων ελέγχου στο IE. Λεπτομερής παρουσίαση της αρχιτεκτονικής ασφαλείας θα παρουσιαστεί στο 4^{ο} Κεφάλαιο.

Το κατανεμημένο περιβάλλον Grid (Computational & Instrumentation) εντάσσονται στο γενικότερο πλαίσιο της αρχιτεκτονικής SOA (Service Oriented Architecture), που προτείνει την χρήση της υπηρεσίας (service) ως δομικό συστατικό ενός κατανεμημένου συστημάτων, συνήθως υλοποιημένο με βάση τις τεχνολογίες Web Services (WS).

2.4. Αρχιτεκτονικές SOA και Web Services

Οι **Υπηρεσιοστρεφείς Αρχιτεκτονικές (Service Oriented Architecture - SOA)** είναι ένα σύνολο από ευέλικτες σχεδιαστικές αρχές που χρησιμοποιούνται κατά τον σχεδιασμό κατανεμημένων αρχιτεκτονικών. Δομικό συστατικό των SOA είναι η έννοια της **Υπηρεσίας (Service)**.

Οι προδιαγραφές και τα πρότυπα στις Αρχιτεκτονικές SOA εκπορεύονται από Οργανισμούς Ανοιχτών Προτύπων, όπως το **OASIS** [OASIS], **το Open Group** [OPENGRO] και το **OMG** [OMG], όπου συμμετέχουν εταιρίες και φορείς που δραστηριοποιούνται στο χώρο. Αποτέλεσμα της δουλειάς τους είναι πρότυπα, ορολογίες, γλώσσες περιγραφής SOA και πρότυπες αρχιτεκτονικές, που συχνά αλληλεπικαλύπτονται.

2.4.1. Ορισμός των SOA

Στο [Kreg09] παρουσιάζεται μια μέθοδος που θα διευκολύνει τους εμπλεκόμενους (system architects, developers, organizations) στην κατανόηση των προτύπων των SOA. Εκεί προτείνονται οι βασικές αρχές (SOA Principles): Service, Real World Effect, Visibility, Service Description, Policies and Contracts, Execution Context και Interaction. Εκτός από το πρώτο (service), οι υπόλοιπες αρχές είναι στην

ουσία χαρακτηριστικά της ίδιας της υπηρεσίας και θα περιγραφούν στην ενότητα 2.4.2.

Ο ορισμός για τα SOA του **OASIS** [OASIS] που δίνεται στο **Reference Model for Service Oriented Architecture 1.0, SOA-RM** [MacK06] είναι:

> *Τα SOA είναι ένα υπόδειγμα (paradigm) για την οργάνωση και χρησιμοποίηση κατανεμημένων δυνατοτήτων (capabilities) που μπορεί να είναι υπό τον έλεγχο διαφορετικών οργανισμών (ownership domains). Προσφέρει ένα ομογενοποιημένο τρόπο για την προσφορά, ανακάλυψη και αλληλεπίδραση με τις προσφερόμενες από τους οργανισμούς δυνατότητες, στοχεύοντας στην παραγωγή συνεπών αποτελεσμάτων με μετρήσιμες προϋποθέσεις και προσδοκίες.*

Η έννοια της Υπηρεσίας (Service) αποτελεί κεντρική έννοια στα SOA γιατί αποκαλύπτει μια κατανεμημένη δυνατότητα, όπως θα περιγράψουμε στην επόμενη ενότητα.

2.4.2. Ορισμός της Υπηρεσίας σύμφωνα με τα SOA

Σύμφωνα με το **OpenGroup** [OGSOA] μια ***Υπηρεσία***:

- Είναι μια λογική ***αναπαράσταση*** μιας επαναλαμβανόμενης ***Επιχειρησιακής Δραστηριότητας*** που έχει συγκεκριμένο αποτέλεσμα.
- Είναι ***αυτόνομη*** ώστε να υπάρχει δυνατότητα ευέλικτης σύνθεσης άλλων υπηρεσιών και να μην επηρεάζεται από δυσλειτουργίες άλλων υπηρεσιών.
- Μπορεί να είναι ***σύνθετη*** και να αποτελείται από άλλες υπηρεσίες.
- Ακολουθεί την ***λογική του Μαύρου Κουτιού (Black Box)*** για τους καταναλωτές της Υπηρεσίας.

Το Reference Model for SOA [MacK06] του OASIS δίνει έμφαση στο πως πρέπει να είναι οι υπηρεσίες SOA για να καταναλωθούν. Συγκεκριμένα γίνεται αντιστοίχιση των αναγκών (**Needs**) ενός Καταναλωτή (**Consumer**) με τις δυνατότητες (**Capabilities**) πόρων (resources) μέσω ενός Παρόχου Υπηρεσίας (**Service Provider - SP)**. Στα πλαίσια αυτά, μια Υπηρεσία SOA έχει τα εξής χαρακτηριστικά:

- ***Περιγραφή Υπηρεσίας (Service Description)***: Είναι η απαραίτητη πληροφορία για να μπορέσει κάποιος consumer να ανακαλύψει την υπηρεσία, να κατανοήσει τι αυτή προσφέρει και τέλος να την χρησιμοποιήσει. Είναι αναγκαία απαίτηση, ειδικά όταν οι consumers της υπηρεσίας προέρχονται από διαφορετικές διαχειριστικές περιοχές.
- ***Ορατότητα (Visibility)***: Η σχέση των SPs και των consumers σε μια υπηρεσία SOA πρέπει να είναι διαφανής με την παροχή περιγραφών των λειτουργιών, των τεχνικών απαιτήσεων, των περιορισμών, της πολιτικής και των μηχανισμών πρόσβασης στην υπηρεσία.
- ***Αλληλεπίδραση (Interaction)***: Διεξάγεται με την ανταλλαγή μηνυμάτων μεταξύ service provider και consumers, μέσω των οποίων ανταλλάσσεται πληροφορία.
- ***Αποτέλεσμα Αντιληπτό στον Πραγματικό Κόσμο (Real World Effect)***: Είναι το φυσικό αποτέλεσμα χρήσης μια υπηρεσίας. Μπορεί να είναι η επιστροφή πληροφορίας ή η αλλαγή κατάστασης της οντότητας που προσφέρει την δυνατότητα.
- ***Περιβάλλον Εκτέλεσης (Execution Context)***: Είναι το σύνολο των τεχνικών και επιχειρησιακών στοιχείων που δημιουργούν το μονοπάτι μεταξύ αυτών που διαθέτουν τους πόρους (resources) ή δυνατότητες (capabilities) και των SPs επιτρέποντας έτσι την αλληλεπίδραση των υπηρεσιών με τους καταναλωτές τους. Προσφέρει ένα Σημείο Απόφασης (Decision Point) για τις εφαρμοζόμενες στην υπηρεσία πολιτικές και τα αντίστοιχα συμβόλαια.
- ***Συμβόλαιο και Πολιτική (Contract & Policy)***: Μια πολιτική αντιπροσωπεύει ένα είδος περιορισμού και προϋποθέσεων στην χρήση, εγκατάσταση ή περιγραφή μιας υπηρεσίας μεταξύ των συμμετεχόντων (Consumers και SPs). Το συμβόλαιο αντιπροσωπεύει την συμφωνία μεταξύ δύο ή περισσοτέρων συμβαλλόντων, όπως Consumers, SPs και κατόχων των δυνατοτήτων (capabilities) ή πόρων (resources) μιας υπηρεσίας.

Να τονίσουμε πως τα SOA δεν είναι μια συγκεκριμένη αρχιτεκτονική αλλά αρχιτεκτονικές-σχεδιαστικές αρχές βασισμένες στην έννοια της υπηρεσίας. Για αυτό κάθε λύση που σχεδιάζεται ακολουθώντας την προσέγγιση SOA πρέπει να δίνει λύση

στα ιδιαίτερα προβλήματα της περιοχής ενδιαφέροντος που εφαρμόζεται σεβόμενη παράλληλα τις βασικές αρχές της.

2.4.3. Πρότυπες Αρχιτεκτονικές SOA

Η προτυποποίηση στα SOA δεν περιορίζεται μόνο στις γενικές αρχές της υπηρεσίας όπως τις παρουσιάσαμε στο 2.4.2, αλλά προτείνονται **Πρότυπες Αρχιτεκτονικές (Reference Architectures)** όπως του **OASIS Reference Architecture for SOA** [Este09]. Οι αρχιτεκτονικές αυτές προϋποθέτουν μια κοινή γλώσσα για την κατανόηση σημαντικών χαρακτηριστικών της λύσης SOA, η οποία εξειδικεύει τους όρους και συνθήκες της κάθε περιοχής ενδιαφέροντος. Ωστόσο δεν επεκτείνεται σε θέματα που άπτονται στην δημιουργία, χρήση και κατοχή ενός συστήματος SOA. Αυτά εξειδικεύονται σε συγκεκριμένες υλοποιήσεις.

Μια πρότυπη αρχιτεκτονική SOA για Grids είναι το **OGSA** [Fost06b]. Η ιδέα αυτή αρχικά περιγράφηκε στο [Fost02a]. Η ανάπτυξη της αρχιτεκτονικής έχει προέλθει από τον οργανισμό **Global Grid Forum (GGF),** ο οποίος ενώθηκε το 2006 με τον οργανισμό Enterprise Grid Alliance και δημιουργήθηκε το **Open Grid Forum (OGF)** [OGF]. Υλοποιεί τις Υπηρεσίες ως **Web Services** (συντακτικό XML και πρωτόκολλο μεταφοράς μηνυμάτων SOAP πάνω από πρωτόκολλο HTTP, βλέπε την επόμενη ενότητα) και παρέχει, εν συντομία, κατανεμημένη αλληλεπίδραση και υπολογιστική ισχύ βασισμένη σε υπηρεσίες. Καλύπτει την διαλειτουργικότητα σε ετερογενή συστήματα, έτσι ώστε να μπορούν διαφορετικοί τύποι πόρων (resources) να επικοινωνούν και να μοιράζονται πληροφορίες.

Η αρχιτεκτονική **OGSA** ορίζει εφτά ανεξάρτητες υπηρεσίες: **Infrastructure services**, **Execution Management services**, **Data services**, **Resource Management services**, **Security services**, **Self-Management services** και **Information services**.

2.5. Web Services

Στο Web Services Glossary [Hass04] του οργανισμού W3C [W3C] δίνεται ο εξής ορισμός:

> "***Ένα Web Service είναι ένα σύστημα λογισμικού σχεδιασμένο να υποστηρίζει την διάδραση μηχανής-προς-μηχανή μέσω δικτύου. Διαθέτει μία διαπροσωπία (interface) που περιγράφεται σε μορφή ικανή προς επεξεργασία από μηχανή (συγκεκριμένα με τη Web Service Definition Language - WSDL). Τα άλλα συστήματα αλληλεπιδρούν με***

το Web Service με τρόπο που προδιαγράφεται στην περιγραφή WSDL χρησιμοποιώντας μηνύματα SOAP, που συνήθως μεταφέρονται με HTTP, με την χρήση της XML για την μετατροπή των δεδομένων (XML Serialization) και σε συνεργασία με άλλα πρότυπα σχετικά με το Web."

Στην συνέχεια της ενότητας θα παρουσιάσουμε την στοίβα των πρωτοκόλλων Web Services και τα περιφερειακά πρωτόκολλα που προσφέρουν αυξημένες δυνατότητες στα Web Services.

2.5.1. Τα Είδη και η βασική Στοίβα Πρωτοκόλλων Web Services

Το βασικό χαρακτηριστικό των WSs είναι η απλότητα στην δημιουργία τους. Ακολουθούν μια πολυεπίπεδη διαστρωμάτωση πρωτοκόλλων, ορίζοντας μια **Αρχιτεκτονική Web Services** και παρέχουν την βασική λειτουργικότητα ανά επίπεδο και ανεξάρτητα του συστήματος που προσφέρεται η υπηρεσία ή της γλώσσας κατασκευής της. Η ανεξαρτησία αυτή αποτελεί το βασικό προτέρημα σε σχέση με άλλα πρωτόκολλα επικοινωνίας μεταξύ μηχανών όπως η ***Common Object Request Broker Architecture – CORBA*** [CORBA] του οργανισμού Object Management Group [OMG], το πρωτόκολλο Java ***Remote Method Invocation - RMI*** [Woll96] (Java specific), ή του ***Distributed Component Object Model (DCOM)*** [DCOM] της Microsoft (.NET specific).

Τα Web Services (WSs) διαχωρίζονται σε τρία βασικά είδη:

- **Remote Procedure Calls – RPC**: Το Web Service αντιστοιχίζεται σε μια λογική ή φυσική λειτουργία του κατανεμημένου συστήματος. Ο τρόπος αυτός είναι ο πιο φυσικός όταν ήδη υπάρχει το σύστημα με καθορισμένες τις λειτουργίες του, επειδή ουσιαστικά γίνεται έκθεση των λειτουργιών αυτών ως Web Service. Ωστόσο, επειδή φέρει πολλές λεπτομέρειες του API του αντίστοιχου κατανεμημένου συστήματος, το Web Service δεν υποστηρίζει την χαλαρή διασύνδεση (loosely coupling) consumer (client) και service (server) και την αυτονομία των υπηρεσιών που απαιτείται σε μια αρχιτεκτονική SOA.
- **RESTful** (**Representational State Transfer – REST**) [Boot04]: βασίζονται σε μεταφορά κατάστασης με χρήση XML, URI και HTTP. Το REST [Fiel00] όπως όλες οι αρχιτεκτονικές τεχνοτροπίες (architectural

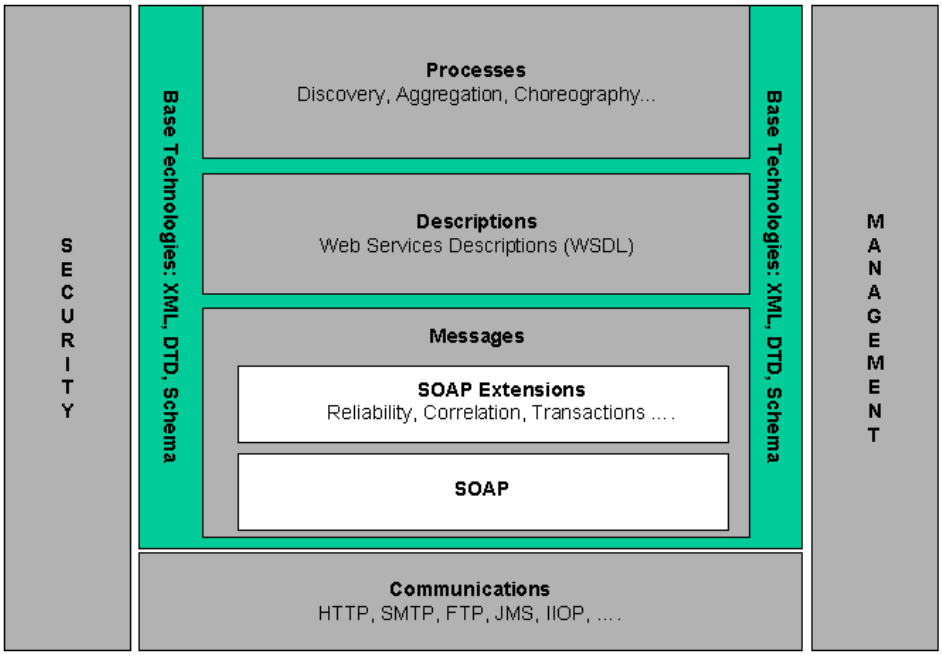
patterns), ορίζει ένα σύνολο κανόνων που αφορούν υποσυστήματα και συνδέσεις μεταξύ τους. Το REST ως βασική σχεδιαστική αρχή θεωρεί μια αρχιτεκτονική πελάτη-εξυπηρετητή χωρίς μνήμη, όπου τα Web Services αντιμετωπίζονται ως πόροι που αναγνωρίζονται από τα URL τους και είναι διαχειρίσιμοι μέσω αυτών. Δεν ορίζουν συγκεκριμένο συμβόλαιο, το οποίο στα SOAP Web Services εκφράζεται μέσω του WSDL. Επίσης χρησιμοποιούν ως operations τις μεθόδους του πρωτοκόλλου HTTP. Το γεγονός έλλειψης συμβολαίου και της χρήσης των περιορισμένων μεθόδων HTTP (βασικές εντολές GET, POST, PUT, DELETE [Field99]) αποτρέπουν την ευρεία χρήση τους σε ένα περιβάλλον SOA. Τέλος η μη διατήρηση κατάστασης (state) από την ίδια την υπηρεσία ανά πελάτη καθιστά τα RESTful Web Services ακατάλληλα για περιβάλλοντα Computational Grid. Για τον λόγο αυτό δεν θα επεκταθούμε παραπάνω σε αυτές.

- **Message Oriented**: Το Web Service σε αυτήν την περίπτωση βασίζεται στην μεταγωγή μηνυμάτων (Messaging System). Θεωρούνται τα καταλληλότερα για χρήση σε περιβάλλον SOA.

Από τα παραπάνω είδη WSs, που τεχνικά διαχωρίζονται στο πως δομούν την πληροφορία και σχεδιαστικά, πως αντιμετωπίζουν τις κατανεμημένες δυνατότητες που παρέχονται από τους διαφόρους παρόχους. Τα RPC και RESTful δεν περιλαμβάνονται στο πλαίσιο διαλειτουργικότητας [Ball04] των WSs, όπως το ορίζει ο οργανισμός **Web Services Interoperability Organization** [WS-I].

Αντίθετα, τα WSs που είναι message oriented μπορούν να προσφέρουν την λειτουργία της χαλαρής διασύνδεσης (loose coupling), απαραίτητη για συστήματα SOA και περιγράφονται στο πλαίσιο διαλειτουργικότητας του WS-I. Στην διατριβή όταν αναφερόμαστε σε WSs αναφερόμαστε στα message oriented.

Η κύρια στοίβα των βασικών πρωτοκόλλων από το επίπεδο μεταφοράς προς το επίπεδο της εφαρμογής, συμπεριλαμβανομένου και των βασικών πρωτοκόλλων ορισμού και αναζήτησης υπηρεσιών, απεικονίζονται στο Σχήμα 5 είναι:



Σχήμα 5: Αρχιτεκτονική Web Services [Boot04]

- **Επίπεδο Μεταφοράς (Υπηρεσίας) (Communications)**: Είναι υπεύθυνο για την μεταφορά μηνυμάτων μεταξύ του πελάτη και του παρόχου της υπηρεσίας. Τα πρωτόκολλα που δύναται να χρησιμοποιηθούν είναι τα **HTTP** [Fiel99], **SMTP** [Post82] [Klen08], **FTP** [Post85], **BEEP** [Rose01] κ.α. το ευρέως διαδεδομένο είναι το **HTTP** με τη χρήση των διευθύνσεων **URL** (Uniform Resource Location). Δεν αποκλείονται όμως και άλλα πρωτόκολλα που μπορούν να δράσουν ως μέσο μεταφοράς των μηνυμάτων.
- **Επίπεδο Μηνυμάτων (Messages)**: Είναι υπεύθυνο για την κωδικοποίηση των μηνυμάτων σε μορφή **XML** [XML], ώστε να είναι δυνατή η κωδικοποίηση/αποκωδικοποίηση της πληροφορίας στα δύο άκρα. Σε αυτό το επίπεδο ανήκουν πρωτόκολλα όπως το **SOAP 1.2** [Mitr07] [Gudg07], που περιγράφουν το κυρίως μήνυμα, το πρότυπο **WS-Addressing** [Gudg06] που προδιαγράφει ονοματολογία υπηρεσίας ανεξάρτητη του επιπέδου μεταφοράς. Επίσης συμπεριλαμβάνονται τα πρωτόκολλα που υποστηρίζουν λειτουργίες ασύγχρονων ειδοποιήσεων όπως τα **WS-BrokeredNotification** [Chap06], **WS-BaseNotification** [Grah06a] και **WS-Topics** [Vamb06]. Το πρότυπο **Attachments Profile** [Ferr04]

προδιαγράφει πως ενσωματώνονται συνημμένα σε ένα μήνυμα Web Services.

- **Επίπεδο Περιγραφής Υπηρεσίας (WSDL Descriptions)**: Η περιγραφή μιας υπηρεσίας ορίζεται με την χρήση XML Schema [Fall04]. Αποκαλείται ***WSDL Document της υπηρεσίας*** [Chris01] και χρησιμοποιείται για να περιγράψει το public interface (διαπροσωπία) ενός Web Service και επομένως να διευκολύνει την τεχνική «κατανάλωσή» του από ένα ενδιαφερόμενο. Η ίδια η διαπροσωπία μαζί με το URL της υπηρεσίας, παρέχονται από τεχνικής άποψης από το πρωτόκολλο **WSDL** (Web Service Definition Language) στις εκδόσεις **1.1** [Chris01], **1.2** [Chin03] και **2.0** [WSDL2.0]. Μέσω WSDL μπορεί να πραγματοποιηθεί η αυτόματη κατανάλωση μιας υπηρεσίας και αποτελεί βασικό πρωτόκολλο στα SOAP Web Services. Επιπλέον στο επίπεδο της περιγραφής ορίζονται και άλλα πρωτόκολλα που διευκολύνουν την χρήση της Υπηρεσίας καλύπτοντας την σημασιολογική ανάγκη αλλά την περιγραφή των διαφόρων πολιτικών που ακολουθεί η ίδια η υπηρεσία. Η σημασιολογική περιγραφή δίνεται με πρωτόκολλα όπως το **Web Services Semantics (WSDL-S)** [Akki05].
- **Επίπεδο Ανακάλυψης (Υπηρεσίας)**: Συγκεντρώνει σε ένα κοινό αποθετήριο της πληροφορίας για την τοποθεσίας και την περιγραφή της Υπηρεσίες Ιστού. Αυτές τις πληροφορίες τις δημοσιεύουν οι ίδιες οι υπηρεσίες. Το πρωτόκολλο **Universal Description Discovery and Integration UDDI** [Clem04] έχει προταθεί από το OASIS για αυτόν το σκοπό.

2.5.2. Web Services και Grids

Τα πρότυπα και πρωτόκολλα των Web Services που αναφέραμε παραπάνω, προσδίδουν την απαραίτητη λειτουργικότητα στα Web Services ώστε να μπορούν να χρησιμοποιηθούν ως Υπηρεσίες σε μια SOA Αρχιτεκτονική προσαρμοσμένη σε Computational Grids. Όπως αναφέρθηκε στην ενότητα 2.2.2, από την έκδοση 4 του Globus Toolkit τα Web Services χρησιμοποιούνται ως τεχνολογική υποδομή για την παροχή υπηρεσιών στο Grid. Τα Grid βασίζονται σε δύο έννοιες: τους κατανεμημένους πόρους (**resources**) και τις υπηρεσίες (**services**). Η διαχείριση πόρων απαιτεί υποστήριξη κατάστασης (**stateful resources**) ενώ τα Web Services και

το πρωτόκολλο ανταλλαγής μηνυμάτων SOAP δεν την προσφέρουν (**stateless protocols**).

Η επέκταση των Web Services ώστε να χειρίζονται καταστάσεις (states) σε πόρους (resources) γίνεται με το πρωτόκολλο **Web Services Resource Framework – WSRF [Bank06]**. Το WSRF παρέχει ένα σύνολο από λειτουργίες που μπορούν να υλοποιηθούν από το Web Service για να αποκτήσει κατάσταση (stateful). Στην περιγραφή WSDL μιας υπηρεσίας ορίζει ένα έγγραφο XML, το ***Resource Properties***, το οποίο περιγράφει την κατάσταση (state) ενός ***WS-Resource*** (στην ορολογία WSRF ο πόρος αποκαλείται WS-Resource [Grah06b]). Το Resource Properties είναι διαφορετικό ανά υπηρεσία και σχετίζεται με τις ιδιότητες του πόρου (WS-Resource) που αυτή αντιπροσωπεύει. Ο έλεγχος της κατάστασης του πόρου γίνεται με την προσθήκη στο WS των εξής λειτουργιών πάνω στο Resource Properties:

- GetResourceProperty ή GetMultipleResourceProperties που επιστρέφει μια τιμή ενός συγκεκριμένου ή πολλαπλών στοιχείων (XML Elements) του Resource Property,
- QueryResourceProperties που επιτρέπει την εκτέλεση ενός ερωτήματος (π.χ. XPATH) στο Resource Properties,
- UpdateResourceProperties που επιτρέπει την ενημέρωση τιμών συγκεκριμένων στοιχείων του Resource Properties,
- InsertResourceProperties που επιτρέπει την εισαγωγή τιμής συγκεκριμένων νέων στοιχείων στο Resource Properties,
- DeleteResourceProperties που αφαιρεί τιμές από συγκεκριμένα στοιχεία και τέλος το
- SetResourceProperties που επιτρέπει συνδυασμό από διαγραφές, εισαγωγές και ενημερώσεις σε στοιχειά του Resource Properties.

3. Αρχιτεκτονικές Ασφαλείας

3.1. Εισαγωγή

Στο παρόν Κεφάλαιο θα μελετήσουμε τα θέματα ασφάλειας που άπτονται των ενδιαφερόντων της διατριβής. Συγκεκριμένα, θα δούμε τα βασικά θέματα των λειτουργιών και αρχιτεκτονικών ασφαλείας, θα αναφερθούμε στην βασική θεωρία της κρυπτογραφίας, τους κύριους μηχανισμούς ταυτοποίησης και τις σχετικές τεχνολογίες Web Services Security που θα μας απασχολήσουν στην διατριβή.

Η Ασφάλεια των Πληροφοριακών Συστημάτων (Information Security) ασχολείται με την ασφάλεια των πληροφοριών. Ακολουθεί ως αρχές την "**CIA Triad" - Confidentiality-Integrity-Availability** (Εμπιστευτικότητα, Ακεραιότητα, Διαθεσιμότητα). Ωστόσο έχουν προταθεί και εναλλακτικές ταξινομήσεις των βασικών αρχών, όπως η **Parkerian Hexad** [Park98]. Αυτή επεκτείνει την τριάδα CIA προτείνοντας έξι ατομικά στοιχεία (atomic elements), τα οποία δεν μπορούν να διασπασθούν σε απλούστερα. Αυτά είναι: **Confidentiality, Possession, Integrity, Authenticity, Availability και Utility** (Εμπιστευτικότητα, Κατοχή, Ακεραιότητα, Αυθεντικότητα, Διαθεσιμότητα και Χρήση). Όλες οι άλλες λειτουργίες ασφάλειας είναι παράγωγα αυτών των ατομικών στοιχείων.

Ο ορισμός Αρχιτεκτονικής Ασφαλείας σύμφωνα με το [OSA2010] μπορεί να αποδοθεί σαν:

> ***Η Αρχιτεκτονική Ασφαλείας είναι τα σχεδιαστικά στοιχεία που περιγράφουν πώς οι έλεγχοι ασφάλειας (=μέτρα ασφαλείας) τοποθετούνται και σχετίζονται με την συνολική Αρχιτεκτονική του Συστήματος. Αυτοί οι έλεγχοι υπηρετούν τον σκοπό της διατήρησης των χαρακτηριστικών ποιότητας του συστήματος, ανάμεσα τους: την εμπιστευτικότητα, ακεραιότητα, διαθεσιμότητα, υπευθυνότητα και διαβεβαίωση (confidentiality, integrity, availability, accountability and assurance)***.

Μια αρχιτεκτονική ασφαλείας υλοποιεί ένα σύστημα ασφαλείας. Σημαντική ιδιότητά του είναι:

> ***«Ένα σύστημα ασφαλείας είναι ισχυρό όσο και ο πιο αδύνατος κρίκος του»*** *[Ferg10]*

Η παραπάνω ιδιότητα πρέπει να λαμβάνεται πάντα υπόψη από τον σχεδιαστή της αρχιτεκτονικής.

Στα πλαίσια της διατριβής θα ασχοληθούμε με το τμήμα της Ασφάλειας των Πληροφοριακών Συστημάτων που αφορά στην **Ασφάλεια των Επικοινωνιών**, δηλαδή την ασφαλή ανταλλαγή δεδομένων μέσω δικτύου. Θα επικεντρωθούμε στα αντίστοιχα πρωτόκολλα και μηχανισμούς

Στην συνέχεια του κεφαλαίου θα δούμε ένα ευρύ σύνολο λειτουργιών ασφαλείας και τα είδη κρυπτογραφίας που βασίζονται, θα αναφερθούμε στους μηχανισμούς ταυτοποίησης που αφορούν την παρούσα διατριβή και τέλος στις τεχνολογίες/μηχανισμούς προστασίας μηνυμάτων SOAP (Web Services) που άπτονται των ενδιαφερόντων της διατριβής.

3.2. Λειτουργίες Ασφαλείας

Όλες οι λειτουργίες ασφάλειας βασίζονται στην κρυπτογραφία. Αποτελούν τις πρωταρχικές λειτουργίες που χρησιμοποιεί ο σχεδιαστής μιας εφαρμογής ή ενός συστήματος βάση των προδιαγραφών που του θέτονται. Οι βασικές λειτουργίες ασφαλείας παρουσιάζονται στον Πίνακα 1.

Πίνακας 1: Λειτουργίες Ασφάλειας Πληροφοριακών Συστημάτων [Mene97]

Λειτουργίες Ασφάλειας	Επεξήγηση
Ιδιωτικότητα ή Εμπιστευτικότητα (Privacy or Confidentiality)	**Η δυνατότητα που έχει μια οντότητα να διατηρεί την πληροφορία κρυφή από όλους, εκτός από αυτούς που είναι διαπιστευμένοι να την δουν**
Ακεραιότητα Δεδομένων (Data Integrity)	**Η λειτουργία της επιβεβαίωσης ότι η πληροφορία δεν έχει αλλαχθεί από μη εξουσιοδοτημένη οντότητα ή με άγνωστο τρόπο**
Ταυτοποίηση οντότητας (Entity Authentication or Identification)	**Η επιβεβαίωση της ταυτότητας οντότητας (πχ ατόμου, υπολογιστή, πιστωτικής κάρτας κτλ).**
Ταυτοποίηση Μηνύματος (Message Authentication)	**Η επιβεβαίωση της αυθεντικότητας της πληροφορίας που μεταδίδεται σε ένα μήνυμα, δηλαδή η επιβεβαίωση της ταυτότητας του αποστολέα.**
Υπογραφή (Signature)	**Ο μηχανισμός που «δένει» πληροφορία με μια οντότητα**

Εξουσιοδότηση ή Έγκριση (Authorization)	**Η μεταβίβαση σε μια άλλη οντότητα της δυνατότητας να κάνει κάποια συγκεκριμένη ενέργεια στην πληροφορία**
Επικυροποίηση (Validation)	**Ένας τρόπος για να έγκριση προς μια οντότητα σε κάποιο συγκεκριμένο χρόνο να χρησιμοποιήσει ή να διαχειριστεί πληροφορία ή πόρους**
Έλεγχος Πρόσβασης (Access Control)	**Επιτρέπει την πρόσβαση στους πόρους (που ελέγχει) σε προνομιούχες οντότητες.**
Πιστοποίηση (Certification)	**Η εσώκλειση πληροφορίας από μια έμπιστη οντότητα.**
Χρονοσήμανση (Timestamping)	**Η καταγραφή του χρόνου δημιουργίας ή ύπαρξης της πληροφορίας**
Μαρτυρία (Witnessing)	**Η επαλήθευση της δημιουργίας ή της ύπαρξης πληροφορίας από μια οντότητα διαφορετική από τον δημιουργό της**
Απόδειξη (Receipt)	**Η επιβεβαίωση ότι η πληροφορία έχει ληφθεί**
Επιβεβαίωση (Confirmation)	**Η επιβεβαίωση ότι οι υπηρεσίες έχουν προσφερθεί**
Κατοχή (Ownership)	**Ένας τρόπος που παρέχει σε μία οντότητα που κατέχει το νομικό δικαίωμα της κατοχής, να χρησιμοποιεί ή να μεταβιβάζει ένα πόρο σε τρίτους**
Ανωνυμία (Anonymity)	**Η απόκρυψη της ταυτότητας μια οντότητας που συμμετέχει σε μία διαδικασία**
Μη Άρνηση Ενεργειών (Non-Repudiation)	**Η εμπόδιση της άρνησης παρελθόντων υποχρεώσεων ή πράξεων**
Άρση Εμπιστοσύνης (Revocation)	**Η άρση της Πιστοποίησης ή Έγκρισης**

Οι παραπάνω λειτουργίες ασφάλειας είναι παράγωγα των πρωταρχικών αρχών (τριάδα CIA και Parkerian Hexad) που είδαμε στην ενότητα 3.1. Υπόστρωμά τους αποτελεί η κρυπτογραφία και αυτές με την σειρά τους αποτελούν τα δομικά κομμάτια οποιασδήποτε αρχιτεκτονικής ασφαλείας. Όσον αφορά την κρυπτογραφία, υπάρχουν δύο βασικές προσεγγίσεις***: Κρυπτογραφία Συμμετρικού Κλειδιού*** και η ***Κρυπτογραφία Δημοσίου Κλειδιού***. Τα είδη αυτά θα τα παρουσιάσουμε στην επόμενη ενότητα.

3.3. Είδη κρυπτογραφίας

Η κρυπτογραφία είναι η επιστήμη και η τέχνη της απόκρυψης της πληροφορίας. Χρησιμοποιείται για την ταυτοποίηση οντοτήτων (χρηστών και υπηρεσιών), υλοποίηση ψηφιακών υπογραφών και πολλών άλλων λειτουργιών ασφαλείας όπως είδαμε στον Πίνακα 1.

Η κρυπτογράφηση είναι ουσιαστικά η πράξη της εφαρμογής ενός **αλγόριθμου κρυπτογράφησης (cipher)** σε ένα **καθαρό κείμενο (cleartext)** με βάση κάποιο κλειδί (key) και έχει ως αποτέλεσμα το **κρυπτογραφημένο κείμενο (ciphertext)**. Το

καθαρό κείμενο μπορεί να ανακτηθεί από το κρυπτογραφημένο μόνο με την εφαρμογή του αλγορίθμου και του κατάλληλου κλειδιού στο κρυπτογραφημένο κείμενο.

Υπάρχουν ουσιαστικά δύο βασικές κατηγορίες αλγορίθμων κρυπτογραφίας που χρησιμοποιούνται στην επιστήμη της Πληροφορικής: Η ***Κρυπτογραφία Συμμετρικού Κλειδιού*** και η ***Κρυπτογραφία Δημοσίου Κλειδιού***. Στην συνέχεια της ενότητας παρουσιάζουμε τα δύο είδη εν συντομία.

3.3.1. Κρυπτογραφία Συμμετρικού Κλειδιού

Στην συμμετρική κρυπτογραφία οι συμμετέχοντες στην ανταλλαγή των κρυπτογραφημένων δεδομένων μοιράζονται το ίδιο κλειδί. Οι συμμετέχοντες στην ανταλλαγή μηνυμάτων (αποστολέας και παραλήπτης) ανταλλάσσουν μηνύματα κρυπτογραφημένα με συμφωνηθέντα αλγόριθμο και τα αποκρυπτογραφούν με το κοινό κλειδί.

Υπάρχουν ουσιαστικά τρία είδη αλγορίθμων κρυπτογράφησης συμμετρικού κλειδιού:

- **Αλγόριθμοι που λειτουργούν σε τμήματα (block ciphers)**
- **Αλγόριθμοι που λειτουργούν σε ροή πληροφορίας (stream ciphers)**
- **Αλγόριθμοι επιβεβαίωσης ταυτότητας μηνύματος (Hashed Message Authentication Codes)**

Οι **αλγόριθμοι τμημάτων (block ciphers)** χρησιμοποιούν ως είσοδο ένα τμήμα (block) προκαθορισμένου μεγέθους του καθαρού κειμένου και εφαρμόζοντας το συμμετρικό κλειδί παράγουν ένα τμήμα του κρυπτογραφημένου κειμένου ίδιου μεγέθους. Μερικοί από τους συνηθισμένους αλγόριθμους τμημάτων είναι οι **Data Encryption Standard (DES)** και **triple-DES** [FIPS46-3], **Advanced Encryption Standard (AES)** [FIPS197] [Daem99], **Blowfish** [Schn94], **CAST-128** [Adam97], **RC5** [Rive94] και **IDEA** [Xuej91].

Στους **αλγορίθμους ροής πληροφορίας (stream ciphers)**, η έξοδος του αλγορίθμου είναι μια κρυπτογραφημένη ροή που δημιουργείται από την συσχέτιση (correlation) σε επίπεδο bit ή χαρακτήρα του καθαρού κειμένου (ως ροή) με ένα απλό τυχαίο κείμενο, το οποίο αποτελεί την εσωτερική κατάσταση του αλγορίθμου και αλλάζει δυναμικά κατά την διάρκεια του χρήσης του αλγορίθμου. Η κατάσταση αυτή

αρχικοποιείται με το κλειδί κρυπτογράφησης. Γνωστός αλγόριθμος ροής είναι ο **RC4** [Rive87], ο οποίος χρησιμοποιείται στο **SSL/TLS [**Dier06**]** και στο **WEP [WEP]**. Είναι ιδιαίτερα ταχύς αλγόριθμος, ωστόσο παρουσιάζει αδυναμία στην αρχικοποίηση του και στα ψευδοτυχαία κλειδιά που χρησιμοποιεί [Fluh01]. Επίσης μπορεί να χρησιμοποιηθεί ως αλγόριθμος ροής και κάποιος αλγόριθμος block cipher τροποποιώντας όμως τον τρόπο λειτουργίας του, π.χ. ο AES, χρησιμοποιείται στο πρωτόκολλο ασφάλειας WPA2 [WPA2] [CCMP]των ασύρματων δικτύων Wi-Fi.

Οι αλγόριθμοι **Hashed Message Authentication Codes** (**HMACs**) [RSAfaq] παράγουν πληροφορία ταυτοποίησης κάθε μηνύματος, η οποία και προστίθεται στο κυρίως μήνυμα. Ένα HMAC παράγεται με την κρυπτογράφηση του αποτελέσματος κατακερματισμού του μηνύματος με την χρήση συμμετρικού αλγορίθμου και κλειδιού. Με τον τρόπο αυτό διασφαλίζεται η ακεραιότητα του μηνύματος από παρεμβάσεις τρίτων. Εάν επιπλέον απαιτείται και εμπιστευτικότητα στην μετάδοση του μηνύματος θα πρέπει να κρυπτογραφηθεί και το κυρίως μήνυμα.

Οι **συναρτήσεις κατακερματισμού (hash functions)** που χρησιμοποιούνται από τον αλγόριθμο HMAC έχουν ως είσοδο ένα μήνυμα οποιουδήποτε μεγέθους, και έξοδο μια σταθερού μήκους τιμή κατακερματισμού (digest) μεταξύ 128 και 512 bits, ανάλογα τον αλγόριθμο. Βασικό χαρακτηριστικό των συναρτήσεων κατακερματισμού είναι η αδυναμία να βρεθεί το αρχικό μήνυμα από την τιμή κατακερματισμού. Ωστόσο οι συναρτήσεις κατακερματισμού παρουσιάζουν το μειονέκτημα των συγκρούσεων, δηλαδή μπορεί δύο διαφορετικά μηνύματα να έχουν την ίδια τιμή κατακερματισμού. Ως αλγόριθμοι αντιμετωπίζουν πρόβλημα ασφάλειας όταν ένας τρίτος (hacker) μπορεί για μια τιμή κατακερματισμού να βρίσκει το αρχικό μήνυμα. Αλγόριθμοι ασθενείς σε επιθέσεις [Wang04b] είναι οι παλαιότεροι **MD4** [Rive92a] και **MD5** [Rive92b]. Ισχυρότερος είναι ο **SHA-1** [FIPS180-2] αλλά προτείνονται ακόμα ισχυρότεροι όπως ο **SHA-256** [FIPS180-2].

Η κρυπτογραφία συμμετρικού κλειδιού αποτελεί την βασική λύση που χρησιμοποιείται από την πλειοψηφία των μηχανισμών ασφαλείας. Γενικά, η εφαρμογή κρυπτογραφικών αλγορίθμων σε ένα μήνυμα επιβαρύνει την απόδοση του συστήματος που τις εκτελεί. Ωστόσο οι συμμετρικοί αλγόριθμοι εμφανίζουν σαφώς καλύτερη απόδοση από τους αλγορίθμους δημοσίου κλειδιού που θα δούμε στην συνέχεια. Όμως, το βασικό πρόβλημα που αντιμετωπίζει ο σχεδιαστής ενός συστήματος ασφαλείας, όταν χρησιμοποιεί συμμετρική κρυπτογραφία, είναι να

προδιαγράψει ένα ασφαλή τρόπο να διανομής των συμμετρικών κλειδιών κρυπτογράφησης. Το πρόβλημα αυτό αντιμετωπίζεται με διάφορους τρόπους π.χ. χρήση κρυπτογραφίας δημοσίου κλειδιού για την μετάδοση συμμετρικών κλειδιών (SSL/TLS) όπως θα δούμε στην επόμενη ενότητα ή με την χρήση του πρωτοκόλλου Kerberos (ενότητα 3.4.2).

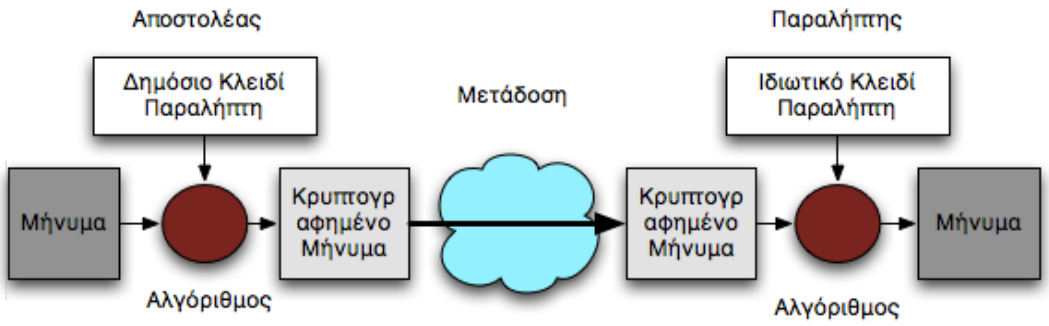
3.3.2. Κρυπτογραφία Δημοσίου Κλειδιού

Η κρυπτογραφία Δημοσίου Κλειδιού (Public Key Cryptography) αποτελεί σχετικά νέα εξέλιξη στο χώρο της κρυπτογραφίας (1976) και ήταν καθοριστική για τη μεγάλη διάδοση των ηλεκτρονικών συναλλαγών μέσω δικτύων και Internet. Βασίστηκε στην σχεδόν παράλληλη, αλλά ανεξάρτητη έρευνα των **Whitfield Diffie** και **Martin Hellman** και του **Ralph C. Merkle**, οι οποίοι πρότειναν τα πρωτόκολλα ανταλλαγής κλειδιών **Diffie-Hellman key exchange** [Diff76a] και **Merkle's Puzzles** [Merk78] αντίστοιχα.

Η σημαντική λειτουργική διαφορά της κρυπτογραφίας **Δημοσίου Κλειδιού** σε σχέση με την κρυπτογραφία συμμετρικού είναι ότι δεν απαιτείται οι συναλλασσόμενοι να ανταλλάσσουν εμπιστευτικά κλειδιά ανά δύο. Βασίζονται σε ασύμμετρους αλγόριθμους, οι οποίοι, δημιουργούν **ένα μαθηματικά συσχετιζόμενο ζευγάρι κλειδιών**, του **ιδιωτικού** και του **δημόσιου**. Είναι δομημένοι με τέτοιο τρόπο ώστε ο υπολογισμός του ιδιωτικού κλειδιού από το δημόσιο να είναι υπολογιστικά αδύνατος, αν και τα δύο κλειδιά είναι μεταξύ τους συσχετισμένα. Βασίζονται σε υπολογιστικά δύσκολα προβλήματα (NP-Hard), όπως στο μαθηματικό πρόβλημα διακριτών λογαρίθμων (discrete logarithm problem) της Θεωρίας των Αριθμών, του αλγόριθμου Diffie-Hellman.

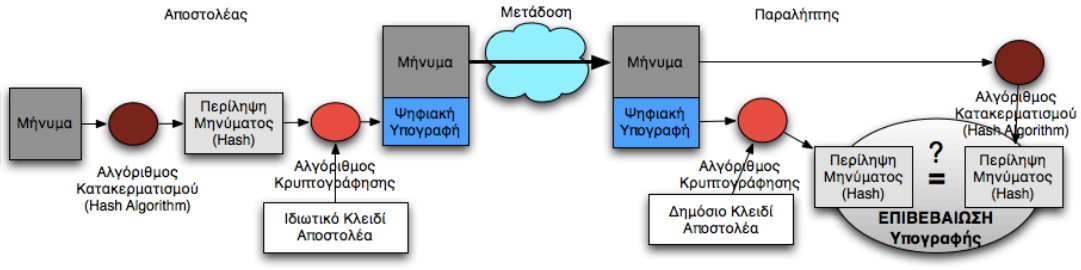
Μοιράζοντας το δημόσιο κλειδί σε όλους τους δυνατούς παραλήπτες, και κρατώντας προστατευμένο το ιδιωτικό κλειδί, είναι δυνατόν να υλοποιηθούν όλες οι βασικές λειτουργίες ασφάλειας:

- **Εμπιστευτικότητα (Confidentiality)**: (Σχήμα 6) Ο αποστολέας κρυπτογραφεί το μήνυμα με το δημόσιο κλειδί του παραλήπτη. Το μήνυμα μπορεί να αποκρυπτογραφηθεί μόνο με το ιδιωτικό κλειδί του παραλήπτη.



Σχήμα 6: Εμπιστευτικότητα (Confidentiality) με την χρήση Κρυπτογραφίας Δημοσίου Κλειδιού

- **Ακεραιότητα (Integrity) και Ταυτοποίηση Μηνύματος (Message Authenticity)**: Γίνεται μέσω ηλεκτρονικών υπογραφών. Ο αποστολέας παράγει μια ηλεκτρονική υπογραφή, δηλαδή κρυπτογραφεί με το ιδιωτικό του κλειδί το αποτέλεσμα μιας συνάρτησης κατακερματισμού πάνω στο μήνυμα. Στην συνέχεια ενσωματώνει την υπογραφή στο μήνυμα. Τέλος ο παραλήπτης μπορεί να επιβεβαιώσει μέσω του δημοσίου κλειδιού του αποστολέα και της υπογραφής του μηνύματος ότι το μήνυμα προέρχεται από τον αποστολέα (ταυτοποίηση μηνύματος) και δεν έχει τροποποιηθεί (ακεραιότητα) κατά την μεταφορά. Η όλη διαδικασία περιγράφεται στο Σχήμα 7. Τον ρόλο των ψηφιακών υπογραφών όταν έχουμε χρήση συμμετρικής κρυπτογραφίας γίνεται μέσω των HMAC όπως είδαμε. Σε σχέση με τις ψηφιακές υπογραφές, υπάρχει μια σημαντική διαφορά: ο υπολογισμός και η επιβεβαίωση του HMAC γίνεται με το ίδιο κλειδί. Επομένως μόνο ο παραλήπτης μπορεί να επιβεβαιώσει το μήνυμα. Στις ψηφιακές υπογραφές την δυνατότητα αυτή την έχουν όλοι μέσω του δημοσίου κλειδιού.



Σχήμα 7: Ταυτοποίηση και Ακεραιότητα Μηνύματος με την χρήση Ψηφιακών Υπογραφών

Οι αλγόριθμοι κρυπτογράφησης γνωρίζουν ευρεία διάδοση. Ο αλγόριθμος Diffie-Hellman έχει προταθεί από τον οργανισμό IETF ως ένας μηχανισμός που μπορεί να χρησιμοποιηθεί για ανταλλαγή κλειδιών στο Internet (**Internet Key**

Exchange – IKE [Kivi03]). Ωστόσο, ο περισσότερο διαδεδομένος αλγόριθμος είναι ο **RSA** [Rive78]. Βασίζεται στο πρόβλημα παραγοντοποίησης ακεραίων αριθμών (integer factorization problem). Χρησιμοποιείται κυρίως στο ηλεκτρονικό εμπόριο και είναι ο πρώτος αλγόριθμος δημοσίου κλειδιού κατάλληλος για κρυπτογράφηση και ηλεκτρονική υπογραφή. Οφείλει ιστορικά την επικράτησή του ως ο κυρίαρχος αλγόριθμος δημοσίου κλειδιού στην δημιουργία Αρχής Πιστοποίησης (Certification Authority) βασισμένη σε αυτόν από την RSA Security. Η **RSA Security** που κατέχει τα δικαιώματα (patent) του αλγορίθμου, δημιούργησε (spinoff) την **VeriSign** [VeriSign], η οποία κατέχει δεσπόζουσα θέση ανάμεσα στις Αρχές Πιστοποίησης.

Λόγω της ασύμμετρης πολυπλοκότητας των αλγορίθμων που χρησιμοποιεί η κρυπτογραφία δημοσίου κλειδιού, εμφανίζει υψηλές υπολογιστικές απαιτήσεις τόσο στην κρυπτογράφηση όσο και στην αποκρυπτογράφηση. Εξαιτίας αυτού, κατά την ανταλλαγή μηνυμάτων, οι αλγόριθμοι δημοσίου κλειδιού χρησιμοποιούνται για την κρυπτογράφηση ενός συμμετρικού κλειδιού, το οποίο έχει χρησιμοποιηθεί μαζί με ένα συμμετρικό αλγόριθμο για κρυπτογράφηση του μηνύματος***. Όσον αφορά την απόδοσή τους, η RSA Security [RSAPerf] αναφέρει ότι η πολυπλοκότητα του RSA είναι $O(k^2)$ για λειτουργίες που βασίζονται στο δημόσιο κλειδί (κρυπτογράφηση και επιβεβαίωση υπογραφής), $O(k^3)$ για τις λειτουργίες που βασίζονται στο ιδιωτικό κλειδί (αποκρυπτογράφηση και δημιουργία υπογραφής), και $O(k^4)$ για την παραγωγή κλειδιών (ιδιωτικού και δημοσίου). Συγκρίνοντας τον με τον συμμετρικό αλγόριθμο DES, αυτός είναι τουλάχιστον 100 φορές πιο γρήγορος από τον RSA όταν υλοποιούνται με λογισμικό και 1.000 ως 10.000 γρηγορότερος χρησιμοποιώντας υλοποιήσεις που βασίζονται σε hardware.***

Το μειονέκτημα που παρουσιάζει η κρυπτογραφία δημοσίου κλειδιού, όσον αφορά την απόδοσή της στις διάφορες κρυπτογραφικές λειτουργίες, το αντισταθμίζει με την ευκολία που προσφέρει στην ανταλλαγή κλειδιών στην υλοποίηση του μηχανισμού της ηλεκτρονικής υπογραφής επιτρέπει τις εξής λειτουργίες ασφαλείας: Ταυτοποίηση, Επιβεβαίωση Λήψης (receipt) και μη Άρνηση Ενεργειών (non repudiation).

Στην συνέχεια του κεφαλαίου θα δούμε τους βασικούς μηχανισμούς Ταυτοποίησης που ασχολείται η διατριβή. Αυτοί βασίζονται σε λειτουργίες κρυπτογραφίας και προσφέρουν την απαραίτητη διανομή κλειδιών μεταξύ

συναλλασσομένων ώστε να επιτελεστούν οι αναγκαίες λειτουργίες που ορίζει μια αρχιτεκτονική ασφαλείας.

3.4. Πρωτόκολλα και Συστήματα Ταυτοποίησης

Η λειτουργία της ταυτοποίησης (Authentication) μιας οντότητας βασίζεται στην επιβεβαίωση συγκεκριμένων παραγόντων (factors). **Παράγοντες ταυτοποίησης** μπορεί να είναι η επίδειξη μυστικής πληροφορίας (π.χ. password), η κατοχή κάποιου αυστηρά προσωπικού αντικειμένου (π.χ. smartcard, hasp), βιομετρικό αποτύπωμα, μαρτυρία τρίτης έμπιστης οντότητας (π.χ. Third Trusted Party) [Brai06].

Η Ταυτοποίηση αποτελεί βασική λειτουργία σε οποιαδήποτε αρχιτεκτονική ασφαλείας. Η Ταυτοποίηση μπορεί να χρησιμοποιεί έναν παράγοντα - weak authentication (π.χ. username-password), δύο παράγοντες – two-factor authentication (username-password και one-time password) ή και παραπάνω παράγοντες. Αυξάνοντας τους παράγοντες γενικά αυξάνεται το επίπεδο ασφάλειας της Ταυτοποίησης.

Σημαντική επίσης είναι η λειτουργία **Single Sign On - SSO**. Η λειτουργία αυτή είναι αναγκαία σε κατανεμημένα συστήματα όπως τα Grids γιατί επιτρέπει σε μια οντότητα να ταυτοποιείται μία φορά και να αποκτά πρόσβαση σε πολλά συστήματα για ένα ορισμένο χρονικό διάστημα.

Ιδιαίτερο ενδιαφέρον εμφανίζουν λύσεις σε ομοσπονδιακό περιβάλλον (**Federated Authentication**) όπου η ταυτοποίηση σε ένα περιοχή ασφαλείας κοινοποιείται σε συνδεόμενες περιοχές. Τέτοιες λύσεις είναι ευρέως διαδεδομένες στο Internet, όπως το **OpenID** [OpenID] να υποστηρίζεται από όλους τους μεγάλους παρόχους υπηρεσιών και ιστοχώρων στο Διαδίκτυο [OIDMEMBER]. Το OpenID είναι ένα ανοιχτό πρότυπο που επιτρέπει στους χρήστες, με την ταυτότητα που έχουν σε ένα πάροχο υπηρεσίας, να ταυτοποιούνται σε τρίτες υπηρεσίες και ιστοχώρους, επιτρέποντας να έχουν ουσιαστικά μια κοινή ταυτότητα. Η διάδοση του είναι μεγάλη φτάνοντας το ένα δισεκατομμύριο λογαριασμούς [Kiss09] και υποστήριξη τους από εννέα εκατομμύρια web sites.

Οι ανοιχτές λύσεις τύπου OpenID δεν προορίζονται για επιχειρησιακά περιβάλλοντα, γιατί βασίζονται στην παραδοχή ότι η πραγματική πιστοποίηση της ταυτότητας του χρήστη δεν είναι κρίσιμη ιδιότητα, αλλά ενδιαφέρον παρουσιάζει η

μοναδικότητα μιας δηλωθείσας ταυτότητας. Επίσης δεν υποστηρίζονται συστήματα και υπηρεσίες, αλλά μόνο χρήστες.

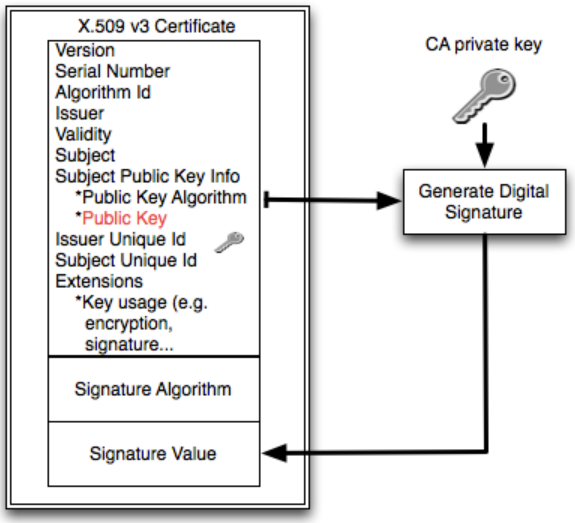
Στην διατριβή, ωστόσο, δεν χρησιμοποιήσαμε τέτοια πρωτόκολλα λόγο του περιβάλλοντος που κινείται και καλείται να δώσει λύση. Τα Computational Grids βασίζονται σε τεχνολογίες Δημοσίου Κλειδιού για την ταυτοποίηση και τις κρυπτογραφικές λειτουργίες τους. Άλλα περιβάλλοντα κατανεμημένης επεξεργασίας χρησιμοποιούν πρωτόκολλα ταυτοποίησης με βάση το πρωτόκολλο Kerberos (π.χ. Windows Active Directory). Η λύση που προτείναμε στην διατριβή είναι συνδυασμός των δύο ανωτέρων αρχιτεκτονικών. Στην συνέχεια της παρούσης ενότητας θα δούμε με μεγαλύτερη λεπτομέρεια αυτές τις τεχνολογίες.

3.4.1. Υποδομή Δημοσίου Κλειδιού (Public Key Infrastructure - PKI)

Η κρυπτογραφία Δημοσίου Κλειδιού για να γίνει λειτουργική σε κατανεμημένα περιβάλλοντα χρειάζεται την απαραίτητη **Υποδομή Δημοσίου Κλειδιού (Public Key Infrastructure – PKI)**. Η υποδομή είναι αναγκαία, γιατί παίζει τον ρόλο της **Τρίτης Έμπιστης Οντότητας** (**Third Trusted Party – TTP**) που οι συνδιαλλασσόμενοι γνωρίζουν και εμπιστεύονται εκ των προτέρων. Προπομπός θεωρήθηκε η χρήση Δημόσιου Αρχείου [Diff76b] και η δουλεία του Kohnfelder [Kohn78]. Σε επόμενο στάδιο προτάθηκαν και τυποποιήθηκαν τα Πιστοποιητικά X.509 [ITU05], [Adam99] που χρησιμοποιήθηκαν αρχικά για ταυτοποίηση πρόσβασης στους καταλόγους X.500 [X.500].

Η επιρροή από το X.500 είναι φανερή στα πιστοποιητικά X.509, εξαιτίας της ιεραρχικής δομής που ακολουθούν. Η ιεραρχική δομή αποτελείται από μια ακολουθία κόμβων που ονομάζονται Relative Distinguished Names – RDN οι οποίοι σχηματίζουν ένα Distinguished Name – DN (π.χ. o=ntua, ou=ece, ou=netmode, cn=moralis). Το DN αποτελεί το μοναδικό αναγνωριστικό του Πιστοποιητικού X.509.

Ένα Πιστοποιητικό X.509 V3 αποτελείται από τρία τμήματα [Hous02], Certificate, Certificate Signature Algorithm και Certificate Signature Value όπως παρουσιάζεται στο Σχήμα 8.



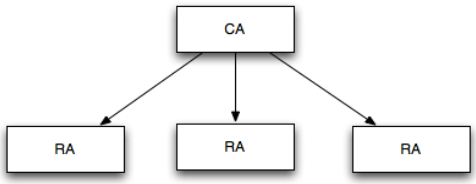
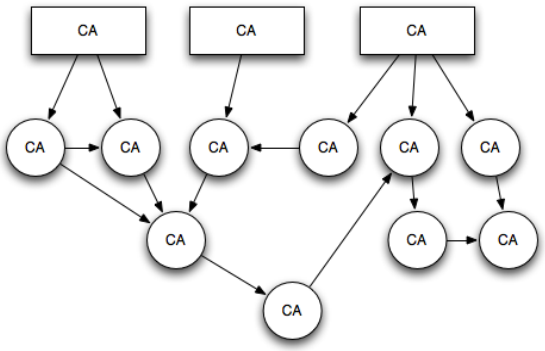
Σχήμα 8: Δομή Πιστοποιητικού X.509 v3

Τα πεδία των τριών τμημάτων είναι:

- **Πιστοποιητικό (Certificate)**
  - Version: Η έκδοση που ακολουθεί το Πιστοποιητικό (v1, v2 ή v3)
  - serialNumber: Ο σειριακός αριθμός του Πιστοποιητικού
  - algorithm id: Το αναγνωριστικό των αλγορίθμων (κατακερματισμού και κρυπτογράφησης) που χρησιμοποιήθηκαν για την υπογραφή π.χ. md5WithRSAEncryption
  - issuer: Η Εκδούσα Αρχή (Certification Authority)
  - validity: Οι ημερομηνίες έναρξης και λήξης ισχύος του Πιστοποιητικού
  - subject: Το όνομα του κατόχου του Πιστοποιητικού
  - subjectPublicKeyInfo: Ο αλγόριθμος του δημοσίου κλειδιού και το **δημόσιο κλειδί**
  - issuerUniqueId: Το μοναδικό αναγνωριστικό της εκδούσας αρχής
  - subjectUniqueID: Το μοναδικό αναγνωριστικό του Πιστοποιητικού (DN)
  - extensions: Επεκτάσεις, όπως πεδίο χρήσης του κλειδιού (key usage) κτλ.
- **Αλγόριθμος Υπογραφής Πιστοποιητικού (Certificate Signature Algorithm):** Ο αλγόριθμος που χρησιμοποιήθηκε από την Εκδούσα Αρχή (Certification Authority) για την υπογραφή του πιστοποιητικού
- **Κείμενο Ψηφιακής Υπογραφής (Certificate Signature Value):** Η τιμή της υπογραφής (κρυπτογραφημένο κείμενο) με την οποία η Εκδούσα Αρχή επιβεβαιώνει την εγκυρότητα του πιστοποιητικού

Στην υποδομή PKI τον ρόλο της Third Trusted Party τον παίζει η Αρχή Πιστοποίησης - Certification Authority (CA). Βρίσκεται συνήθως εκτός δικτύου και υπογράφει τα πιστοποιητικά των χρηστών και υπηρεσιών. Άλλη βασική αρχή είναι **η Αρχή Εγγραφής (Registration Authority - RA)**, η οποία πιστοποιεί τους χρήστες, παραλαμβάνει την αίτηση πιστοποιητικού και την προωθεί στην Αρχή Πιστοποίησης. Το γεγονός, ότι η Αρχή Πιστοποίησης είναι εκτός δικτύου, την προστατεύει από επιθέσεις. Ιδανικά θα ήταν επιθυμητό να υπήρχε μια παγκόσμια Αρχή Πιστοποίησης, ωστόσο, αυτό για ευνόητους λόγους (πολιτικούς, οργανωτικούς) αυτό είναι αδύνατο. Υπάρχουν ωστόσο πολλά **Μοντέλα Οργάνωσης Εμπιστοσύνης μεταξύ Υποδομών Δημοσίου Κλειδιού**. Συγκεκριμένα παρουσιάζονται [Perl99]: α) Απλό Μοντέλο μιας CA., β) μια CA με πολλαπλές RA, γ) Ολιγαρχία από CA και δ) μια CA που εκδίδει δικαιώματα σε άλλες CA. Τα μοντέλα αυτά απεικονίζονται στον Πίνακα 2.

Πίνακας 2: Μοντέλα Οργάνωσης Εμπιστοσύνης μεταξύ Υποδομών Δημοσίου Κλειδιού

α) Απλό Μοντέλο μιας CA	
β) μια CA με πολλαπλές RAs	
γ) Ολιγαρχία από CAs	
δ) μια CA που μεταβιβάζει δικαιώματα σε άλλες CAs	

Ένα βασικό θέμα του PKI είναι η ανάκληση ή ακύρωση ενός πιστοποιητικού. Αυτή γίνεται αυτόματα όταν αυτό λήξει ή όταν ανακληθεί. Η ανάκληση κοινοποιείται

μέσω **Λιστών Ανάκλησης Πιστοποιητικών** (**Certificate Revocation Lists - CRLs**). Ο μηχανισμός αυτός είναι αρκετά προβληματικός [Gutm02], γιατί η ανάκληση δεν γίνεται άμεσα και μπορεί να περάσει μεγάλο χρονικό διάστημα για να ενημερωθούν οι συνδιαλλασσόμενοι για την ανάκληση. Επίσης δεν υπάρχουν προτυποποιημένοι και αυτοματοποιημένοι μηχανισμοί ενημέρωσης χρηστών για τις αλλαγές που συμβαίνουν στις Revocations Lists και επαφίεται στους χρήστες να ενημερώνονται για ανακλημένα πιστοποιητικά. Συνήθως ο μέσος χρήστης αρκείται στα certificates των CAs (δημόσια κλειδιά) που είναι αποθηκευμένες στο browser του από το οποία επιβεβαιώνει την ταυτότητα ενός server μέσω του υπογεγραμμένου από μία CA server certificate. Αυτό από μόνο του δεν εξασφαλίζει ότι το server certificate είναι έγκυρο, γιατί μπορεί να έχει ανακληθεί και να βρίσκεται σε Revocation List.

Να σημειωθεί πως όταν δεν απαιτείται πιστοποίηση από Τρίτη Έμπιστη Οντότητα, μπορούν να εκδοθούν αυτόνομα πιστοποιητικά (self-signed Certificates) για ένα server, υπογράφοντας το πιστοποιητικό με το ιδιωτικό κλειδί του server.

Η υποδομή PKI αποτελεί την βάση ταυτοποίησης σε συστήματα Grid όπως θα δούμε στην ενότητα 3.4.3. Επιτρέπει την ταυτοποίηση του χρήστη σε πολλαπλούς πόρους μέσω της μεταβίβασης των διαπιστευτηρίων που προκύπτουν κατά την ταυτοποίηση του/της μέσα στο πλέγμα.

3.4.2. Kerberos

Το πρωτόκολλο Kerberos **βασίζεται στο πρωτόκολλο Needham-Schroeder** [Need78] και **δημιουργήθηκε στα πλαίσια του έργου Athena** [ATHENA] [Mill87] του MIT με σκοπό να παρέχει δικτυακή ταυτοποίηση χρηστών. Η πρώτη έκδοση του είναι η έκδοση 4 [Neum94], και ακολούθησε η έκδοση 5 [Neum05]. **Χρησιμοποιεί συμμετρική κρυπτογραφία**. Αποτελεί το πρωτόκολλο ταυτοποίησης σε Microsoft Active Directory [MSAC].

Το πρωτόκολλο βασίζει στην λειτουργία του στην ύπαρξη του Kerberos **Key Distribution Center (KDC)** που λειτουργεί σαν **Τρίτη Έμπιστη Οντότητα (Third Trusted Party - TTP)**. Το KDC ταυτοποιεί αμοιβαία χρήστες και υπηρεσίες, μοιράζοντάς τους ταυτόχρονα συμμετρικό κλειδί για την κρυπτογραφημένη επικοινωνία μεταξύ τους. Αποτελείται από δύο υπηρεσίες: α) *την υπηρεσία **AS (Authentication Server)** που ταυτοποιεί τον πελάτη* και β) *την υπηρεσία **TGS (Ticket Granting Server)** που εκδίδει τα διαπιστευτήρια* που στην περίπτωση του Kerberos

λέγονται εισιτήρια (service tickets) για την πρόσβαση σε συγκεκριμένες υπηρεσίες που ζητά ο πελάτης.

Η **ταυτοποίηση βασίζεται στη δομή του «εισιτηρίου» (Kerberos ticket)**. Ένα ticket του πρωτοκόλλου Kerberos είναι **μια συμμετρικά κρυπτογραφημένη δομή με το κλειδί της υπηρεσίας** για την οποία προορίζεται. Περιέχει τα εξής στοιχεία:

- Την ταυτότητα του πελάτη
- Την διεύθυνση του πελάτη
- Το κλειδί της συνόδου (session key)
- Μια χρονική σήμανση (timestamp), μέχρι την οποία είναι έγκυρο το κλειδί
- Flags του εισιτηρίου που σηματοδοτούν επιπλέον δυνατότητες

Το εισιτήρια, όπως αναφέραμε κρυπτογραφούνται με το κλειδί της υπηρεσίας, επομένως κανένας εκτός από την ίδια την υπηρεσία δεν μπορεί να εξάγει την πληροφορία που εσωκλείουν.

Το πρωτόκολλο **υποστηρίζει την ταυτοποίηση ενός χρήστη (Principal) σε διαφορετικές περιοχές ασφαλείας,** που στην ορολογία του πρωτοκόλλου Kerberos λέγονται **realms**. Το realm ενός χρήστη ακολουθεί το μοναδικό όνομά του. Για την **inter-realm** ταυτοποίηση απαιτείται οι διαχειριστές περιοχών ασφαλείας να έχουν εγκαταστήσει κρυπτογραφικό κλειδί για τέτοια χρήση. Επίσης οι περιοχές ασφαλείας μπορεί να οργανωθούν ιεραρχικά με τη κάθε περιοχή να μοιράζεται με τον γονέα της (parent) κοινό κλειδί. Αν δεν ακολουθείται ιεραρχική οργάνωση, θα χρειαστεί αναζήτηση σε βάση που διατηρεί τις σχέσεις εμπιστοσύνης (trust) μεταξύ των περιοχών ασφαλείας (realms).

Στα **flags ενός εισιτηρίου** περιλαμβάνεται η πληροφορία για το αν είναι έγκυρο (valid), ανανεώσιμο (renewable), μεταχρονολογημένο (postdated), πληρεξούσιο (proxy), προωθούμενο (forwardable) κ.α.

Στην συνέχεια **θα εξηγήσουμε την λειτουργία του πρωτοκόλλου Kerberos** και των συστημάτων/υπηρεσιών που το αποτελούν σε βήματα:

a. Ο πελάτης πιστοποιείται στον Authentication Server (AS) του KDC και επιδεικνύει τα διαπιστευτήρια του. Ως απάντηση λαμβάνει δύο μηνύματα: i) ένα κλειδί συνόδου πελάτη/TGS κρυπτογραφημένο με το συμμετρικό κλειδί του και ii) ένα ειδικό εισιτήριο που λέγεται Ticket Granting Ticket – TGT. Αυτό προορίζεται για την υπηρεσία Ticket Granting Service –TGS του KDC, και είναι κρυπτογραφημένο με το συμμετρικό κλειδί της υπηρεσίας TGS.

b. Όταν ο πελάτης θέλει να ταυτοποιηθεί σε μία υπηρεσία, στέλνει την πρώτη φορά δύο μηνύματα στην υπηρεσία TGS: i) το TGT που απέκτησε στο βήμα a και ii) ένα μήνυμα ταυτοποίησης που περιέχει χρονοσήμανση, την διεύθυνση του πελάτη και την υπηρεσία. Το μήνυμα αυτό είναι κρυπτογραφημένο με το κλειδί συνόδου (session key) πελάτη/TGS.

c. Η υπηρεσία TGS λαμβάνει από το TGT το session key πελάτη/TGS και με βάση αυτό ταυτοποιεί τον χρήστη επιβεβαιώνοντας το $2^{ο}$ μήνυμα. Παράγει ένα session key πελάτη/υπηρεσίας και ένα εισιτήριο του πελάτη στην υπηρεσία (service ticket). Επιστρέφει στον πελάτη δύο μηνύματα: i) ένα εισιτήριο πελάτη-υπηρεσίας (service ticket) (κρυπτογραφημένο με το κλειδί της υπηρεσίας) και ii) το session key πελάτη/υπηρεσίας κρυπτογραφημένο με το κλειδί πελάτη/TGS.

d. Ο πελάτης πλέον μπορεί να ταυτοποιηθεί στην υπηρεσία και να επικοινωνήσει με ασφάλεια. Η διαδικασία της ταυτοποίησης στην υπηρεσία πραγματοποιείται με δύο μηνύματα: i) το service ticket του βήματος c και ii) ένα μήνυμα ταυτοποίησης (authentication message) που περιέχει μια χρονοσήμανση και την διεύθυνση του πελάτη, κρυπτογραφημένο με το session key πελάτη/υπηρεσίας.

e. Η υπηρεσία ταυτοποιείται στον πελάτη στέλνοντας ένα μήνυμα που περιέχει την τιμή της χρονοσήμανση που έλαβε στο βήμα d αυξημένη κατά 1 βήμα. Με αυτό αποδεικνύει στον πελάτη ότι είναι η ίδια η υπηρεσία, αφού μπορεί να διαβάσει το εισιτήριο πελάτη/υπηρεσίας (κρυπτογραφημένο με το κλειδί της υπηρεσίας) και από το δεύτερο μήνυμα να διαβάσει την χρονοσήμανση.

Ακολουθώντας τα παραπάνω βήματα, ο πελάτης και η υπηρεσία έχουν ταυτοποιηθεί αμοιβαία και μοιράζονται ένα κοινό συμμετρικό κλειδί συνόδου (session key). Στην συνέχεια ο πελάτης και η υπηρεσία μπορούν να υλοποιήσουν τις υπόλοιπες λειτουργίες ασφαλείας μέσα από το ήδη κρυπτογραφημένο κανάλι έμπιστης επικοινωνίας. Η ταυτοποίηση των μηνυμάτων γίνεται ουσιαστικά με χρονοσημάνσεις (timestamps), ώστε να αποφευχθούν οι επιθέσεις επανάληψης (replay attacks). Η λειτουργία αυτή απαιτεί αυστηρό συγχρονισμό των ρολογιών μεταξύ πελάτη και υπηρεσίας, αλλιώς η ταυτοποίηση αποτυγχάνει.

Οι αρχικές υλοποιήσεις του πρωτοκόλλου Kerberos έπασχαν από την ***αδυναμία διανομής των συμμετρικών κλειδιών*** και βασίζονταν σε προεγκατάσταση των συμμετρικών κλειδιών στο KDC μέσα από διαδικασίες εκτός δικτύου (π.χ. με ανθρώπινη συνεννόηση πρόσωπο με πρόσωπο). Οι νεώτερες υλοποιήσεις υποστηρίζουν στη φάση αρχικοποίησης την χρήση του πρωτοκόλλου Public Key Cryptography for Initial Authentication in Kerberos - PKINIT [Zhu06]. Αυτό επιτρέπει την χρήση ψηφιακών πιστοποιητικών (digital certificates) X.509 για την αρχική είσοδο στο KDC (βήμα a. ανωτέρω).

Η σημαντική αδυναμία του πρωτοκόλλου Kerberos πηγάζει από την ύπαρξη του KDC μέσα στο δίκτυο. Αφενός αποτελεί σημείο αποτυχίας (single point of failure). Σε περίπτωση αστοχίας, η οποία μπορεί να οφείλεται σε αστοχία υλικού ή σε διαδικτυακές επιθέσεις άρνησης υπηρεσίας (Distributed Denial of Service - DDoS attacks), κανένας δεν μπορεί να χρησιμοποιήσει τις υπηρεσίες. Συνήθως επιλύεται με την ύπαρξη πολλαπλών KDCs που ενεργοποιούνται σε περίπτωση αστοχίας. Αφετέρου, το KDC αποτελεί πρωτεύον στόχος των κακόβουλων χρηστών σε ένα σύστημα. Σε περίπτωση που κάποιος αποκτήσει τον έλεγχο του KDC, όλο το σύστημα ταυτοποίησης καταρρέει γιατί ο εισβολέας μπορεί να αποκτήσει την ταυτότητα οποιουδήποτε χρήστη ή υπηρεσίας.

Παρόλα τα παραπάνω μειονεκτήματα, ο Kerberos αποτελεί μια πλήρη λύση ταυτοποίησης, προτυποποιημένη που βρίσκει μεγάλη διάδοση. Αποτελεί το πρωτόκολλο ταυτοποίησης πολλών αρχιτεκτονικών και συστημάτων, συμπεριλαμβανομένου και των MS Windows Domains. Επίσης υλοποιεί την ταυτοποίηση στην Αρχιτεκτονικής Ασφαλείας που προτείνουμε.

3.4.3. Ασφάλεια σε Computational Grids

Τα Computational Grids εκτείνονται μεταξύ πολλαπλών ανεξάρτητων οργανισμών και διαχειριστικών οντοτήτων. Το γεγονός αυτό θέτει επιπλέον απαιτήσεις από την αρχιτεκτονική ασφαλείας. Οι απαιτήσεις αυτές είναι:

- Πως θα γίνει η ασφαλής αντιπροσώπευση των χρηστών από ενδιάμεσες υπηρεσίες (delegation)
- Πολιτικές ασφαλείας και πως αυτές ορίζουν την εξουσιοδότηση των χρηστών στους πόρους (authorization)

Η ταυτοποίηση (authentication) των χρηστών στα Grids βασίζεται στο GSI και η εξουσιοδότηση (authorization) σε Virtual Organizations (VOs) (Εικονικούς Οργανισμούς). Στην συνέχεια θα αναλύσουμε παρακάτω το GSI αλλά την χρήση των VOs για την εξουσιοδότηση σε Grids.

3.4.3.1. Ταυτοποίηση (Authentication) στα Computational Grids

Η defacto τεχνολογία ταυτοποίησης χρηστών σε Computational Grid είναι το **Grid Security Infrastructure (GSI)** [Fost01]. Βασίζεται σε Κρυπτογραφία Δημοσίου Κλειδιού και σε Υποδομή Δημοσίου Κλειδιού (PKI) χρησιμοποιώντας πιστοποιητικά X.509 μεταξύ τελικών χρηστών μέσω του κόμβου διασύνδεσης - User Interface (UI) και των κατανεμημένων υπηρεσιών της αρχιτεκτονικής Grid, όπως περιγράψαμε στην ενότητα 2.2.3.

Το GSI εισάγει μια σημαντική επέκταση στο PKI με τα προσωρινά **Πιστοποιητικά Πληρεξουσίου** (**Proxy Certificates**) [Tuec04] στην όλη διαδικασία ασφαλής ταυτοποίησης του χρήστη. Τα proxy certificates είναι επέκταση των πιστοποιητικών X.509. Επιτρέπουν την λειτουργία αντιπροσώπευσης (delegation) ενός χρήστη από έναν ενδιάμεσο κόμβου του Grid, ο οποίος στην συνέχεια μπορεί να καλεί άλλες τελικές υπηρεσίες εκ μέρους του χρήστη. Οι υπηρεσίες αυτές λαμβάνουν τα διαπιστευτήριά του αρχικού χρήστη μέσω του πληρεξούσιου κόμβου με την μορφή proxy certificate που ξεκινά από τον αρχικό χρήστη. Μαζί με το αρχικό proxy certificate ο αρχικός χρήστης δίνει στον πληρεξούσιο κόμβο και ένα προσωρινό ζεύγος κλειδιών (public και private keys). Ο πληρεξούσιος κόμβος αναμεταδίδει στις άλλες υπηρεσίες το proxy certificate (που περιέχει προσωρινό public key του αρχικού χρήστη) και το υπογράφει με το private key που έλαβε από

τον αρχικό χρήστη. Έτσι οι τελικές υπηρεσίες επαληθεύουν πως ο πληρεξούσιος κόμβος δρα εκ μέρους του αρχικού χρήστη.

Η λειτουργία ***Delegation*** είναι βασική σε περιβάλλον Grid, γιατί επιτρέπει στις υπηρεσίες εκτέλεσης, π.χ. σε έναν Resource Broker (RB) να εκτελεί τις επιμέρους υπηρεσίες μίας ροής εργασίας (workflow) ενός χρήστη, εκ μέρους του χρήστη και χρησιμοποιώντας τα δικαιώματά του. Η λειτουργία αυτή είναι ιδιαίτερη χρήσιμη σε εργασίες των Grids που απαιτούν πολλές ημέρες για την ολοκλήρωση τους και κλήση πολλών άλλων υπηρεσιών όπως περιγράφονται στη ροή εργασίας.

Επίσης, με το proxy certificate απαλλάσσεται ο αρχικός χρήστης από να χρησιμοποιεί τα μόνιμα διαπιστευτήριά του (long-term keys, password) για πρόσβαση σε πολλαπλές κατανεμημένες υπηρεσίες: Αντ' αυτού η πρόσβαση γίνεται μέσω του πληρεξούσιου κόμβου με την χρήση των proxy certificates υλοποιώντας την λειτουργία ***Single Sign On (SSO)***.

Το μειονέκτημα των proxy certificates είναι πως αν υποκλαπούν ή αν ο πληρεξούσιος κόμβος κάνει κατάχρηση της εξουσιοδότησης δημιουργούν μεγάλο ζήτημα ασφαλείας στην αξιοπιστία του χρήστη. Για το λόγο είναι προσωρινά (έχουν μικρή διάρκεια ζωής, συνήθως 12 ώρες) και δεν χρησιμοποιούνται για κρίσιμες συναλλαγές.

Τα **Proxy Certificates** ακολουθούν την *δομή ενός πιστοποιητικού X.509*, με τις εξής διαφορές: α) το πληρεξούσιο πιστοποιητικό ***εσωκλείει εκτός από το πιστοποιητικό και το ιδιωτικό κλειδί κρυπτογράφησης***, β) έχει ***μικρή διάρκεια ζωής*** και γ) στις επεκτάσεις του πιστοποιητικού (extensions, δείτε 3.4.1) ορίζεται η πληροφορία του proxy. Αυτή περιλαμβάνει το είδος του Proxy Certificate [PrCert] (**Full Proxy, Limited Proxy, Independent Proxy, Restricted Proxy**), με τα Full Proxy να επιτρέπουν όλες τις λειτουργίες που επιτρέπονται και στο πιστοποιητικό της οντότητας, και με Limited Proxy όλες εκτός την εκτέλεση εργασιών. Το DN του proxy certificate αποτελείται από το DN του χρήστη με την προσάρτηση του μοναδικού σειριακού αριθμού που παράγει ο χρήστης (π.χ. o=ntua, ou=ece, ou=netmode, cn=moralis, cn=32432423).

Τα Computational Grid της επιστημονικής κοινότητας στην Ευρώπη έχουν ιδρύσει τον οργανισμό EU Grid Policy Management Authority - **EUGridPMA** [EUGridPMA], ο οποίος συντονίζει τις εθνικές προσπάθειες στον τομέα της

πιστοποίησης, εκδίδοντας οδηγίες για την διαδικασία πιστοποίησης και λειτουργίας τους. Παρόμοιες προσπάθειες επεκτάθηκαν σε όλο τον κόσμο με τους οργανισμούς Asian Pacific Grid PMA (APGridPMA) και The America Grid PMA (TAGPMA). Ακολουθούν ουσιαστικά την ιεραρχική μορφή οργάνωσης PKI που παρουσιάστηκε στον Πίνακα 2 (δ) της ενότητας 3.4.1.

3.4.3.2. Εξουσιοδότηση (Authorization) στα Computational Grids

Η εξουσιοδότηση των χρηστών ενός Computational Grid βασίζεται στην ομαδοποίησή τους σε **Εικονικούς Οργανισμούς (Virtual Organizations – VOs**). Σύμφωνα με το [Fost01], ένας VO είναι ***ένα σύνολο χρηστών ή/και ιδρυμάτων, που ανήκουν σε διαφορετικούς φυσικούς Οργανισμούς και έχουν κοινούς κανόνες διαμοιρασμού των πόρων***. Παραδείγματα VOs είναι οι Φυσικοί και τα όργανα/συσκευές/υπολογιστές που διαθέτουν για την εκτέλεση ενός συνεργατικού πειράματος (π.χ. το πείραμα Compact Muon Solenoid - CMS [CMS] του Large Hadron Collider – LHC του CERN), τα μέλη μια κοινοπραξίας που έχουν σκοπό να κατασκευάσουν ένα αεροπλάνο κ.α. Οι VOs είναι δυνατόν να διαφέρουν σημαντικά μεταξύ τους στο σκοπό, στην εμβέλεια, στο μέγεθος, στη δομή και στην κοινωνική τους σύνθεση.

Η εξουσιοδότηση χρηστών στους πόρους και στις υπηρεσίες στα Grids στις πρώτες υλοποιήσεις τους ακολούθησε την απλή προσέγγιση της χρήσης του **gridmap-file** σε κάθε κόμβο του Grid. Αυτός είναι ένας απλός μηχανισμός όπου σε ένα αρχείο αντιστοιχίζονται οι χρήστες του Grid (όπως εκφράζονται με το DN του μόνιμου πιστοποιητικού τους X.509) σε τοπικούς χρήστες των υπηρεσιών του κόμβου. Ο έλεγχος πρόσβασης (εξουσιοδότηση) ακολούθως γινόταν με βάση τα δικαιώματα (ρόλους) τοπικών χρηστών του συστήματος. Η λίστα αυτή υπήρχε σε κάθε κόμβο του Grid και η ανανέωσή της γινόταν με ftp, από κάποιο έμπιστο ftp site.

Ωστόσο, η λύση χρήσης του gridmap-file δεν είναι εύκολα διαχειρίσιμη σε εφαρμογές ευρείας κλίμακας, όπου απαιτείται μεγάλος αριθμός χρηστών και πόρων, διότι αυξάνει το μέγεθός του υπερβολικά και απαιτεί κεντρική διαχείριση για όλους τους VOs. Για αυτό, προτάθηκαν λύσεις όπως **το Community Authorization Service CAS** [Pear02] και **Virtual Organizations Management Membership Service (VOMS)** [Alfi03]. Οι λύσεις αυτές επιτρέπουν στον ίδιο τον VO να διαχειριστεί τους χρήστες του, τους ρόλους και τις δυνατότητές τους, αφήνοντας

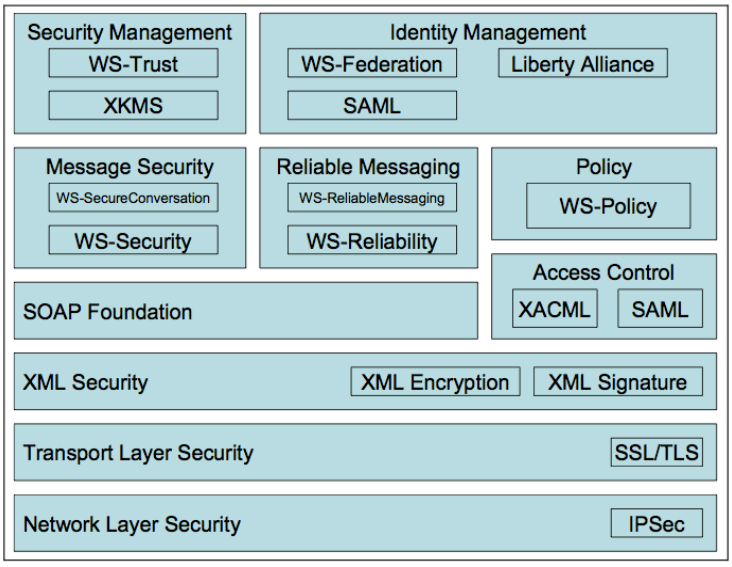
παράλληλα στους διαχειριστές των πόρων να έχουν τον απόλυτο έλεγχο πρόσβασης σε τοπικές υπηρεσίες ευθύνης τους.

Ένας χρήστης-μέτοχος ενός VO παράγει ένα GSI proxy certificate το οποίο αποστέλλει σε ένα εξυπηρετητή VOMS. Ο εξυπηρετητής, αφού αναζητήσει (σε βάση που διατηρεί για κάθε VO) την ομάδα, τους ρόλους και τις δυνατότητες του χρήστη (user capabilities) παράγει από το ληφθέν proxy, ένα νέο proxy certificate του χρήστη, στο οποίο έχει προσθέσει τις εξουσιοδοτήσεις του. Αυτό το proxy είναι το **Attribute Certificate** [Alfi05].

Παρουσιάζοντάς ο χρήστης το Attribute Certificate σε μια υπηρεσία, πραγματοποιείται η ταυτοποίηση του χρήστη και ταυτόχρονα μέσω του Attribute Certificate η υπηρεσία αναγνωρίζει σε ποιο VO ο χρήστης ανήκει και τους ρόλους που έχει. Με βάση αυτά και την τοπική πολιτική πρόσβασης, ορίζεται το επίπεδο πρόσβασης του χρήστη στην υπηρεσία.

3.5. Πρωτόκολλα Ασφαλείας των Web Services

Τα Web Services παρέχουν τους βασικούς μηχανισμούς για την δημιουργία κατανεμημένων εφαρμογών που επεκτείνονται στα σημερινά σύνθετα ετερογενή περιβάλλοντα.



Σχήμα 9: Η στοίβα πρωτοκόλλων WS- που συμπεριλαμβάνει το Security [Sing07]*

Τα πρωτόκολλα ασφαλείας Web Services είναι σχεδιασμένα να δίνουν λύση σε ένα κομμάτι του προβλήματος ασφαλής επικοινωνίας. Στο [Sing07] όπως βλέπουμε και στο Σχήμα 9 περιγράφεται η στοίβα των πρωτοκόλλων ασφαλείας των WS. Η

προτυποποίηση λαμβάνει χώρα μέσα στους μεγάλους οργανισμούς προτυποποίησης όπως ο W3C και OASIS και σημαντικές εταιρίες του χώρου της πληροφορικής όπως η IBM, Microsoft και VeriSign έχουν συμβάλλει στην προτυποποίηση τους [Neal03].

Στην συνέχεια της παρούσας ενότητας θα παρουσιάσουμε τα κύρια πρωτόκολλα ασφαλείας που αφορούν άμεσα ή έμμεσα την παρούσα διατριβή. Συγκεκριμένα θα παρουσιάσουμε το WS-Security και θα αναλύσουμε στη συνέχεια μηχανισμούς που χρησιμοποιεί, όπως το XML Encryption, XML Digital Signatures, WS-SecureConversation, αλλά και επικουρικούς μηχανισμούς και συγκεκριμένα WS-Policy και WS-Trust. Το πρωτόκολλο WS-Security χρησιμοποιείται από την αρχιτεκτονική που προτείνεται στην παρούσα διατριβή για την μεταφορά των μηνυμάτων σε μορφή XML και με το πρωτόκολλο ανταλλαγής μηνυμάτων SOAP.

3.5.1. Web Services' end-to-end Security

Παραδοσιακά, τα πρωτόκολλα που εφαρμόζονται στο Internet για να προστατέψουν τις τηλεπικοινωνίες, εφαρμόζονται πάνω από το επίπεδο της μεταφοράς (transport). Τέτοιες τεχνολογίες είναι το **SSL** (Secure Sockets Layer) και το **TLS** (Transport Layer Security) [Dier06] που προστατεύουν συγκεκριμένες εφαρμογές αντιστοιχισμένες σε συνόδους (TCP sessions) π.χ. ασφαλής σύνδεση http (https). Εναλλακτικά το πρωτόκολλο **IPSec** (Internet Protocol Security) [Kent96] προσφέρει κρυπτογράφηση ανά πακέτο IP ακολουθώντας συμμετρική κρυπτογράφηση μετά από μια φάση ανταλλαγής κλειδιών που ορίζει μία ασφαλή σύνοδο. Τα πρωτόκολλα προσφέρουν ταυτοποίηση, ακεραιότητα και εμπιστευτικότητα δεδομένων ορίζοντας μία ασφαλή σύνοδο (session) από **σημείο-σε-σημείο (point-to-point)**. Σε περιπτώσεις επεξεργασίας από ενδιάμεσους κόμβους (π.χ. Http proxies), απαιτούνται δύο ασφαλείς συνδέσεις: Μία σύνοδος μέχρι τον ενδιάμεσο κόμβο και μια δεύτερη ως τον τελικό κόμβο.

Οι τεχνολογίες ασφαλείας των **Web Services** που θα αναφερθούμε, εφαρμόζονται στο επίπεδο του μηνύματος (**Message Security**) και επιτυγχάνεται ασφάλεια από **άκρο-σε-άκρο (end-to-end)**. Η end-to-end λειτουργία βασίζεται στην μεταφορά των security tokens και των διαπιστευτηρίων (credentials) μέσα στο μήνυμα εφαρμόζοντας λειτουργίες ασφαλείας (κρυπτογράφηση και ηλεκτρονικές υπογραφές) σε όλο το μήνυμα ή μέρος του.

Σε σενάρια όπου τα μηνύματα SOAP χρειάζεται να περάσουν μέσω πολλαπλών ενδιάμεσων που εκτελούν λειτουργίες πάνω στο μήνυμα (π.χ. Firewalls και proxies που εκτελούν δρομολόγηση στο επίπεδο του μηνύματος στέλνοντας το στην αντίστοιχη εφαρμογή), τα πρωτόκολλα TLS και IPSec σε επίπεδο συνόδου (session network protocol) δεν παρέχουν την αναγκαία λειτουργικότητα των end-to-end ασφαλών επικοινωνιών. Η εφαρμογή της ασφάλειας στο επίπεδο του μηνύματος (μηνύματος SOAP) απαιτείται για να προσδώσει την αναγκαία λειτουργικότητα.

Ένας επίσης σημαντικός λόγος, ο οποίος επιβάλλει σε κάποιες σενάρια την χρήση ασφάλειας end-to-end είναι οι εφαρμογές που απαιτούν διαφορετικά επίπεδα διαβάθμισης ασφαλείας της μεταδιδόμενης πληροφορίας. Η πληροφορία σχετικά με την ασφάλεια του μηνύματος πρέπει να αποτυπώνεται καθαρά στο μήνυμα, ώστε να είναι προσπελάσιμη από τους παραλήπτες του μηνύματος για περαιτέρω επεξεργασία. Η λειτουργία αυτή δεν είναι δυνατόν να υλοποιηθεί με μηχανισμού ασφαλείας σε επίπεδο συνόδου (session) που μπορεί να περιλαμβάνει πολλά μηνύματα.

3.5.2. WS-Security

Ο μηχανισμός **WS-Security (WSS)** [Nada06a] **προσφέρει message security** και επιτρέπει ασφαλείς ανταλλαγές μηνυμάτων SOAP **end-to-end**. Προσφέρει **ένα σύνολο από επεκτάσεις στο πρωτόκολλο SOAP**, οι οποίες επιτρέπουν την υλοποίηση **ακεραιότητας, εμπιστευτικότητας και ταυτοποίησης του μηνύματος**. Τα βασικά χαρακτηριστικά συνοψίζονται παρακάτω:

- ***Υποστήριξη διαφορετικών ειδών Αντικειμένων Ασφαλείας (Security Tokens Formats), π.χ. X.509 και Kerberos. Αυτά τα security tokens φέρουν την πληροφορία που απαιτείται για να ταυτοποιηθεί ο αποστολέας του μηνύματος στην υπηρεσία***
- ***Πολλαπλές Περιοχές Εμπιστοσύνης (multiple Trust Domains)***
- ***Πολλαπλές τεχνολογίες Κρυπτογράφησης και Ηλεκτρονικές Υπογραφές (Digital Signatures) ή HMACs (Hashed Message Authentication Codes) για τμήματα του ιδίου μηνύματος SOAP***
- ***End-to-end ασφάλεια του περιεχομένου του μηνύματος***

Το πρωτόκολλο WS-Security επιτρέπει την ασφαλή ανταλλαγή μηνυμάτων SOAP από πελάτη σε εξυπηρετητή. Ωστόσο, δεν μπορεί αυτόνομα να προσφέρει την

αναγκαία λειτουργικότητα ασφαλείας σε ένα ομοσπονδιακό περιβάλλον με ετερογενείς μηχανισμούς ασφαλείας. Σε ένα τέτοιο περιβάλλον **κάθε οργανισμός έχει την δική του αρχιτεκτονική και πολιτική ασφαλείας** που βασίζεται σε συγκεκριμένες τεχνολογίες. Μέσα στην αρχιτεκτονική ασφαλείας του κάθε οργανισμού δεσπόζει το σύστημα του **Παροχέα Ταυτότητας (Identity Provider)**. Οι Identity Providers είναι οντότητες που διατηρούν στοιχεία των υπόλοιπων οντοτήτων στο σύστημα (χρήστες, υπολογιστές, εξυπηρετητές, υπηρεσίες) και εκδίδουν διαπιστευτήρια (credentials) σε χρήστες και υπηρεσίες και επιβεβαιώνουν τα εκδιδόμενα διαπιστευτήρια. Σε ένα περιβάλλον που υπάρχει μόνο μία Περιοχή Ασφαλείας (Security Domain) και ουσιαστικά ένας Identity Provider το πρωτόκολλο WS-Security μπορεί να δώσει λύση. Σε ομοσπονδιακά περιβάλλοντα το WS-Security που περιγράψαμε μπορεί να προσφέρει λύση συνεπικουρούμενο από επιπλέον υπηρεσίες (Web Services), όπως το WS-Trust για την εγκαθίδρυση trust μεταξύ των domains.

Η προδιαγραφή του WS-Security ορίζει δύο τμήματα: Το πρώτο ακολουθεί την προδιαγραφή **Core** και το δεύτερο τα **Token Profiles**.

3.5.2.1. WS-Security Core

Η προδιαγραφή Core ορίζει ένα μοντέλο ασφαλείας για να προσφέρει εμπιστευτικότητα, ακεραιότητα και ταυτοποίηση του μηνύματος SOAP, χρησιμοποιώντας Ηλεκτρονικές Υπογραφές, HMAC και Security Tokens.

Συγκεκριμένα, η εμπιστευτικότητα του μηνύματος επιτυγχάνεται με την κρυπτογράφηση του σώματος ή ακόμα και επιλεγμένων τμημάτων του μηνύματος SOAP μέσω του προτύπου XML-Encryption που θα δούμε στην 3.5.3. Η ακεραιότητα και η ταυτοποίηση του μηνύματος επιτυγχάνεται με Ηλεκτρονικές Υπογραφές ή HMAC στο σώμα ή σε στοιχεία του μηνύματος μέσω των XML Digital Signatures που παρουσιάζονται στην ενότητα 3.5.4.

Η προδιαγραφή WS-Security ορίζει επίσης πως τα διάφορα Security Tokens προστίθενται στο μήνυμα, αλλά ***δεν προδιαγράφει*** πως εκδίδονται, αποκτούνται, ανανεώνονται, εγκυροποιούνται από τους συμμετέχοντες στην ανταλλαγή μηνυμάτων. Αυτές είναι λεπτομέρειες υλοποίησης που μπορεί να διαφοροποιούνται σύμφωνα με τους διαφορετικούς μηχανισμούς ασφαλείας και τα συστήματα.

Όλη η πληροφορία ασφάλειας του μηνύματος SOAP που προορίζεται για ένα συγκεκριμένο παραλήπτη προστίθεται στην **επικεφαλίδα ασφαλείας (XML element <wss:security>)**, η οποία προστίθεται στην επικεφαλίδα του μηνύματος SOAP. Εάν το μήνυμα προορίζεται για περισσότερους από έναν παραλήπτες, μπορεί να υπάρχουν ισάριθμες επικεφαλίδες ασφαλείας. Μέσα στην επικεφαλίδα ασφαλείας μπορούν να προστεθούν διάφορα security tokens, όπως φαίνονται στον Πίνακα 3.

Πίνακας 3:Security Tokens του πρωτοκόλλου WS-Security (WSS Core)

Security Tokens	Περιγραφή
Username Tokens	**Προστίθεται προαιρετικά και προσφέρει ένα Όνομα Χρήστη (username).**
Binary Security Tokens	**Είναι αντικείμενα σε ψηφιακή μορφή (όχι XML), όπως X.509 Ηλεκτρονικά Πιστοποιητικά και Kerberos tickets.**
XML Tokens	**Χρησιμοποιείται για Αντικείμενα Ασφαλείας βασισμένα στην XML.**
EncryptedData Tokens	**Για να κρυπτογραφεί άλλα αντικείμενα ασφαλείας (όταν χρειάζεται) χρησιμοποιώντας ένα ενθυλακωμένο στο μήνυμα κλειδί ή μία αναφορά σε ένα άλλο κλειδί κρυπτογράφησης (π.χ. ιδιωτικό κλειδί του παραλήπτη).**
SecurityTokenReference Elements	**Προσφέρει μια αναφορά σε ένα αντικείμενο ασφαλείας ή κάποιον μηχανισμό που πρέπει να χρησιμοποιηθεί για να ανακτηθεί το κλειδί κρυπτογράφησης ή ηλεκτρονικής υπογραφής. Περιέχει μια Απευθείας Αναφορά που χρησιμοποιεί ένα URI Key Identifier που δείχνει προς ένα αντικείμενο ασφαλείας ή μια ενθυλακωμένη αναφορά.**
Signatures	**Επιδεικνύει την γνώση ενός κλειδιού που σχετίζεται με ένα συνοδεύον αντικείμενο ασφαλείας. Ένα Αντικείμενο Υπογραφής (Signature Token) περιέχει μια XML Υπογραφή που μπορεί να είναι ενθυλακωμένη σε μια κεφαλίδα ασφαλείας.**
XML Encryption Reference List	**Απαριθμεί σε λίστα όλα τα κρυπτογραφημένα <xenc: EncryptedData> όπου είναι όλα τα κρυπτογραφημένα XML στοιχεία μέσα στο φάκελο του SOAP.**

Η επικεφαλίδα ασφαλείας πρέπει να περιέχει τουλάχιστον ένα Security Token ή μια αναφορά σε αυτό, ώστε να μπορεί ο παραλήπτης να βρει τα διαπιστευτήρια που χρησιμοποιήθηκαν για την προστασία του μηνύματος. Μετά, ανάλογα με την λειτουργία ασφαλείας (εμπιστευτικότητα ή ακεραιότητα), μπορεί τα στοιχεία του μηνύματος SOAP να αντικατασταθούν από τα κρυπτογραφημένα στοιχεία που περιέχουν το κρυπτογραφημένο κείμενο (ciphertext) και να προστεθεί σε αυτό μια Ηλεκτρονική Υπογραφή.

3.5.2.2. WS-Security X.509 και Kerberos Token Profiles

Εκτός από την προδιαγραφή Core, υπάρχουν επιπλέον προδιαγραφές που ανήκουν στο WS-Security (WSS), οι οποίες ορίζουν με ακρίβεια πως χειρίζονται συγκεκριμένα Security Tokens όπως το **WSS X.509 Certificate Token Profile**

[Nada06b], το **WSS Kerberos Token Profile** [Nada06c] και το **WSS SAML Token Profile** [Monz06]. Ως επεκτάσεις στο προδιαγραφή Core, έχουν σκοπό, ακολουθώντας τις γενικές αρχές της, να καλύψουν τις εξειδικευμένες απαιτήσεις του κάθε μηχανισμού ασφαλείας.

Τα Kerberos και X.509 Certificate Token Profiles αποτελούν αντικείμενο μελέτης της διατριβής και παρουσιάζεται λεπτομερώς τη λειτουργία τους στις ενότητες 6.2.4 και 6.2.3 αντίστοιχα. Εν συντομία, το WSS X.509 Token Profile ορίζει πως ένα X.509 Certificate ενσωματώνεται ως Binary Security Token μέσα στην επικεφαλίδα ασφαλείας του μηνύματος SOAP και πως ενσωματώνονται οι αναφορές σε πιστοποιητικά ως SecurityTokenReferences. Ορίζει τριών ειδών αναφορές σε X.509 Certificate: Αναφορά σε Certificate Subject, αναφορά σε Binary Token (X.509 Certificate) και αναφορά σε Issuer and Serial number ενός Certificate. Με βάση την αναφορά ή το ίδιο το certificate (ενσωματωμένο ως Binary Token) η υπηρεσία μπορεί να βρει το πιστοποιητικό που χρησιμοποιήθηκε για να υπογράψει/κρυπτογραφήσει το μήνυμα, από εκεί να ανακτήσει το δημόσιο κλειδί του αποστολέα και να επαληθεύσει την υπογραφή (εφόσον υπάρχει) χρησιμοποιώντας XML Signatures και να αποκρυπτογραφήσει το κρυπτογραφημένο στοιχείο. Συνήθως όταν χρησιμοποιείται WSS X.509 Token Profile, σε κάθε μήνυμα χρησιμοποιείται το δημόσιο κλειδί της υπηρεσίας για να κρυπτογραφηθεί και να αποσταλεί ένα συμμετρικό κλειδί στο μήνυμα. Η υπηρεσία-παραλήπτης του μηνύματος αποκρυπτογραφεί το συμμετρικό κλειδί και το χρησιμοποιεί στην συνέχεια για να αποκρυπτογραφήσει τα στοιχεία που μηνύματος που αυτό έχει χρησιμοποιηθεί (μέσω της XML Encryption Reference List). Να σημειωθεί ότι κάθε μήνυμα SOAP έχει αυτόνομο μηχανισμό ασφαλείας και μπορεί να αποτελείται από πολλά πακέτα IP.

Το WSS Kerberos Token Profile ορίζει πως ένα Kerberos Service Ticket ενσωματώνεται σε ένα WSS προστατευμένο μήνυμα και πως μέσω αυτού επιτρέπει στην υπηρεσία να ταυτοποιήσει και να επαληθεύσει υπογραφές και αποκρυπτογραφήσει το μήνυμα. Ορίζει πως στο πρώτο μήνυμα στέλνεται το Service Ticket ως Binary Security Token. Με βάση αυτό ταυτοποιείται το μήνυμα στην υπηρεσία και στα επόμενα αποστέλλεται μια αναφορά σε αυτό ως SecurityTokenReference. Το πρωτόκολλο ορίζει πως οι υπογραφές στο μήνυμα πρέπει να είναι HMAC και πρέπει να ακολουθείται συμμετρική κρυπτογράφηση για να κρυπτογραφηθούν στοιχεία του μηνύματος SOAP.

3.5.3. XML Encryption

Το πρότυπο XML Encryption παρέχει τους κανόνες για την **κρυπτογράφηση και αποκρυπτογράφηση Εγγράφων XML** με την βοήθεια Security Tokens (Ηλεκτρονικά Πιστοποιητικά, Εισιτήρια Kerberos κτλ.). Η κρυπτογράφηση μπορεί να εφαρμοστεί σε όλο το έγγραφο XML ή σε μέρη αυτού. Το πρότυπο έχει προταθεί από το W3C XML Encryption Working Group [Imam02] και χρησιμοποιείται από το πρότυπο WS-Security για να παρέχει εμπιστευτικότητα μηνύματος.

Το XML Encryption (**XML-Enc**), επιτρέπει στα ανταλλασσόμενα Έγγραφα XML, να διατηρούν ασφαλείς και απροστάτευτες καταστάσεις της πληροφορία ακόμα και στο ίδιο έγγραφο. Συγκεκριμένα μπορεί να χρησιμοποιηθεί για να **κρυπτογραφεί**:

- **Όλο το έγγραφο XML**
- **Ένα στοιχείο (xml element) του εγγράφου XML**
- **Το περιεχόμενο ενός στοιχείου XML**
- **Binary Δεδομένα, π.χ. ψηφιοποιημένες εικόνες μέσα σε ένα Έγγραφο XML**

Τα **κλειδιά** για κρυπτογράφηση/αποκρυπτογράφηση μπορεί να είναι **είτε συμμετρικά είτε ασύμμετρα**, ακολουθώντας τους μηχανισμούς της Συμμετρικής και της Δημοσίου Κλειδιού Κρυπτογραφίας αντίστοιχα. Αρκετοί κρυπτογραφικοί μηχανισμοί υποστηρίζονται από το πρότυπο (π.χ. Triple-DES, AES-128, RSA-1.5, Diffie Hellman, SHA1 κτλ.). Μπορεί επίσης το χρήστης να ορίσει δικούς του αλγόριθμους ακολουθώντας το συντακτικό και τις συστάσεις του προτύπου.

Το XML Encryption δίνει την δυνατότητα κρυπτογράφησης όλου ή μέρους ενός αρχείου XML παράγοντας ένα κρυπτογραφημένο στοιχείο XML στην θέση του. Το κρυπτογραφημένο στοιχείο περιέχει το κρυπτογραφημένο κείμενο (ciphertext) και προαιρετικά α) την μέθοδο κρυπτογράφησης (αλγόριθμο) του στοιχείου, β) πληροφορία σχετικά με την ανάκτηση του κρυπτογραφικού κλειδιού και γ) των παραμέτρων κρυπτογράφησης που χρησιμοποιήθηκαν κατά την κρυπτογράφηση και είναι απαραίτητα κατά την λειτουργία της αποκρυπτογράφησης, π.χ. Ημερομηνία/Ώρα κρυπτογράφησης, encoding για την μετατροπή του σε character oriented (π.χ. Base64) κτλ.

Το πρότυπο XML-Enc χρησιμοποιείται από το WS-Security για να παρέχει κρυπτογράφηση των στοιχείων του μηνύματος SOAP, προσφέροντας λύση στο πρόβλημα της εμπιστευτικότητας του μηνύματος.

3.5.4. XML Digital Signature

Το πρότυπο **XML Digital Signature** [Bart08] **ορίζει το συντακτικό και τους κανόνες επεξεργασίας των Ψηφιακών Υπογραφών σε μορφή XML**. Μια Ψηφιακή Υπογραφή σε XML μπορεί να εφαρμοστεί σε οποιοδήποτε ψηφιακό περιεχόμενο, συμπεριλαμβανομένων εγγράφων XML όπως μηνύματα SOAP. **Οι υπογραφές σχετίζονται με ψηφιακό περιεχόμενο με την χρήση Μοναδικού Αναγνωριστικού Πόρου (Unique Resource Identifier – URI)**. Ένα έγγραφο XML μπορεί να περιέχει παραπάνω από μια υπογραφές XML, που σχετίζονται με στοιχεία του εγγράφου ή είναι υπογραφές διαφορετικών οντοτήτων πάνω στο έγγραφο ή στοιχείων αυτού.

Οι ψηφιακές υπογραφές XML, όταν χρησιμοποιούνται στις επικοινωνίες βασισμένες στο πρωτόκολλο SOAP, μπορούν να **προσφέρουν ταυτοποίηση (authentication), ακεραιότητα (integrity), μη άρνηση ενεργειών (non-repudiation) στο μήνυμα**. Αν και ο όρος Ψηφιακή Υπογραφή αναφέρεται σε Κρυπτογραφία Δημοσίου Κλειδιού, οι Ψηφιακές Υπογραφές XML μπορούν να χρησιμοποιήσουν Συμμετρική Κρυπτογραφία. Σε αυτήν την περίπτωση ονομάζονται **Hashed Message Authentication Codes (HMAC)**, όπως είδαμε και στην παράγραφο 3.3.1.

Στην γενική περίπτωση, μια Ψηφιακή Υπογραφή ενός εγγράφου είναι ένα αντικείμενο που παράγεται από εφαρμογή συνάρτησης κατακερματισμού του Εγγράφου και την κρυπτογράφηση της παραγόμενης τιμής. Οι Ψηφιακές Υπογραφές XML (XML Signatures) ακολουθούν αυτόν τον κανόνα και συμμορφώνονται με το πρότυπο XML. Επομένως, μια Υπογραφή XML ενθυλακώνεται σε έγγραφο XML ή σε SOAP μήνυμα ως ένα στοιχείο υπογραφής (Signature Element).

Μια τυπική XML Signature σύμφωνα με τις προδιαγραφές συνοδεύεται από:

- Το στοιχείο **SignedInfo**, που περιλαμβάνει αναφορά στο αλγόριθμο κανονικοποίησης (π.χ. Base64), στο αλγόριθμο υπογραφής (π.χ. RSA) και στον αριθμό των αναφορών
- Το στοιχείο **SignatureValue**, το οποίο περιέχει την τιμή της υπογραφής

- (προαιρετικά) το στοιχείο KeyInfo που επιτρέπει σε ένα παραλήπτη να αποκτήσει τα διαπιστευτήρια που απαιτούνται να επικυρώσει την υπογραφή και
- (προαιρετικά) το στοιχείο **Object**, που ορίζει την κωδικοποίηση και το τύπο MIME και χρησιμοποιείται για ενημερωτικούς σκοπούς.

Η προδιαγραφή XML Signature, από το σχεδιασμό της, **δεν επιβάλει την χρήση μια συγκεκριμένης πολιτικής εμπιστοσύνης**. Ο υπογράφων ενός εγγράφου δεν είναι υποχρεωμένος να ενσωματώσει κάποια πληροφορία κλειδιού. Μπορεί όμως να ενσωματώσει ένα στοιχείο που ορίζει το ίδιο το κλειδί, ένα όνομα κλειδιού, ένα πιστοποιητικό X.509 κτλ. Εναλλακτικά, μπορεί να ενσωματωθεί ένας σύνδεσμος προς την τοποθεσία που βρίσκεται όλη η πληροφορία για το κλειδί.

Αξίζει να τονίσουμε πως τόσο οι μηχανισμοί που υλοποιούν/ακολουθούν την προδιαγραφή XML Signature αλλά και XML-Enc επιβαρύνουν την απόδοση (performance) εξαιτίας την κανονικοποίησης (canonicalization) που εκτελούν πριν την υπογραφή/κρυπτογράφηση του μηνύματος SOAP [Zhan07]. Το canonicalization ενός αρχείου XML είναι η μετατροπή του σε μια «κανονική» μορφή, όπου έχουν αφαιρεθεί κενά, έχουν μπει σε σειρά και έχουν αφαιρεθεί τα πλεονάζοντα namespaces, έχουν μετατραπεί τα σχετικά URI σε απόλυτα κτλ. [Boye08]. Η κανονικοποίηση είναι απαραίτητη για να εφαρμόζονται οι αλγόριθμοι κατακερματισμού (hashing) και κρυπτογράφησης σε αποστολέα και παραλήπτη στο ίδιο ακριβώς κείμενο, γιατί οι XML parsers μπορεί να αλλοιώνουν τα μηνύματα (π.χ. αφαιρούν κενά). Η διαδικασία parsing προηγείται στον παραλήπτη της επεξεργασίας των στοιχείων ασφαλείας (digital signatures, Security Tokens κτλ.).

3.5.5. WS-SecureConversation

Η προδιαγραφή **WS-SecureConversation** [Nada09] εντάσσεται στις προδιαγραφές ασφαλείας των Web Services. Χρησιμοποιείται συνήθως σε συνέργεια των WS-Security, WS-Trust και WS-Policy που θα δούμε στην επόμενη ενότητα για να επιτρέψει συμμετοχή των περιεχομένων ασφαλείας.

Η προδιαγραφή WS-Security, όπως είδαμε, παρέχει τους μηχανισμούς για να προστατεύει ένα μήνυμα SOAP προς τη μία κατεύθυνση. Σε περιπτώσεις όπου απαιτούνται **πολλαπλές ανταλλαγές μηνυμάτων**, είναι πιο αποδοτικό να δημιουργηθεί ένα **είδος συνόδου μεταξύ της υπηρεσίας και του καταναλωτή της**,

ώστε να μειωθεί το βάρος χειρισμού των πληροφοριών ασφαλείας ανά μήνυμα. Η προδιαγραφή WS-SecureConversation παρέχει την ζητούμενη λειτουργικότητα μέσω επεκτάσεων στους βασικούς μηχανισμούς του WS-Security. Σκοπός του είναι να επιτρέψει τη δημιουργία συμφωνίας περιεχομένου ασφαλείας (security context) μέσω ενός security context token και τον να ορισθεί κλειδί συνόδου όταν γίνονται πολλαπλές ανταλλαγές μηνυμάτων.

Η αποτελεσματική επεξεργασία μηνυμάτων που παρέχει ο μηχανισμός εξαιτίας του γεγονότος ότι δημιουργεί συνοδό**, βελτιώνει την συνολική απόδοση της υπηρεσίας σε επόμενες ανταλλαγές μηνυμάτων**. Συνοψίζοντας, οι κύριοι στόχοι του WS-SecureConversation είναι:

- Να ορίσει πως το security context token εγκαθίσταται,
- Να περιγράψει πως τα security context tokens τροποποιούνται και τέλος
- Να ορίσει πως τα παραγόμενα κλειδιά υπολογίζονται και μεταβιβάζονται στο μήνυμα

Το security context token δεν αποτελεί από μόνο του λύση ασφαλείας, αλλά είναι δομικό στοιχείο που μπορεί να χρησιμοποιηθεί με άλλα πρωτόκολλα που βασίζονται σε Web Services (όπως το WS-Security) για να διευκολύνει στην δημιουργία ενός μεγάλου φάσματος μοντέλων ασφαλείας και τεχνολογιών εντός του πρωτοκόλλου SOAP.

3.5.6. WS-Policy και WS Trust

Το πρότυπο **WS-Policy** [Veda07a] είναι πρόταση του W3C που **ορίζει πως ένα Web Service μπορεί να δημοσιεύσει την πολιτική ασφαλείας του, ως μέρος ενός εγγράφου WSDL**. Μια πολιτική όπως ορίζεται στο πρότυπο είναι μια συλλογή από εναλλακτικές πολιτικές, οι οποίες με την σειρά τους είναι μια συλλογή από μία ή παραπάνω διαβεβαιώσεις πολιτικής. Οι εναλλακτικές πολιτικές δεν είναι ταξινομημένες, επομένως δεν υπάρχει η έννοια της προτεραιότητας/προτίμησης μεταξύ των εναλλακτικών. Το πρότυπο WS-Policy ορίζει τις δυνατότητες και τους περιορισμούς των πολιτικών, π.χ. αναγκαία security tokens, υποστηριζόμενοι αλγόριθμοι κρυπτογράφησης, κανόνες ιδιωτικότητας, πρωτόκολλα μεταφοράς, πολιτικές ιδιωτικότητας, σχήματα ταυτοποίησης, χαρακτηριστικά Ποιότητας Υπηρεσίας (QoS) σε ενδιάμεσους και τελικούς κόμβους. Επίσης έχει σχεδιαστεί να λειτουργεί μαζί με άλλα πρωτόκολλα Υπηρεσιών Ιστού, όπως τα WSDL και UDDI.

Το πρότυπο WS-Policy συμπληρώνεται με το πρότυπο Attachments (WS-PolicyAttachment [Veda07b]), όπου ορίζεται πως οι πολιτικές ενσωματώνονται στο WSDL. Ένας πελάτης που επιθυμεί να καλέσει μια υπηρεσία, μπορεί να εξετάσει την δημοσιευμένη πολιτική της μέσω WSDL, να μαζέψει τα αναγκαία διαπιστευτήρια και να την καλέσει. Όταν η υπηρεσία λάβει μια αίτηση από ένα πελάτη/χρήστη, εξετάζει εάν ικανοποιεί τις διαβεβαιώσεις πολιτικής και δρα ανάλογα.

Σε περιβάλλοντα πολλαπλών περιοχών εμπιστοσύνης ασφαλείας (security trust domains), οι συνδιαλλασσόμενοι από διαφορετικές περιοχές πρέπει να αποφασίσουν εάν μπορούν να «εμπιστευτούν» τα διαπιστευτήρια της άλλης πλευράς. Συγκεκριμένα, δυο συνδιαλλασσόμενοι για να επικοινωνήσουν με ασφάλεια, πρέπει να ανταλλάξουν διαπιστευτήρια ασφαλείας (άμεσα ή έμμεσα). Η λύση σε αυτό το πρόβλημα δίνεται με το WS-Trust.

Το πρωτόκολλο **WS-Trust** [Nada07] είναι ένα πρότυπο του οργανισμού προτυποποίησης OASIS [OASIS] από το 2007 και **περιγράφει ένα πλαίσιο για μοντέλα εμπιστοσύνης διαφορετικών περιοχών ασφαλείας που μπορούν να ενσωματωθούν σε Web Services**. Προδιαγράφει διάφορα χαρακτηριστικά ενός security token στο μήνυμα αίτησης, όπως η περίοδος ζωής του, το μέγεθος του κλειδιού, οι τύποι των κλειδιών και ο εκδότης του.

Το πρότυπο WS-Trust ορίζει την υπηρεσία **Security Token Service (STS)** και ένα απλό πρωτόκολλο αίτησης/απάντησης για την αίτηση/έκδοση/επαλήθευση security tokens, τα οποία περιγράφονται μέσω του πρωτοκόλλου WS-Policy και χρησιμοποιούνται από στο πρωτόκολλο WS-Security. Η υπηρεσία STS δρα ως Identity Provider μεταξύ των διαφορετικών security domains και ο πρωταρχικός σκοπός της είναι η έκδοση **identity security tokens**, τα οποία περιέχουν **ισχυρισμούς (claims)** για την ταυτότητα του αιτούντα. Ένας πελάτης ζητά να ανταλλάξει ένα security token με ένα token που καταλαβαίνει η υπηρεσία στο άλλο domain, στέλνοντας μια αίτηση RequestSecurityToken (RST) στην υπηρεσία STS. Η υπηρεσία STS απαντάει με ένα μήνυμα RequestSecurityTokenResponse (RSTR), που περιέχει το νέο security token.

4. Προτεινόμενη Αρχιτεκτονική

4.1. Εισαγωγή

Στο παρόν κεφάλαιο, θα παρουσιάσουμε την αρχιτεκτονική ασφαλείας που προτείνεται στη διατριβή. Σκοπός μας είναι ο σχεδιασμός μιας αρχιτεκτονικής ασφαλείας, η οποία ταυτοποιεί και εξουσιοδοτεί χρήστες ή ενδιάμεσες υπηρεσίες, αλλά παράλληλα βελτιώνει την ποιότητα της υπηρεσίας προσφέροντας μικρότερους χρόνους απόκρισης στις τελικές υπηρεσίες. Έχει σχεδιαστεί για Instrumentation Grids, τα οποία σέβονται τις αρχές των Υπηρεσιοστρεφών Αρχιτεκτονικών (SOA). Το γενικό περιβάλλον παρουσιάσθηκε στο 2^{ο} Κεφάλαιο.

Η υλοποίηση της αρχιτεκτονικής θα παρουσιασθεί εκτενώς στο 5^{ο} Κεφάλαιο και τα αποτελέσματα της στο 6^{ο} Κεφάλαιο. Το παρόν κεφάλαιο διαρθρώνεται ως εξής: Αρχικά παρουσιάζονται οι βασικές αρχές που ακολουθήσαμε και η προτεινόμενη Αρχιτεκτονική Ασφαλείας. Ακολουθεί η ανάλυση της, παρουσιάζοντας την συνολική λειτουργία της σε βήματα και την λειτουργία του κάθε υποσυστήματος (component) ξεχωριστά. Στην συνέχεια του κεφαλαίου ακολουθεί η διαστρωμάτωση των πρωτοκόλλων ασφαλείας της αρχιτεκτονική και η συμμόρφωσή της με τα πρότυπα των Συστημάτων Υπολογιστικού Πλέγματος - Computational Grid.

4.2. Απαιτήσεις της Προτεινόμενης Αρχιτεκτονικές

Στην παρούσα ενότητα θα παρουσιαστούν οι απαιτήσεις ασφαλείας που πρέπει να τηρεί η προτεινόμενη αρχιτεκτονική. Ακολουθώντας τις αρχές των Computational Grids και των αρχών ασφαλείας όπως περιγράφονται στις ενότητες 3.4.3, 3.1 και 3.2, οι **βασικές απαιτήσεις** από υποσυστήματα ασφαλείας μιας τέτοιας αρχιτεκτονικής είναι [Butl00]:

- **Single Sign on (SSO)**: Ένας χρήστης παρουσιάζει τα διαπιστευτήρια του και ταυτοποιείται μία φορά. Εφεξής μπορεί να χρησιμοποιεί πολλαπλούς πόρους, εφόσον φυσικά διαθέτει την απαραίτητη εξουσιοδότηση, χωρίς την ανάγκη να τους παρουσιάσει τα διαπιστευτήριά του. Διαπιστευτήρια μπορεί να είναι το username-

password ή η κατοχή ενός ιδιωτικού κλειδιού που αντιστοιχεί σε ένα ηλεκτρονικό πιστοποιητικό. Τα Computational Grids χρησιμοποιούν ηλεκτρονικά πιστοποιητικά και συγκεκριμένα proxy certificates.

- **Εξουσιοδότηση (Authorization) σε κατανεμημένους πόρους**: Ένας χρήστης αποκτά πρόσβαση σε ένα πόρο σύμφωνα με την τοπική πολιτική πρόσβασης που εφαρμόζεται σε κάθε πόρο.
- **Μεταβίβαση Διαπιστευτηρίων (Credential Delegation)**: Ένας χρήστης μπορεί να μεταβιβάσει τα διαπιστευτήρια τους σε ενδιάμεσες υπηρεσίες για να εκτελέσουν εργασίες εκ μέρους του. Πραγματοποιείται με την χρήση proxy certificates.
- **Ακεραιότητα της Επικοινωνίας (Integrity)**: Τα δεδομένα που ανταλλάσσονται μέσω δικτύου δεν πρέπει να μπορούν να αλλοιωθούν από κάποιον ενδιάμεσο.
- **Εμπιστευτικότητα της Επικοινωνίας (Confidentiality)**: Τα δεδομένα δεν πρέπει να αποκαλύπτονται σε μη εξουσιοδοτημένα συστήματα ή χρήστες.

Παράλληλα με τις παραπάνω απαιτήσεις ασφαλείας που διέπουν τα συνήθη Συστήματα Υπολογιστικού Πλέγματος, σκοπός της προτεινόμενης αρχιτεκτονικής ασφαλείας να βελτιώσει την **απόδοση των λειτουργιών ασφαλείας** (ταυτοποίηση – ακεραιότητα – εμπιστευτικότητα) με σκοπό να βελτιώσουν την απόδοση κατά την λειτουργία ελέγχου των πόρων σε Instrumentation Grids, τα οποία παρουσιάσαμε στην ενότητα 2.3.

Στην περίπτωση του Instrumentation Grid, η βελτίωση της απόδοσης αφορά κυρίως τον συνολικό χρόνο απόκρισης μιας ενέργειας ελέγχου σε μια εφαρμογή και κατά δεύτερο λόγο την βελτίωση της απόδοσης στην υπηρεσία του Grid που ελέγχει τα όργανα, π.χ. στο Instrument Element. Ο συνολικός χρόνο απόκρισης εξαρτάται από την καθυστέρηση του δικτύου (Round Trip Time RTT) και τον χρόνο επεξεργασίας της αίτησης στον εξυπηρετητή όπου βρίσκεται η υπηρεσία ελέγχου. Ο χρόνος εξυπηρέτησης της αίτησης είναι ένας σύνθετος χρόνος και αποτελείται από το χρόνους της αποκωδικοποίησης του μηνύματος, του χρόνου των λειτουργιών ασφαλείας (αποκρυπτογράφησης, ταυτοποίησης κ.α.) και τέλος του χρόνου της εξυπηρέτησης του αιτήματος στην υπηρεσία. Όσον αφορά τη βελτίωση της απόδοσης

στο Instrument Element (IE), αυτό αφορά την αύξηση του ρυθμού επεξεργασίας μηνυμάτων (message throughput).

Ο κύριος στόχος της διατριβής είναι να βελτιώσει τον χρόνο απόκρισης υπηρεσίας όσον αφορά τις απαραίτητες λειτουργίες ασφαλείας των διαδικτυακών πόρων που ελέγχονται μέσω αντίστοιχων υπηρεσιών (IEs), βελτιώνοντας τον συνολικό χρόνο απόκρισης της εφαρμογής και την διαδραστικότητα που αντιλαμβάνεται ο τελικός χρήστης/εφαρμογή και βελτιώνοντας την ρυθμαπόδοση της υπηρεσίας ελέγχου των πόρων.

Όπως είδαμε στο 3^{ο} Κεφάλαιο η ταυτοποίηση σε εφαρμογές υπολογιστικού πλέγματος γίνεται με την χρήση του Grid Security Infrastructure – GSI. Το GSI βασίζεται σε κρυπτογραφία Δημοσίου Κλειδιού. Σύμφωνα με το GSI, η ταυτότητα ενός χρήστη είναι ένα πιστοποιητικό X.509. Κατά την ανταλλαγή μηνυμάτων σε τέτοια συστήματα έχουμε 2 στάδια. Σε πρώτο στάδιο έχουμε την ανταλλαγή κλειδιού συμμετρικής κρυπτογράφησης, το οποίο κρυπτογραφείται με κρυπτογραφία Δημοσίου Κλειδιού και σε δεύτερο στάδιο έχουμε την ανταλλαγή μηνυμάτων χρησιμοποιώντας το κλειδί που ανταλλάχθηκε στο πρώτο στάδιο. Αυτή η λειτουργία πραγματοποιείται σε κάθε συνοδό επικοινωνίας (session).

Αυτός ο μικτός τρόπος κρυπτογραφίας μειώνει τις υπολογιστικές απαιτήσεις σημαντικά σε σχέση με την λειτουργία βασισμένη αποκλειστικά σε κρυπτογραφία Δημοσίου Κλειδιού, ωστόσο το βάρος της ασύμμετρης κρυπτογράφησης/αποκρυπτογράφησης κατά την ανταλλαγή του συμμετρικού κλειδιού, σε περιβάλλοντα όπου έχουμε μεγάλο αριθμό μηνυμάτων, προσθέτει μεγάλο υπολογιστικό φόρτο. Αυτό είναι περισσότερο εμφανές στον εξυπηρετητή ή στην Υπηρεσία που εξυπηρετεί μηνύματα πολλών πελατών. Την πολυπλοκότητα της Κρυπτογραφίας Δημοσίου Κλειδιού την είδαμε στην παράγραφο 3.3.2. Φυσικό αποτέλεσμα του επιπλέον υπολογιστικού φόρτου στην Υπηρεσία είναι η εξυπηρέτηση λιγότερων μηνυμάτων στην μονάδα του χρόνου.

Η χρήση κρυπτογραφίας Δημοσίου κλειδιού έχει ως άμεση συνέπεια την εξυπηρέτηση λιγότερων μηνυμάτων και επομένως την αύξηση του χρόνου απόκρισης υπό υψηλό υπολογιστικό φορτίο. Επίσης στην περίπτωση μιας Αρχιτεκτονικής SOA που βασίζεται σε Web Services, κάθε μήνυμα είναι αυτόνομο. Αυτό σημαίνει πως τα

δύο στάδια που περιγράψαμε επαναλαμβάνονται σε κάθε μήνυμα και επομένως η επιβάρυνση στην απόδοση είναι σημαντική.

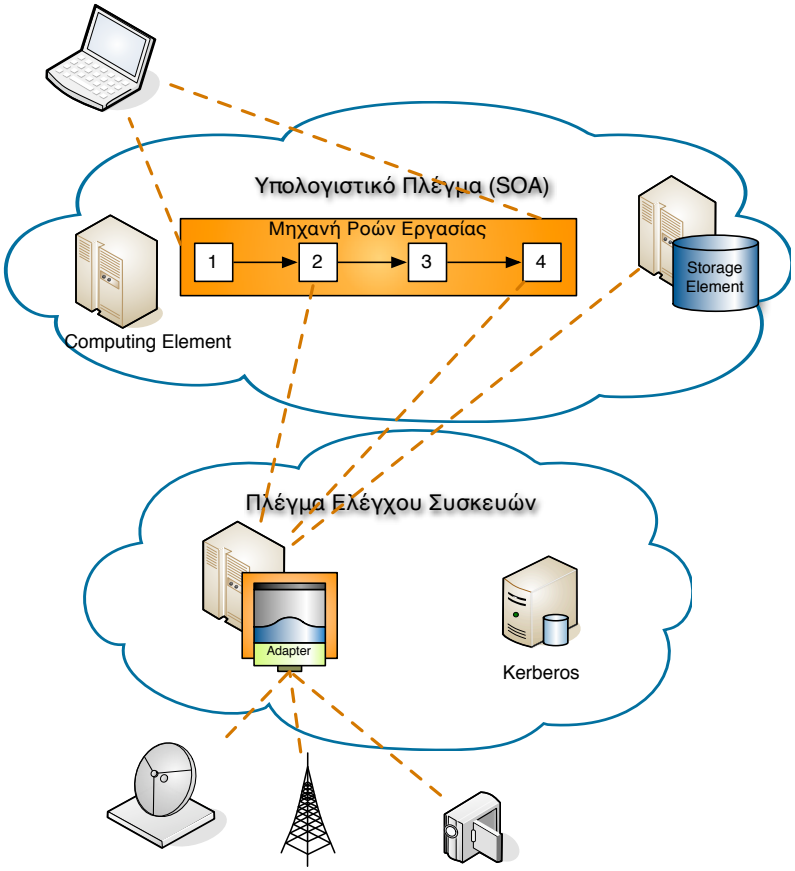
Στην Αρχιτεκτονική Ασφαλείας της διατριβής ακολουθήσαμε την αρχή της εξολοκλήρου χρήσης συμμετρικής κρυπτογραφίας κατά την επικοινωνία με τον εξυπηρετητή που αντιπροσωπεύει τα όργανα. Αυτό επιτυγχάνεται με την χρήση του Πρωτοκόλλου Kerberos [Neum05], το οποίο το παρουσιάσαμε στην ενότητα 3.4.2. Το πρωτόκολλο Kerberos περιγράφει όλους τους απαραίτητους μηχανισμούς για να πιστοποιηθεί ένας χρήστης ή μια εφαρμογή σε πόρους και υπηρεσίες ενός δικτύου. Χρησιμοποιεί αποκλειστικά συμμετρική κρυπτογραφία για να μοιράσει τα συμμετρικά κλειδιά, τα οποία θα χρησιμοποιηθούν στην συνέχεια για την ασφαλή επικοινωνία.

Ωστόσο, η χρήση του πρωτοκόλλου Kerberos σε ένα Περιβάλλον Υπολογιστικού Πλέγματος δεν μπορεί να υιοθετηθεί όπως είναι, γιατί κάθε χρήστης αναγνωρίζεται στο Grid, όπως είδαμε και στην ενότητα 3.4.3, από ένα ηλεκτρονικό πιστοποιητικό χρήστη X.509 ή από ένα Πιστοποιητικό Ενδιαμέσου (Proxy Certificate) [Tuec04]. Τα proxy certificates εκδίδονται από τους χρήστες και προωθούνται σε ενδιάμεσες υπηρεσίες/συστήματα για να τους εκπροσωπούν και να προβούν σε ενέργειες εκ μέρους τους, χρησιμοποιώντας τα δικαιώματά τους (βλέπε παράγραφο 3.4.3.1).

Το πρόβλημα συμβατότητας πιστοποιητικών – Kerberos, αντιμετωπίζεται από την αρχιτεκτονική ασφαλείας που προτείνουμε στην διατριβή, πιστοποιώντας αρχικά τον χρήστη με το ηλεκτρονικό πιστοποιητικό του και στην συνέχεια αντιστοιχίζοντάς το αυτόματα σε μία ταυτότητα του πρωτοκόλλου Kerberos. Η αντιστοίχηση πραγματοποιείται με την χρήση του πρωτοκόλλου Public Key Cryptography for Initial Authentication in Kerberos - PKINIT [Zhu06]. Στην συνέχεια η επικοινωνία του χρήστη με τις εφαρμογές γίνεται εξολοκλήρου με χρήση συμμετρικής κρυπτογράφησης. Ως χρήστης μπορεί επίσης να είναι μια οποιαδήποτε υπηρεσία που κατέχει κάποιο proxy certificate και δρα εκ μέρους του χρήστη.

Στο Σχήμα 10 δείχνουμε την εικόνα ενός Computational Grid συνδεόμενο με Instrumentation Grid (Πλέγμα Συσκευών). Παρουσιάζεται η γενική αρχιτεκτονική σε δύο επίπεδα. Το πάνω επίπεδο που περιλαμβάνει το κλασικό Υπολογιστικό Πλέγμα (Computational Grid), στο οποίο έχουμε απεικονίσει τις βασικές υπηρεσίες και το

κάτω επίπεδο που περιλαμβάνει το Πλέγμα Ελέγχου Συσκευών (Instrumentation Grid) το οποίο αλληλεπιδρά με κλασικό Υπολογιστικό Πλέγμα.



Σχήμα 10: Γενική Αρχιτεκτονική Διατριβής

4.3. Παρουσίαση Αρχιτεκτονικής Ασφαλείας

Το πρόβλημα που καλείται να λύσει η αρχιτεκτονική ασφαλείας είναι η βελτίωση στο απόδοσης και ειδικά των χρόνων απόκρισης της υπηρεσίας όσον αφορά τις λειτουργίες ασφαλείας των διαδικτυακών πόρων που ανήκουν σε Instrumentation Grid, βελτιώνοντας τον συνολικό χρόνο απόκρισης της εφαρμογής, δηλαδή την διαδραστικότητα που αντιλαμβάνεται ο τελικός χρήστης. Στο 2ο Κεφάλαιο όπως και στην προηγούμενη ενότητα αναφερθήκαμε στα Instrumentation Grids, το οποίο αποτελεί το περιβάλλον της αρχιτεκτονικής μας. Τα Instrumentation Grids χρησιμοποιούν τροποποιημένο ενδιάμεσο λογισμικό (Grid Middleware) για να προσφέρουν την διαλειτουργικότητα των κατανεμημένων πόρων που στην πλειοψηφία τους είναι όργανα μετρήσεων και ελέγχου.

Τα Computational Grids χρησιμοποιούν ως υποδομή ασφαλείας το GSI (βλέπε 3.4.3.1), το οποίο βασίζεται σε κρυπτογραφία Δημοσίου Κλειδιού. Η προτεινόμενη

αρχιτεκτονική ασφαλείας έρχεται να αντικαταστήσει το GSI κατά την επικοινωνία των τμημάτων του Instrumentation Grid με τις συσκευές. Το GSI, αν και παρέχει την απαραίτητη λειτουργικότητα, κρίναμε ακατάλληλο να χρησιμοποιηθεί στον έλεγχο συσκευών λόγω της κρυπτογραφίας Δημοσίου Κλειδιού. Συγκεκριμένα, κατά την λειτουργία ενός Web Services based Instrumentation Grid, κάθε μήνυμα SOAP αντιμετωπίζεται ως ξεχωριστό session (στο επίπεδο του WS-Security). Σε κάθε μήνυμα απαιτείται η ανταλλαγή κλειδιού συμμετρικής κρυπτογράφησης με την χρήση κρυπτογραφίας Δημοσίου Κλειδιού. Αυτό επιβαρύνει σημαντικά την απόδοση.

Ακόμα και αν προσπαθήσουμε να βελτιώσουμε την απόδοση εγκαθιστώντας session, χρησιμοποιώντας μηχανισμούς πάνω από το WS-Security όπως το WS-SecureConversation (δες παράγραφο 3.5.5), το πρόβλημα δεν εξαλείφεται πλήρως. Αντιθέτως προστίθεται ένα επιπλέον πρωτόκολλο στην επεξεργασία του μηνύματος αυξάνοντας την υπολογιστική πολυπλοκότητα και το σημαντικότερο παραμένει το πρόβλημα της αρχικοποίησης της συνόδου (session) με την χρήση κρυπτογραφίας Δημοσίου Κλειδιού τουλάχιστον κατά το πρώτο μήνυμα στο session. Το πρόβλημα της αρχικοποίησης της συνόδου μπορεί να γίνει έντονο σε περιβάλλοντα Instrumentation Grid, όταν υπάρχει μια υπηρεσία η οποία διαχειρίζεται πολλά όργανα και οι αιτήσεις. Σε αυτήν την περίπτωση είναι πιθανή η δημιουργία για συνόδους ελέγχου με την μορφή χιονοστιβάδας, δηλαδή να έχουμε πολλές αιτήσεις σε μικρό χρονικό διάστημα. Στην κατηγορία αυτή ανήκουν πολύπλοκα όργανα με τεράστιο αριθμό αισθητήρων ($O(10^4)$) όπως το CMS του Large Hadron Collider του CERN [CMS08].

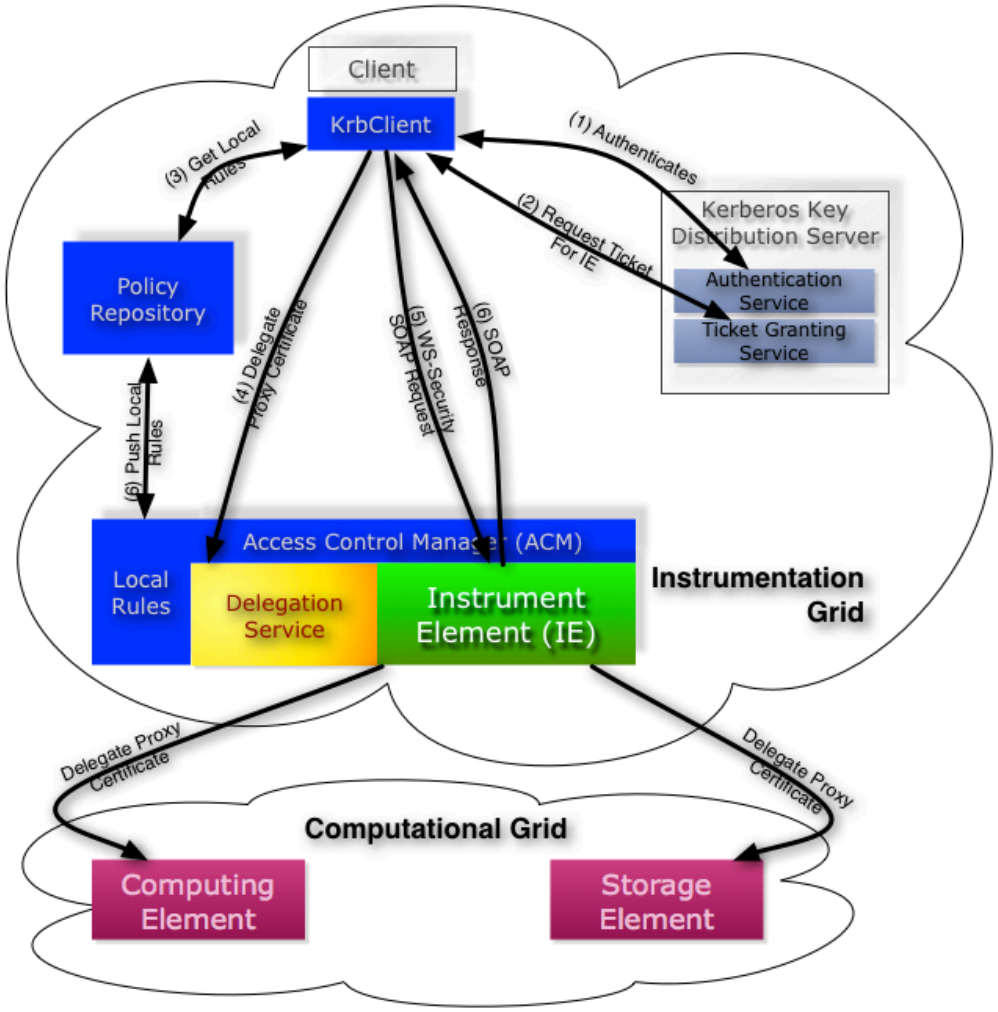
Η αρχιτεκτονική που προτείνουμε αντιμετωπίζει το πρόβλημα απόδοσης που θα προέκυπτε με κρυπτογραφία Δημοσίου Κλειδιού με την χρήση αποκλειστικά συμμετρικής κρυπτογραφίας κατά την αποστολή ανεξάρτητων μηνυμάτων SOAP, προστατευμένων με WS Security. Η διανομή των συμμετρικών κλειδιών γίνεται με την χρήση του πρωτοκόλλου Kerberos (3.4.2) και για αυτό χρησιμοποιείται το WS-Security Kerberos Token Profile (βλέπε 3.5.2.2) για να προστατέψουμε τα μηνύματα SOAP. Να υπενθυμίσουμε πως το WS Security δεν απαιτεί την διατήρηση καταστάσεων ασφαλείας ανά σύνοδο (stateless protocol). Ειδικότερα όμως το Kerberos Token Profile διατηρεί την κατάσταση ασφαλείας ανά σύνοδο με την χρήση του session key και επομένως εισάγει την κατάσταση (state) στην αρχιτεκτονική ασφαλείας, ενώ τα Web Services που προστατεύονται μπορεί να είναι stateless. Η

ιδιότητα αυτή καθιστά την επιλογή του πρωτοκόλλου Kerberos καθοριστική για την απόδοση της αρχιτεκτονικής ασφαλείας σε ένα Instrumentation Grid με απαιτήσεις διαλειτουργικότητας με αρχιτεκτονικές Grid βασισμένες σε Web Services.

Παράλληλα η πρότασή μας προσφέρει end-to-end security (ενότητα 3.5.1) όπως και το GSI θα μπορούσε με την χρήση του WS-Security X.509 Token Profile. Με τον τρόπο αυτό αποφεύγονται υλοποιήσεις τύπου point-to-point security όπως το SSL (π.χ. https), όπου μια ασφαλής σύνοδος γίνεται μεταξύ client και proxy, αντί client και server.

Η προτεινόμενη Αρχιτεκτονική Ασφαλείας είναι πλήρης προσφέροντας ταυτοποίηση αλλά και έλεγχο πρόσβασης σε επίπεδο μηνύματος. Σχεδιάστηκε ώστε να μπορεί εύκολα να εγκατασταθεί σε περιβάλλον Grid, όπου ο χρήστης του συστήματος είτε είναι φυσικό πρόσωπο είτε είναι μηχανή που δρα εκ μέρους κάποιου φυσικού προσώπου, μπορεί να πιστοποιηθεί μία φορά (SSO) και να λάβει διαπιστευτήρια, τα οποία στην συνέχεια μπορεί να τα χρησιμοποιήσει ώστε να επικοινωνήσει με υπηρεσίες που ελέγχουν πόρους, όπως συσκευές και όργανα μετρήσεων και ελέγχου. Παράλληλα διατηρεί την απαραίτητη διαλειτουργικότητα με το GSI, ώστε να το Instrumentation Grid να συνδέεται με υπάρχουσες υποδομές Computing Grid.

Οι υπηρεσίες «ελέγχου» προστατεύονται από ειδικό υποσύστημα της αρχιτεκτονικής που ονομάζεται **Access Control Manager (ACM)**. Το ACM αφενός προστατεύει την υπηρεσία πιστοποιώντας τις αιτήσεις-μηνύματα (Authorization) και αφετέρου εκτελεί τον έλεγχο πρόσβασης (Authorization), αντιπαραβάλλοντας την ταυτότητα προέλευση του κάθε μηνύματος με **Κανόνες Πρόσβασης (Access Rules)**. Οι κανόνες αυτοί περιγράφουν ποιος μπορεί να έχει πρόσβαση ανά πόρο-συσκευή που προστατεύεται από το **ACM**. και θα αναλυθούν με μεγαλύτερη λεπτομέρεια στην ενότητα 4.4.3.



Σχήμα 11: Προτεινόμενη Αρχιτεκτονική Ασφαλείας

Στο Σχήμα 11 παρουσιάζεται η προτεινόμενη αρχιτεκτονική ασφαλείας. Αποτελείται από τα εξής υποσυστήματα – τμήματα λογισμικού (components):

- **Kerberos Key Distribution Server (KDC)**: παρέχει την ταυτοποίηση σύμφωνα με το πρωτόκολλο Kerberos.
- **KrbClient**: παρέχει μια προγραμματιστική διαπροσωπία (Application Programmable Interface - API), η οποία δίνει την δυνατότητα στον προγραμματιστή που δημιουργεί το λογισμικό που υλοποιεί την πλευρά του πελάτη της αρχιτεκτονικής ασφαλείας. Κρύβει από τον προγραμματιστή την πολυπλοκότητα που έχουν οι κρυπτογραφικές λειτουργίες, οι λειτουργίες σύνθεσης του ασφαλούς μηνύματος SOAP και η διαχείριση των διαπιστευτηρίων του χρήστη (user credentials).

- **Access Control Manager (ACM)**: προστατεύει την Υπηρεσία πιστοποιώντας τα εισερχόμενα μηνύματα και ελέγχοντας την πρόσβαση σε επίπεδο λειτουργιών του πόρου.
- **Policy Repository**: ενεργεί ως συνολικό αποθετήριο κανόνων για την αρχιτεκτονική ασφαλείας. Ως αποθετήριο περιέχει όλους του Τοπικούς Κανόνες πρόσβασης για κάθε υπηρεσία, τους οποίους ανακτά σε περιοδικά διαστήματα από το ACM της κάθε υπηρεσίας. Εναλλακτικά, θα μπορούσε να λειτουργήσει προωθώντας του κανόνες στα αντίστοιχα ACM, όταν υπάρχουν αλλαγές.

Θεωρούμε το γενικό περιβάλλον όπως περιγράφτηκε στο 2^{ο} Κεφάλαιο και αναφέρθηκε στην προηγούμενη ενότητα. Το περιβάλλον αυτό είναι ένα **Instrumentation Grid**, το οποίο περιέχει στοιχεία Ελέγχου Συσκευών και οργάνων. Ένα στοιχείο ελέγχου είναι το **Instrument Element (IE)**, το οποίο ακολουθώντας λογική SOA, αντιπροσωπεύει μία ή περισσότερες συσκευές και προσφέρει τις λειτουργίες ελέγχου ως Υπηρεσία. Θεωρούμε επίσης πως η υλοποίηση των Υπηρεσιών γίνεται με την χρήση Web Services. Το Instrumentation τοποθετείται δεν υπάρχει αυτόνομο, αλλά τοποθετείται μέσα σε ένα γενικό Σύστημα Υπολογιστικού Πλέγματος (**Computational Grid**), όπου υπάρχουν υποσυστήματα όπως **Computing Elements (CE)** και **Storage Elements (SE)**, τα οποία όμως θεωρούνται εξωτερικά συστήματα από την πλευρά της προτεινόμενης αρχιτεκτονικής ασφαλείας, ωστόσο επικοινωνούν με τα IEs.

Η αναγκαία διαλειτουργικότητα με τα υπάρχοντα Υπολογιστικά Πλέγματα επιτυγχάνεται, όπως αναφέραμε και στην παράγραφο 4.2, με την χρήση του πρωτοκόλλου PKINIT. Τα πιστοποιητικά X.509 χρησιμοποιούνται για την αρχική ταυτοποίηση στην περιοχή ασφαλείας που ορίζεται από το **Kerberos KDC** και στην συνέχεια καλούνται οι υπηρεσίες που βρίσκονται προστατευμένες από το ACM. Ο έλεγχος πρόσβασης (Access Control) επιτυγχάνεται με την χρήση απλών κανόνων πρόσβασης στο επίπεδο του ACM, που βρίσκεται στην πλευρά του παρόχου της Υπηρεσίας (IE στην περίπτωσή μας), και επιτρέπει έλεγχο σε επίπεδο λεπτομέρειας του αντίστοιχου πόρου, που ο κανόνας περιγράφει.

Η βελτίωση της απόδοσης σε σχέση με τα Grids βασισμένα στο GSI, επιτυγχάνεται μέσω της αποκλειστικής χρήσης, κατά την ασφαλή ανταλλαγή

μηνυμάτων, συμμετρικής κρυπτογραφίας. Όλες οι κρυπτογραφικές λειτουργίες που εφαρμόζονται σε κάθε μήνυμα, είναι συμμετρικής φύσης, λόγο του πρωτοκόλλου Kerberos, το οποίο επιτρέπει να μοιρασθούν συμμετρικά κλειδιά στους συμμετέχοντες στην επικοινωνία (πελάτης – υπηρεσία).

Αναλύοντας με μεγαλύτερη λεπτομέρεια πως λειτουργεί το πρωτόκολλο Kerberos μέσα στην αρχιτεκτονική, βλέπουμε δημιουργείται μια σύνοδος στο επίπεδο ασφαλείας της αρχιτεκτονικής. Αυτό γίνεται μέσω του εισιτηρίου, όπως είδαμε στην ενότητα 3.4.2, το οποίο είναι ένα ψηφιακό αντικείμενο, κρυπτογραφημένο με το ιδιωτικό κλειδί της Υπηρεσίας. Προορίζεται για την συγκεκριμένη Υπηρεσία που ζητείται και περιέχει τα στοιχεία του πελάτη, ένα συμμετρικό κλειδί και τη διάρκεια ζωής του κλειδιού. Το κλειδί αυτό ονομάζεται **Κλειδί Συνόδου** (**session key**). Με αυτό το κλειδί γίνονται όλες οι κρυπτογραφικές λειτουργίες, οι οποίες είναι απαραίτητες για την ταυτοποίηση και προστασία των περιεχομένων ενός μηνύματος. Χρησιμοποιείται τόσο από τον πελάτη όσο και την Υπηρεσία για την περαιτέρω επικοινωνία. Ορίζει μια **Σύνοδο** μεταξύ πελάτη και υπηρεσίας.

Σημαντική παρατήρηση είναι πως **η ταυτοποίηση και ο έλεγχος πρόσβασης ενεργοποιείται στο επίπεδο μηνύματος**, ενώ παράλληλα διατηρείται κατάσταση ασφάλειας τόσο στον πελάτη όσο και στην υπηρεσία μέσω του κλειδιού συνόδου. Επομένως οι πελάτες έχοντας μια φορά εκδώσει και χρησιμοποιήσει το εισιτήριο, μπορούν για την διάρκεια ζωής του εισιτηρίου να επικοινωνούν ασφαλώς με την υπηρεσία.

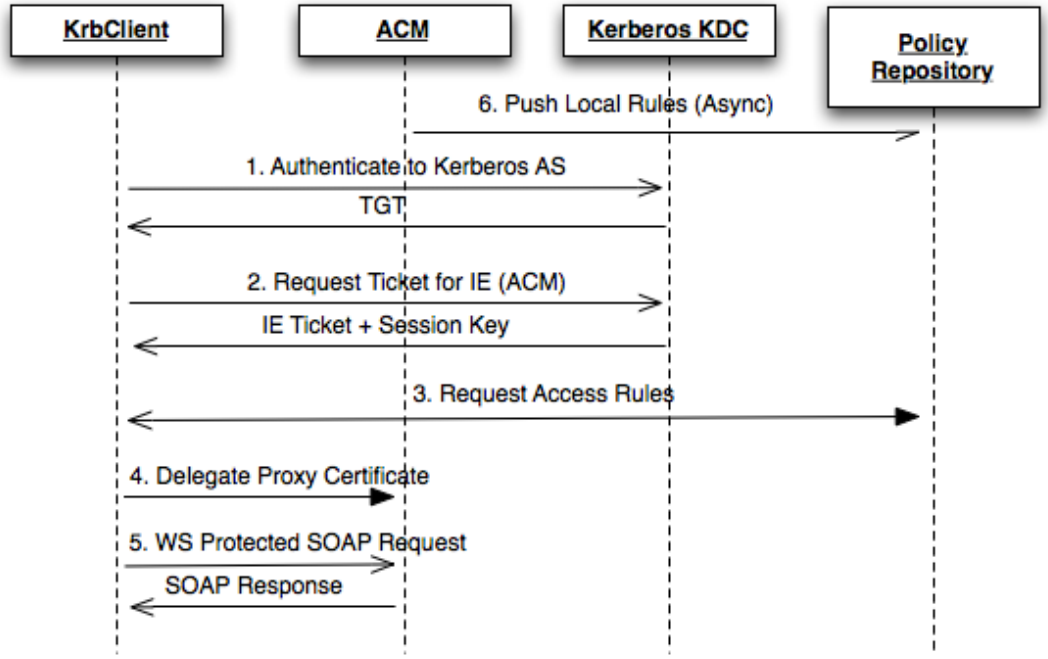
4.4. Ανάλυση Αρχιτεκτονικής

Στην ανάλυση της αρχιτεκτονικής θα δούμε τον κύκλο αποστολής ενός μηνύματος σε μια συσκευή που προστατεύεται από την αρχιτεκτονική ασφαλείας. Επίσης θα αναλύσουμε κάθε τμήμα της αρχιτεκτονικής με λεπτομέρεια και θα δούμε τις βασικές λειτουργίες του ελέγχου πρόσβασης, της αντιστοίχισης των διαπιστευτηρίων (credential mappings) και τον μηχανισμό αντιπροσώπευσης χρήστη (delegation).

4.4.1. Η λειτουργία σε βήματα

Η περιγραφή της λειτουργίας της αρχιτεκτονικής ασφαλείας αποτελείται από τα βήματα-ενέργειες που θα περιγραφούν στην συνέχεια. Στο Σχήμα 11 στα βήματα 3, 4 και 6 είναι μη υποχρεωτικά, και χρησιμοποιούνται για να προσφέρουν στον πελάτη

τις λειτουργίες προώθησης διαπιστευτηρίων για εκπροσώπηση του από την υπηρεσία και ενημέρωση του για τους κανόνες πρόσβασης στους πόρους που αντιπροσωπεύει η υπηρεσία. Στόχος των συγκεκριμένων ενεργειών είναι εύρυθμη λειτουργία του Συστήματος Υπολογιστικού Πλέγματος ελέγχου συσκευών/οργάνων.



Σχήμα 12: Αλληλουχία Μηνυμάτων (Message Sequence Flow)

Θεωρούμε το σενάριο που το λογισμικό που πελάτη θέλει να καλέσει μια υπηρεσία προστατευμένη από την αρχιτεκτονική ασφαλείας που προτείνεται στην διατριβή. Στο Σχήμα 12 παρουσιάζονται η αλληλουχία μηνυμάτων και διακρίνουμε τα εξής βήματα:

1. **Authenticate to Kerberos Authentication Server:** Το τμήμα της αρχιτεκτονικής ασφαλείας που υλοποιεί τις λειτουργίες του πελάτη (KrbClient), αρχικοποιείται από το λογισμικό του πελάτη με το πιστοποιητικό X.509 ή κατάλληλο πιστοποιητικό ενδιαμέσου (proxy certificate), το οποίο ανήκει σε έναν χρήστη. Στην συνέχεια, ο KrbClient ταυτοποιεί τον χρήστη στην Υπηρεσία Ταυτοποίησης (AS) του Kerberos. Με την επιτυχή ταυτοποίηση του λαμβάνει ο KrbClient ένα ειδικό εισιτήριο που ονομάζεται **Ticket Granting Ticket (TGT)**. Χρησιμοποιώντας αυτό το εισιτήριο και για την διάρκεια ζωής του, στις επόμενες κλήσεις προς KDC δεν χρειάζεται να καλέσει την υπηρεσία ταυτοποίησης. Η διάρκεια ζωής

του TGT προτείνεται στην αρχιτεκτονική μας να είναι ίση ή μικρότερη της διάρκειας των αρχικών διαπιστευτηρίων, εφόσον αυτά έχουν διάρκεια ζωής, όπως συμβαίνει με τα proxy certificates, ή μιας τιμή που θα προκύπτει από τα σενάρια χρήσης. Σε κάθε περίπτωση μετά την λήξη του, απαιτείται ανανέωση. Το TGT χρησιμοποιείται από το KrbClient για να ζητήσει εισιτήρια υπηρεσιών μέσω της υπηρεσίας **Ticket Granting Service (TGS)** του KDC. . Η λειτουργία του TGS προσφέρει στην προτεινόμενη αρχιτεκτονική την λειτουργία του Single Sign On - SSO.

2. **Request Tickets for the Instrument Element (IE):** Όταν το λογισμικό του πελάτη θέλει να καλέσει μια υπηρεσία προστατευμένη από το ACM, χρησιμοποιεί το KrbClient API. Μέσω αυτού δημιουργείται ένα αντικείμενο, το οποίο την πρώτη φορά, ζητά από το TGS ένα εισιτήριο για την ζητούμενη υπηρεσία, παρουσιάζοντας το TGT.

3. **Request Access Rules:** Αυτό το βήμα είναι προαιρετικό. Πριν καλέσει μια υπηρεσία, ο KrbClient μπορεί να ζητήσει από το Αποθετήριο Κανόνων (Policy Repository) εάν υπάρχει κανόνας πρόσβασης που να επιτρέπει την κλήση. Με αυτόν τον τρόπο η άρνηση πρόσβασης είναι γνωστή πριν καλέσει την υπηρεσία και δεν επιβαρύνεται το σύστημα με άσκοπες κλείσεις υπηρεσιών. Όπως αναφέραμε και πιο πάνω, το Policy Repository έχει συγκεντρωμένη όλη την πολιτική για το σύστημα που περιγράφει η αρχιτεκτονική ασφαλείας.

4. **Delegate Proxy Certificate:** Αν χρειάζεται, ο KrbClient μπορεί αυτόματα να προωθήσει (delegate) το αρχικό πιστοποιητικό, εκδίδοντας ένα νέο πιστοποιητικό ενδιαμέσου (proxy certificate) προς την Υπηρεσία **Delegation Service**. Αυτή η υπηρεσία υπάρχει στο container που βρίσκεται το IE και επιτρέπει την επικοινωνία της υπηρεσίας εκ μέρους του χρήστη με άλλους πόρους του Υπολογιστικού Πλέγματος όπως Storage Elements και Computing Elements.

5. **WS Security Protected SOAP Request:** Έχοντας ανακτήσει το εισιτήριο της υπηρεσίας στο βήμα 2, ο KrbClient επικοινωνεί με την υπηρεσία με την αποστολή ενός μηνύματος SOAP, το οποίο είναι προστατευόμενο με το πρωτόκολλο Web Service Security. Συγκεκριμένα, το SOAP εμπεριέχει

στην επικεφαλίδα του το εισιτήριο της υπηρεσίας. Ένα εισιτήριο υπηρεσίας περιλαμβάνει το αναγνωριστικό του πελάτη, την διεύθυνση IP του, το διάστημα που είναι έγκυρο, και το κλειδί του συνόδου. Όλα αυτά είναι κρυπτογραφημένα με το κλειδί της υπηρεσίας. Επομένως μόνο η εν λόγο υπηρεσία μπορεί να το διαβάσει. Η λειτουργία αυτή υλοποιεί την διαδικασία της ταυτοποίησης του μηνύματος, για το πρώτο μήνυμα, και μέσω του κλειδιού συνόδου για τα επόμενα. Η ασφαλής επικοινωνία διασφαλίζεται με την χρήση συμμετρικής κρυπτογραφίας με κλειδί αυτό της συνόδου. Ο πελάτης έχει την επιλογή να κρυπτογραφήσει ή/και να υπογράψει ψηφιακά το μήνυμα SOAP. Η κρυπτογράφηση του μηνύματος προσφέρει εμπιστευτικότητα μηνύματος (message confidentiality), ενώ η ψηφιακή υπογραφή ή το HMAC προσφέρει μόνο ακεραιότητα μηνύματος (message integrity). Επίσης μπορούν να χρησιμοποιηθούν χρονοσημάνσεις (Timestamps) για να προστατέψουν το σύστημα από επιθέσεις επανάληψης (replay attacks). Στην συνέχεια ελέγχεται στο μήνυμα σε σχέση με τους κανόνες πρόσβασης και αν αυτοί επιτρέπουν πρόσβαση του χρήστη προς τον πόρο που ζήτησε και την λειτουργία πάνω σε αυτόν.

6. **Push Local Rules (Async):** Το βήμα 6 ενεργοποιείται ασύγχρονα μόνο όταν συμβεί αλλαγή στους κανόνες πρόσβασης της υπηρεσίας (στο Σχήμα 12 η αλλαγή κανόνων συνέβη πριν την κύρια ανταλλαγή μηνυμάτων, αλλά θα μπορούσε να συμβεί σε οποιαδήποτε χρονική στιγμή). Οι κανόνες πρόσβασης προωθούνται στο αποθετήριο κανόνων Policy Repository. Αυτός είναι ο επιλεγμένος τρόπος λειτουργίας του αποθετηρίου. Εναλλακτικά, ανάλογα με την εφαρμογή, θα μπορούσε το Policy Repository να ανακτά τους κανόνες περιοδικά από τις διάφορες υπηρεσίες.

4.4.2. Παρουσίαση των τμημάτων της αρχιτεκτονικής

Στην παρούσα ενότητα θα αναφερθούμε στα διάφορα τμήματα της αρχιτεκτονικής που παρουσιάστηκαν στην ενότητα 4.3 με μεγαλύτερη λεπτομέρεια, αναλύοντας την λειτουργία τους και παρουσιάζοντας τα πρωτόκολλα και πρότυπα, στα οποία βασίστηκαν. Η υλοποίηση της αρχιτεκτονικής παρουσιάζεται στο 5^{ο} Κεφάλαιο.

4.4.2.1. KrbClient

Ο KrbClient, υλοποιείται ως Java API, εκτελεί όλες τις λειτουργίες ασφαλείας και διαχειρίζεται όλα τα διαπιστευτήρια του χρήστη που σχετίζονται με το λογισμικό πελάτη. Έχουμε ουσιαστικά τρεις συμμετέχοντες: 1) Το χρήστη (άνθρωπος ή μηχανή που δρα εκ μέρους ανθρώπου, π.χ. Workflow engine), 2) τον πελάτη που είναι το λογισμικό που υλοποιεί όλες τις λειτουργίες πελάτη στο σύστημα και 3) ο KrbClient που είναι το τμήμα (component) της αρχιτεκτονικής που υλοποιεί όλες τις λειτουργίες ασφάλειας εκ μέρους του πελάτη και κατ' επέκταση του χρήστη. Ακολουθεί η περιγραφή λειτουργίας του KrbClient:

- Αρχικά τα διαπιστευτήρια του χρήστη (X.509 certificate ή proxy certificate [Tuec04] ή VOMS[1] proxy [ALFI03]) μεταβιβάζονται από το λογισμικό του πελάτη στο KrbClient. Το διαπιστευτήρια αυτά ταυτοποιούν τον χρήστη στο Grid.

- Χρησιμοποιώντας το δοθέν πιστοποιητικό, ο KrbClient ταυτοποιεί τον χρήστη στο Kerberos Key Distribution Center – KDC και λαμβάνει τα διαπιστευτήρια του Κέρβερου που είναι το TGT, όπως είδαμε και παραπάνω. Συγκεκριμένα επικοινωνεί με την υπηρεσία Authentication Server – AS του KDC. Η διαδικασία ταυτοποίησης στο KDC μέσω πιστοποιητικού επιτυγχάνεται μέσω του πρωτοκόλλου PKINIT. Παρόλα αυτά η αρχιτεκτονική δεν εξαιρεί την χρήση και άλλων μεθόδων όπως την χρήση username-password. Ο KrbClient πλέον διατηρεί και χειρίζεται όλα τα διαπιστευτήρια του χρήστη, τα οποία είναι Ηλεκτρονικά Πιστοποιητικά X.509, το Kerberos TGT και Service Tickets του χρήστη για συγκεκριμένες υπηρεσίες.

- Στην συνέχεια όταν κάποιος ο χρήστης μέσω του λογισμικού του πελάτη θελήσει να επικοινωνήσει με μία υπηρεσία, ο KrbClient, επικοινωνεί πριν την πρώτη κλήση με TGS του KDC και λαμβάνει εισιτήριο (Service Ticket + session key, Σχήμα 12) για την συγκεκριμένη υπηρεσία. Στην συνέχεια εφαρμόζει όλες τις κρυπτογραφικές λειτουργίες στο περιεχόμενο του μηνύματος SOAP, χρησιμοποιώντας το κλειδί συνόδου που πήρε ως απάντηση μαζί με το εισιτήριο από το TGT και προσθέτει στην

[1] VOMS = Virtual Organization Management Service. Διαχειρίζεται την εγγραφή ενός χρήστη σε ένα Εικονικό Οργανισμό (Virtual Organization - VO)

επικεφαλίδα το εισιτήριο της υπηρεσίας. Η επιλογή των κρυπτογραφικών λειτουργιών είναι του πελάτη. Τέλος αποστέλλει το μήνυμα.

- Σε επόμενες κλήσεις προς την συγκεκριμένη υπηρεσία που ζητά το λογισμικό του πελάτη, ο KrbClient χρησιμοποιεί το κλειδί συνόδου (Session Key), για την διάρκεια ζωής του εισιτηρίου.Τα μηνύματα που αποστέλλονται ακολουθούν το πρωτόκολλο Web Services Security, Kerberos Token Profile 1.1. Το συγκεκριμένο profile ορίζει πως χειριζόμαστε τα Kerberos tickets μέσα σε ένα μήνυμα SOAP. Περισσότερα για τα WSS Profiles αναφέρθηκαν στην παράγραφο 3.5.2 της διατριβής.

4.4.2.2. Kerberos Key Distribution Center – KDC

Το KDC είναι η Τρίτη Έμπιστη Αρχή (Third Trusted Party - TTP) της αρχιτεκτονικής ασφαλείας που παρουσιάζουμε στην διατριβή. Σκοπός της είναι να ταυτοποιεί αμοιβαία χρήστες και υπηρεσίες και να μοιράζει μέσω του μηχανισμού των εισιτηρίων τα κρυπτογραφικά κλειδιά ώστε να επιτρέψει την ασφαλή επικοινωνία μεταξύ τους. Βασική λειτουργία αποτελεί η υποστήριξη του πρωτοκόλλου PKINIT, η οποία επιτρέπει την χρήση ηλεκτρονικών πιστοποιητικών X.509 και προσωρινών Proxy Certificates.

4.4.2.3. Access Control Management - ACM

Ο ACM είναι το βασικό τμήμα της αρχιτεκτονικής που προτείνει η διατριβή και χειρίζεται την ασφάλεια της τελικής υπηρεσίας, π.χ. το Instrument Element που είδαμε στην ενότητα 2.3. Η υπηρεσία μαζί με τον ACM ακολουθούν το προγραμματιστικό μόρφημα (programming pattern) της Chain of Responsibility και ο ίδιος ο ACM ακολουθεί το State pattern [Vlis99]. Ο ACM μπορεί να εγκατασταθεί μπροστά από οποιαδήποτε υπηρεσία και να επεξεργαστεί το μήνυμα SOAP πριν αυτό φτάσει στην τελική υπηρεσία.

Ένα εισερχόμενο μήνυμα στο ACM μπορεί να έχει ένα ή περισσότερα από τα παρακάτω χαρακτηριστικά σύμφωνα με το πρότυπο Kerberos Token Profile του Web Service Security:

i. Να περιέχει ένα εισιτήριο Kerberos

ii. Να είναι κρυπτογραφημένο

iii. Να είναι υπογεγραμμένο ηλεκτρονικά (με πρόσθεση HMAC – δες παράγραφο 3.3.1)

iv. Να περιέχει χρονοσήμανση (timestamp)

Με την παραλαβή του εισερχομένου μηνύματος, ο ACM, εκτελεί τους εξής ελέγχους:

i. Αν το μήνυμα είναι υπογεγραμμένο, επαληθεύει την υπογραφή. Οι υπογραφές στην περίπτωση συμμετρικής κρυπτογραφίας υλοποιούνται με την χρήση Message Authentication Codes όπως περιγράψαμε στην παράγραφο 3.3.1.

ii. Αν το μήνυμα είναι κρυπτογραφημένο, το αποκρυπτογραφεί

iii. Πιστοποιεί την προέλευση του μηνύματος. Αυτό γίνεται αποκρυπτογραφώντας το εισιτήριο που ενσωματώνεται στο πρώτο μήνυμα, εξάγοντας και το session key. Στα επόμενα μηνύματα χρησιμοποιεί αυτό το session key

iv. Διαβάζει την χρονοσήμανση (timestamp) και ελέγχει αν το μήνυμα βρίσκεται εντός ενός έγκυρου χρονικού διαστήματος

v. Τέλος ελέγχει αν ο χρήστης έχει πρόσβαση σύμφωνα με την τοπική πολιτική ασφαλείας σε συγκεκριμένη λειτουργία του πόρου

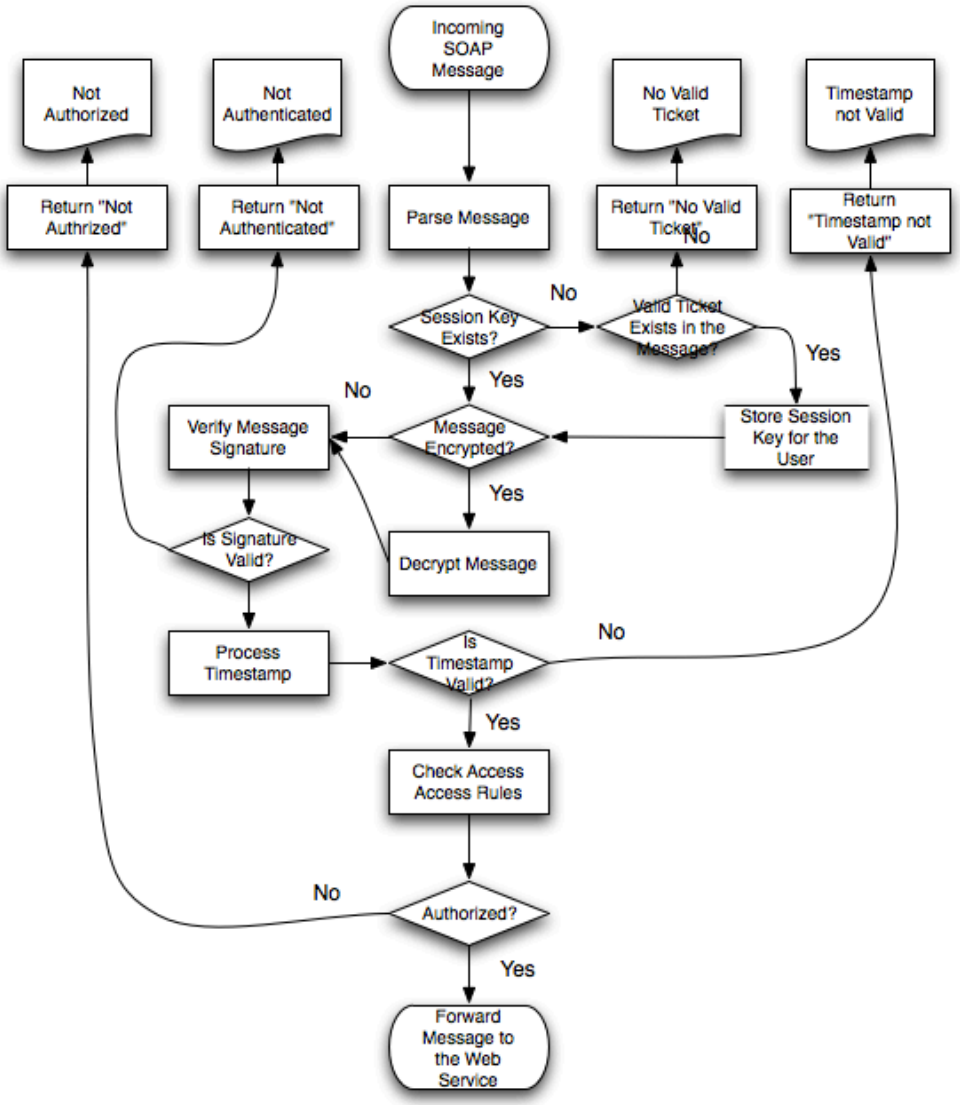
Στην περίπτωση της κρυπτογράφησης του μηνύματος μπορούμε, για λόγους απόδοσης, να παρακάμψουμε την πιστοποίηση μέσω ηλεκτρονικής υπογραφής. Στην γενική της περίπτωση η πρακτική αυτή δεν είναι ορθή, αλλά αν στο κρυπτογραφημένο μέρος του μηνύματος περιμένουμε συγκεκριμένη δομή όπως εντολή – όρισμα τότε η κρυπτογράφηση μπορεί να είναι αρκετή. Αν το μήνυμα δεν αποκρυπτογραφηθεί σωστά, δεν θα περιέχει έγκυρη εντολή και επομένως απορρίπτεται. Στην περίπτωση του Instrumentation Grid τα μηνύματα ελέγχου έχουν συνήθως προκαθορισμένη δομή και επομένως αρκεί η κρυπτογράφηση. Σε περίπτωση μεταφοράς αδόμητης πληροφορίας, π.χ. γραφήματα, εικόνες κτλ. τότε απαιτείται και ηλεκτρονική υπογραφή για την ακεραιότητα του μηνύματος.

Η χρονοσήμανση (timestamp) προστατεύει την υπηρεσία από επιθέσεις επανάληψης (replay attacks). Οι επιθέσεις αυτές προκαλούνται όταν ο επιτιθέμενος, ο

οποίος βρίσκεται στο δικτυακό μονοπάτι των μηνυμάτων, ακούει, αντιγράφει μηνύματα και τα οποία τα επαναλαμβάνει.

Μετά τον έλεγχο των ηλεκτρονικών υπογραφών, την αποκρυπτογράφηση του περιεχομένου και την πιστοποίηση του μηνύματος, ο ACM ελέγχει αν η συγκεκριμένη εντολή μπορεί να εκτελεσθεί με βάση του τοπικούς κανόνες πρόσβασης.

Το διάγραμμα καταστάσεων λειτουργίας του ACM απεικονίζεται στο Σχήμα 13.



Σχήμα 13: Διάγραμμα Καταστάσεων του ACM

Ο ACM χρησιμοποιεί μηχανισμούς που διατηρούν την κατάσταση ασφαλείας. Οι μηχανισμοί αυτοί διατηρούν την σύνοδο σε επίπεδο ασφάλειας μεταξύ του πελάτη και της υπηρεσίας με σκοπό να βελτιώσουν την απόδοση της ανταλλαγής μηνυμάτων. Στο πρώτο μήνυμα, το οποίο ορίζει τον συνοδό σε επίπεδο ασφαλείας μέσω του κλειδιού συνόδου, στέλνεται το εισιτήριο του Κέρβερου, ενώ στα επόμενα μόνο ένα hash προς αυτό το εισιτήριο, δεδομένου ότι ο ACM κράτησε το εισιτήριο από το πρώτο μήνυμα.

Τέλος όταν ο τοπικός διαχειριστής της υπηρεσίας αλλάζει κάποιους τοπικούς κανόνες, ο ACM ανανεώνει τους κανόνες στο Policy Repository.

4.4.2.4. Policy Repository

Το Policy Repository στην αρχιτεκτονική ασφαλείας παίζει τον ρόλο του συλλέκτη πληροφορίας ελέγχου πρόσβασης (authorization). Η πληροφορία αυτή όπως έχουμε αναφέρει μπορεί να προωθείται από τα διάφορα ACMs καθώς τροποποιείται. Εναλλακτικά, μπορεί οι τροποποιήσεις να γίνονται στο ίδιο το Policy Repository και να προωθούνται προς τα ACMs. Ακολουθώντας την δεύτερη προσέγγιση, το Policy Repository, λειτουργεί ουσιαστικά ως μια απλοποιημένη πλατφόρμα κατανεμημένου Policy Management [Slom94], δρώντας ως Policy Decision Point (PDP) [West01]. Το Policy Repository επιτρέπει την κεντρική επιλογή της πολιτικής πρόσβασης, ενώ τα ACMs λειτουργούν ως Policy Enforcing Points (PEP), όπου εφαρμόζεται η πολιτική. Στην πρώτη προσέγγιση, δηλαδή όταν οι κανόνες προωθούνται προς το Policy Repository, η επιλογή και εφαρμογή τους γίνεται τοπικά. Σε αυτήν την περίπτωση ο ACM παίζει τον ρόλο του PEP και PDP ταυτόχρονα. Στην δεύτερη προσέγγιση οι κανόνες καθορίζονται κεντρικά και υλοποιούνται από τους εν μέρει ACMs.

Η επιλογή μιας από τις δύο προσεγγίσεις εξαρτάται κάθε φορά από την φιλοσοφία ακολουθεί η αρχιτεκτονική του Υπολογιστικού Πλέγματος. Αν μια κεντρικοποιημένη αρχή, π.χ. ένας οργανισμός ελέγχει το Grid, τότε ενδείκνυται η προσέγγιση PEP στο Policy Repository και PDP στο ACM, αφού απλοποιεί την διαχείριση της πολιτικής πρόσβασης. Αλλιώς αν οι πόροι προς έλεγχο ανήκουν σε διαφορετικούς οργανισμούς, η πολιτική πρόσβασης ενδείκνυται να εφαρμόζεται τοπικά, από τους επιμέρους διαχειριστές των οργανισμών.

4.4.3. Ταυτότητες και Έλεγχος Πρόσβασης (Authorization)

Στην παρούσα ενότητα θα εμβαθύνουμε στους μηχανισμούς ελέγχου πρόσβασης και διαχείρισης ταυτοτήτων της προτεινόμενης αρχιτεκτονικής. Ο έλεγχος πρόσβασης στις λειτουργίες ενός οργάνου/συσκευής εξαρτάται από την πολιτική πρόσβασης που ορίζει ο οργανισμός στον οποίο ανήκει το όργανο, για τον κάθε Virtual Organization (VO) χρηστών του Grid.

Η ταυτότητα ενός χρήστη σε έναν οργανισμό απεικονίζεται στην προτεινόμενη αρχιτεκτονική ασφαλείας με ένα όνομα (principal_name). Αυτό προκύπτει από τον συνδυασμό του Μοναδικού Ονόματος (Distinguished Name – DN), του Εκδότη (Issuer) και του Σειριακού Αριθμού (Serial Number) στο πιστοποιητικό X.509. Θυμίζουμε ότι το X.509 χρησιμοποιείται για την αρχική ταυτοποίηση (authentication) του χρήστη στο Kerberos KDC, το οποίο στη συνέχεια υλοποιεί τη λειτουργία του Single Sign On (SSO) και χρησιμοποιεί συμμετρική κρυπτογραφία για βελτίωση της επίδοσης των λειτουργιών ασφαλείας. Για τον υπολογισμό του principal_name και τον τρόπο απεικόνισης του θα δούμε περισσότερα στην παράγραφο 4.4.4.

Στην αρχιτεκτονική ασφαλείας που προδιαγράφουμε δεν ορίζονται ταυτότητες μόνο για τους χρήστες αλλά και για τις υπηρεσίες. Ακολουθώντας την ορολογία του πρωτοκόλλου Kerberos, κάθε χρήστης ή υπηρεσία ανήκει σε μια περιοχή ασφαλείας (Realm) και αναγνωρίζεται μέσω μιας μοναδικής ταυτότητας (Principal). Αυτή ορίζεται ως ο παρακάτω συνδυασμός:

principal_name/instance@REALM
ή εναλλακτικά ακολουθώντας ορολογία
Υπολογιστικού Πλέγματος
principal_name/group@VO

Ο όρος *principal_name* ορίζει το όνομα του χρήστη ή της υπηρεσίας, ο όρος *instance* καθορίζει τον ρόλο ενός χρήστη/υπηρεσίας ως προς επιθυμητά αυξημένα δικαιώματα πρόσβασης σε πόρους του Grid (π.χ. admin, webmaster, resource owner) και REALM ή VO ορίζει τον οργανισμό που ανήκει. Το REALM είναι ορολογία του πρωτοκόλλου Kerberos, όπως είδαμε στην παράγραφο 3.4.2, και είναι ανάλογο του Virtual Organization (VO) στα Computational Grids. Η ύπαρξη των REALMs/VOs, όπως και σε όλα τα ιεραρχικά συστήματα ονοματοδοσίας, επιτρέπει ένα αναγνωριστικό χρήστη να ορίζεται μοναδικά σε ένα περιβάλλον που υπάρχουν ανεξάρτητοι οργανισμοί που ορίζουν τα ονόματα και τους ρόλους των

χρηστών/υπηρεσιών τους ανεξάρτητα ο ένας από τον άλλο. Το πρωτόκολλο Kerberos υλοποιεί μηχανισμούς ταυτοποίησης των χρηστών (authentication) και υποβοηθάει τον έλεγχο πρόσβασης (authorization). Το συνολικό σχήμα πρόσβασης επιβεβαιώνει τους ρόλους (instances) των χρηστών όπως είναι καταχωρημένοι σε βάση δεδομένων στο KDC, σύμφωνα και με τις αρχές του Role Based Access Control [RBAC].

Ο κάθε χρήστης μπορεί να έχει παραπάνω από ένα ρόλο (instances) με τον οποίο ελέγχεται η πρόσβασή του στους διαφορετικούς πόρους. Οι ομάδες με κοινούς κανόνες πρόσβασης σε κάθε πόρο είναι επιθυμητό να ταυτίζονται με τα ρόλους στα Virtual Organization του Grid, όπως αυτές ορίζονται σε έναν εξυπηρετητή **VOMS** (παράγραφο 3.4.3.2).

Σύμφωνα με την προτεινόμενη αρχιτεκτονική, ο τελικός έλεγχος πρόσβασης (authorization) γίνεται κατά περίπτωση από τον ACM των συγκεκριμένων πόρων και σύμφωνα με τοπικές πολιτικές. Οι ρόλοι (instances) που περιλαμβάνονται στο Principal αντανακλούν επιθυμίες ενός χρήστη καταχωρημένες στο KDC και δεν είναι υποχρεωτικοί για όλους τους πόρους. Συγκεκριμένα το KDC εκδίδει προς τους χρήστες εισιτήρια υπηρεσιών (Kerberos Service Tickets) τα οποία ενσωματώνουν την ταυτότητα του χρήστη (principal_name) τους επιθυμητούς ρόλους και το VO που ανήκει (principal_name/instance@VO). Η διαδικασία πρόσβασης ολοκληρώνεται στον ACM που διαβάζει το εισιτήριο του χρήστη για υπηρεσίες τις οποίες αυτός ελέγχει. Με τον τρόπο αυτόν υλοποιεί την ταυτοποίηση (authentication) ενός χρήστη και αναζητά σε τοπική λίστα κανόνων πρόσβασης, τον κανόνα που συμφωνεί με την ταυτότητα και την αίτηση του χρήστη (authorization). Ένας κανόνας πρόσβασης ορίζεται ως εξής:

principal_name – instance - VO - WS_URL – operation - operationAttribute

όπου η τριάδα < principal_name, instance, VO> είναι το Principal του χρήση, WS_URL είναι η διεύθυνση του Web Service του πόρου (π.χ. Instrument Element), operation είναι η συγκεκριμένη λειτουργία του πόρου που αντιστοιχίζεται σε συγκεκριμένη λειτουργία του Web Service και operationAttribute είναι οι παράμετροι που μπορούν να τεθούν/εκτελεστούν, εφόσον η λειτουργία απαιτεί παραμέτρους. Θυμίζουμε ότι μια υπηρεσία τύπου Web Service μπορεί να περιλαμβάνει πολλαπλές λειτουργίες (operations).

Ένας κανόνας ουσιαστικά ορίζει ένα χρήστη που ανήκει σε μία ομάδα/ρόλο (instance) και σε ένα REALM/VO, ο οποίος μπορεί να καλέσει μια λειτουργία με συγκεκριμένες παραμέτρους στο Web Service. Η προκαθορισμένη πολιτική πρόσβασης είναι η «άρνηση πρόσβασης» (default deny). Αυτό σημαίνει πως αν δεν βρεθεί κανόνας που την επιτρέπει την πρόσβαση τότε το μήνυμα SOAP απορρίπτεται.

Οι ρόλοι (instances) επιτρέπουν να έχουμε μειωμένο αριθμό κανόνων και επομένως βελτιωμένη απόδοση σε συστήματα με πολλούς χρήστες, αφού οι κανόνες πρόσβασης εφαρμόζονται από το ACM στις ρόλους αντί στους μεμονωμένους χρήστες.

4.4.4. Μηχανισμοί Αντιστοίχισης (Mapping Mechanisms)

Ένας από τις βασικές απαιτήσεις της αρχιτεκτονικής ασφαλείας είναι η διαλειτουργικότητα (interoperability) του Instrumentation Grid με τις υπάρχουσες υποδομές Computational Grid. Προς αυτή την κατεύθυνση χρησιμοποιούνται μηχανισμοί Αντιστοίχισης Ταυτότητας (Mapping Mechanisms) αλλά και η υποστήριξη διαπιστευτηρίων (proxy delegation) όπως θα δούμε στην επόμενη ενότητα.

Η προτεινόμενη αρχιτεκτονική ασφάλειας ορίζει μία περιοχή ασφαλείας (security domain), όπου το πρωτόκολλο Kerberos χρησιμοποιείται για την ταυτοποίηση των μηνυμάτων μεταξύ των οντοτήτων που συμμετέχουν σε αυτήν. Ωστόσο, ο χρήστης/υπηρεσία που προέρχεται από ένα περιβάλλον Computational Grid, αναγνωρίζεται με ένα Ηλεκτρονικό Πιστοποιητικό X.509 και χρειάζεται να αποκτήσει την αντίστοιχη ταυτότητα στην περιοχή ασφαλείας που ορίζει η αρχιτεκτονική μας.

Απαιτείται, λοιπόν, ο ορισμός ενός αλγορίθμου αντιστοίχισης της ταυτότητας του χρήστη από ψηφιακά πιστοποιητικά X.509 σε διαπιστευτήρια Kerberos. Ως αποτέλεσμα, ο χρήστης/υπηρεσία λαμβάνει μια ταυτότητα Kerberos Principal, όπως είδαμε στο 4.3, με την μορφή της τριάδας <principal_name, instance, VO>. Συγκεκριμένα, το πεδίο Virtual Organization VO εξάγεται από το πιστοποιητικό X.509 του GSI. Το instance που ορίζει τον ρόλο του χρήστη, μπορεί να αποσπαστεί από ένα VOMS Attribute Certificate του GSI. Τα VOMS Attribute Certificates [Cias06] είναι Proxy Certificates, τα οποία όμως περιέχουν και attributes που

«χαρακτηρίζουν» τον χρήστη ή την υπηρεσία και ουσιαστικά του αποδίδουν ένα ρόλο. Είναι ένα υπερσύνολο ενός πιστοποιητικού X.509 στο οποίο έχουν κωδικοποιηθεί τα χαρακτηριστικά (ρόλοι) του χρήστη στο VO. Με βάση τα παραπάνω ορίζουμε τον αλγόριθμο που επιτρέπει την κατασκευή ταυτότητας Kerberos από πιστοποιητικό X.509.

Ο **αλγόριθμος** είναι ο εξής: Εξάγουμε το μοναδικό αναγνωριστικό (Distinguished Name – DN) στο πιστοποιητικό X.509. Αυτό στην συνέχεια κατακερματίζεται με μια συνάρτηση κατακερματισμού (hashing function), π.χ. MD5 και κωδικοποιείται κατά Base64 από μορφή binary σε character oriented. Το αποτέλεσμα κατακερματισμού και κωδικοποίησης του DN χρησιμοποιείται ως ένα principal_name μήκους 224 bit. Ο κατακερματισμός γίνεται, επειδή το DN δεν είναι έγκυρο για χρήση ως Kerberos Principal, λόγο του ό,τι περιέχει μη έγκυρους χαρακτήρες όπως "/", κενά κτλ. Στην συνέχεια από το VOMS Attribute Certificate, εφόσον έχει παρασχεθεί, καθορίζουμε τον ρόλο του χρήστη (instance). Επειδή τα attributes, μπορεί να είναι πολλαπλά, στην προτεινόμενη αρχιτεκτονική, θεωρούμε το πρώτο attribute ως το ρόλο του χρήστη. Τέλος το principal συμπληρώνεται από το VO που το λαμβάνουμε είτε από το πιστοποιητικό X.509 είτε από το VOMS Attribute Certificate.

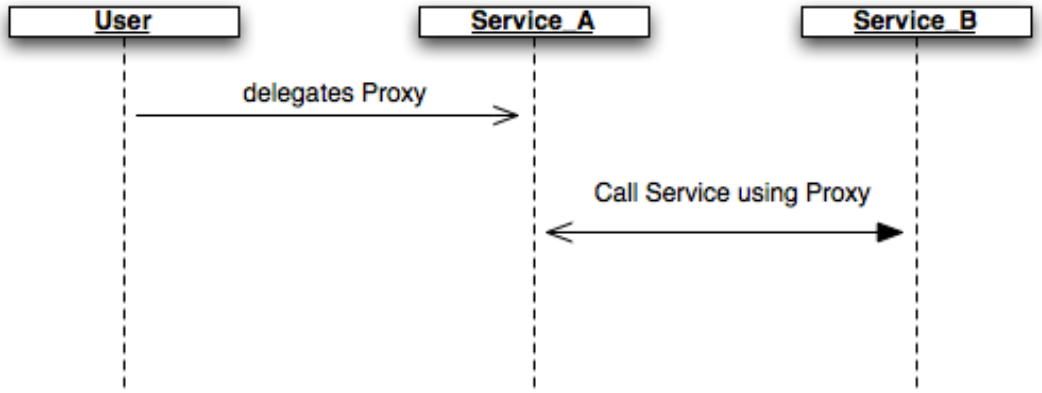
Ο παραπάνω αλγόριθμος υλοποιείται στο KrbClient και επιτρέπει την αυτόματη αντιστοίχιση του δοθέντος πιστοποιητικού σε Kerberos Principal. Η αντίστροφη διαδικασία, δηλαδή πως ένα Kerberos Principal αντιστοιχίζεται σε συγκεκριμένο πιστοποιητικό X.509 δεν μπορεί να αυτοματοποιηθεί λόγω της συνάρτησης κατακερματισμού που χρησιμοποιήθηκε κατά την παραγωγή του principal_name από το DN του πιστοποιητικού X.509. Υπενθυμίζεται, ότι όπως αναφέρθηκε στην παράγραφο 3.3.1, οι συναρτήσεις κατακερματισμού (hash functions) δεν είναι αμφίδρομες. Επομένως, η αντίστροφη διαδικασία που ολοκληρώνει την απαίτηση διαλειτουργικότητας με τα Computational Grids υλοποιήθηκε ενσωματώνοντας το DN στην επικεφαλίδα ασφαλείας του μηνύματος SOAP όπως θα δούμε στην επόμενη παράγραφο.

4.4.5. Υποστήριξη Αντιπροσώπευσης Διαπιστευτηρίων

Η αντιπροσώπευση διαπιστευτηρίων που έχει περιγραφεί στο [Welc04] είναι μια σημαντική λειτουργία σε Computational Grids. Επιτρέπει στον χρήστη να εκχωρεί την αντιπροσώπευση μέρους των δικαιωμάτων του σε μια άλλη οντότητα (proxy). Στο περιβάλλον του Instrumentation Grid που χρησιμοποιείται για τον έλεγχο οργάνων, όπως αυτό που θεωρούμε στην αρχιτεκτονική ασφαλείας της διατριβής, η αντιπροσώπευση διαπιστευτηρίων είναι υψηλής σημασίας, διότι επιτρέπει λόγου χάριν σε Instrument Elements να καλέσουν και να χρησιμοποιήσουν πόρους διασυνδεόμενου Computational Grid εκ' μέρους του χρήστη του Instrumentation Grid. Ένα χαρακτηριστικό παράδειγμα είναι όταν μια συσκευή στο Instrumentation Grid που συλλέγει πληθώρα στοιχείων (π.χ. μετεωρολογικός σταθμός), αντιπροσωπεύεται από μία υπηρεσία Instrument Element (IE) και πρέπει να αποθηκεύσει τα καταγραφέντα δεδομένα σε ένα Storage Element (SE) ενός Computational Grid χρησιμοποιώντας υπηρεσία GridFtp του SE.

Ο μηχανισμός αντιπροσώπευσης διαπιστευτηρίων που χρησιμοποιούμε είναι του gLite [gLite] που υλοποιεί το πρωτόκολλο αντιπροσώπευσης (Delegation Protocol) [DELEG], [Snel04]. Είναι μια υπηρεσία τύπου Web Service η οποία επιτρέπει σε έναν χρήστη να προωθήσει proxy certificate στον εξυπηρετητή όπου απαιτείται αντιπροσώπευση. Την υπηρεσία την ονομάζουμε **Υπηρεσία Αντιπροσώπευσης (Delegation Service)** και βρίσκεται στον ίδιο εξυπηρετητή με την Υπηρεσία που θα χρησιμοποιήσει τα προωθούμενα διαπιστευτήρια. Στο παράδειγμα της προηγούμενη παραγράφου, η υπηρεσία που καλεί το GridFtp ενός Storage Element, είναι το Instrument Element και λαμβάνει το proxy certificate από το Delegation Service που βρίσκεται στον ίδιο εξυπηρετητή.

Στην συνέχεια θα περιγράψουμε πως λειτουργεί ο μηχανισμός καθώς και τις τροποποιήσεις που κάναμε για να ενταχθεί στην προτεινόμενη αρχιτεκτονική.



Σχήμα 14: Μεταβίβαση Διαπιστευτηρίων (Delegate Proxy Certificate)

Στο Σχήμα 14 παρουσιάζουμε ένα σενάριο χρήσης (use case). Ας υποθέσουμε ότι μια υπηρεσία Service_A χρειάζεται να καλέσει μια άλλη υπηρεσία Service_B, εκ' μέρους ενός χρήστη. Απαιτείται ο χρήστης να μεταβιβάσει τα διαπιστευτήρια του στην υπηρεσία Service_A. Αυτό επιτυγχάνεται με τον χρήστη να παράγει ένα πληρεξούσιο πιστοποιητικό (proxy certificate), το οποίο προωθεί στην υπηρεσία Service_A και συγκεκριμένα στην Υπηρεσίας Αντιπροσώπευσής της, πραγματοποιώντας μια αυτόνομη κλήση σε αυτή (delegate Proxy). Στην συνέχεια η υπηρεσία Service_A μπορεί να χρησιμοποιήσει το πληρεξούσιο πιστοποιητικό για να επικοινωνήσει με την υπηρεσία Service_B (Call Service using Proxy).

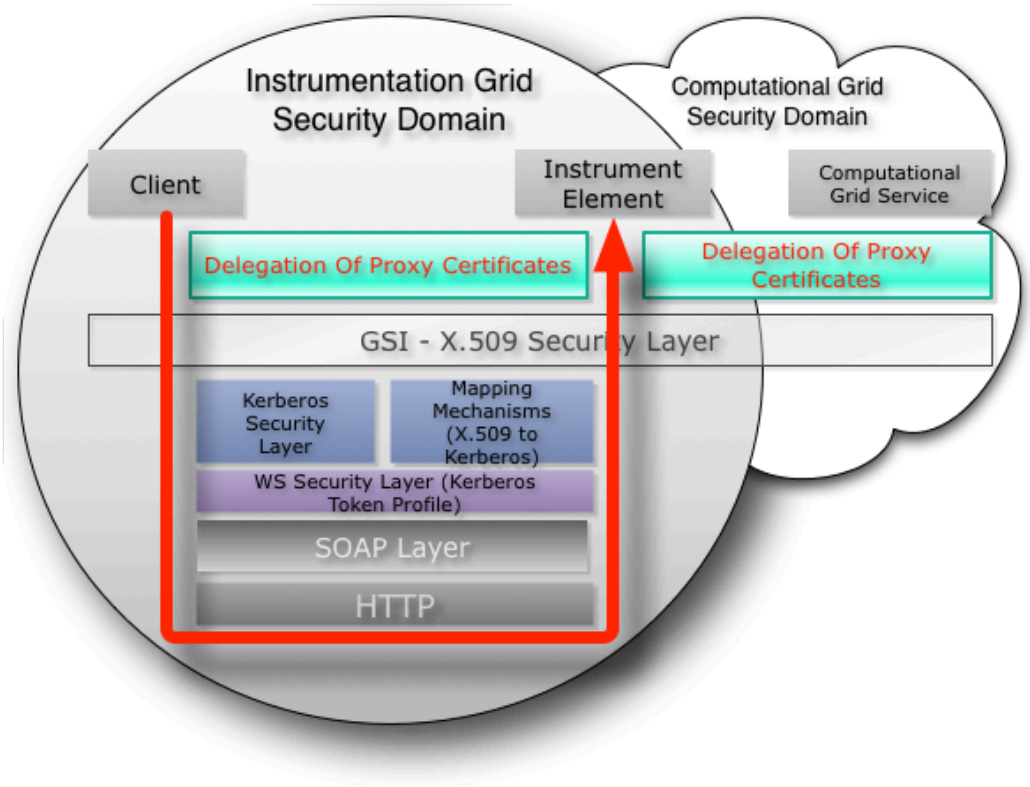
Το απλό σενάριο που περιγράψαμε παραπάνω βρίσκει εφαρμογή αρκετά συχνά στο περιβάλλον που προστατεύεται από την προτεινόμενη αρχιτεκτονική ασφαλείας. Παραδείγματος χάριν, με εντολή προς το IE που αντιπροσωπεύει μια συσκευή που παράγει δεδομένα (Service_A), ο χρήστης ζητά να αποθηκευτούν τα δεδομένα σε ένα Storage Element με την χρήση της υπηρεσίας GridFtp (Service_B) που βρίσκεται σε ένα SE.

Ένα IE λαμβάνει πολλές εντολές από πολλούς χρήστες. Πρέπει να του παρέχεται η δυνατότητα να διαχωρίζει για κάθε εντολή ποιος χρήστης την έχει εκτελέσει. Μέσω της αρχιτεκτονικής μας γνωρίζει την ταυτότητα principal_name/instance@VO του πρωτοκόλλου Kerberos. Ωστόσο για να εκτελέσει την μεταφορά πρέπει να γνωρίζει και την ταυτότητα του χρήστη στο Computational Grid κατά GSI, όπως ορίζεται από το Distinguished Name (DN) του X.509. Μέσω αυτού μπορεί το IE να ανακτήσει από την Υπηρεσία Αντιπροσώπευσης το αντίστοιχο proxy certificate και να καλέσει μια υπηρεσία στο Computational Grid (π.χ. GridFtp).

Στην υλοποίηση του gLite, το DN μεταφέρεται από το πελάτη στην υπηρεσία IE μέσω του πιστοποιητικού του και πρωτοκόλλου https (http over SSL). Ωστόσο στην αρχιτεκτονική που προτείνουμε, δεν χρησιμοποιούμε SSL. Η αρχιτεκτονική βασίζεται σε WS Security για την μεταφορά μηνυμάτων. Για να μπορέσουμε να μεταφέρουμε το DN, το ενσωματώνουμε στην επικεφαλίδα WS Security Header του μηνύματος SOAP. Η επικεφαλίδα περιέχει επίσης το Kerberos Principal, το οποίο είναι το κατατμημένο DN, όπως είδαμε στην παράγραφο 4.4.4. Ο ACM της υπηρεσίας Service_A επαληθεύει την κατάτμηση του DN (Kerberos Principal) με το DN όπως έχει εισαχθεί στην επικεφαλίδα. Ακολουθεί στην συνέχεια η διαδικασία του Σχήμα 14 όπως περιγράφηκε παραπάνω.

4.5. Ένταξη προτεινόμενης αρχιτεκτονικής ασφαλείας σε ένα γενικό πλαίσιο διαλειτουργικότητας

Στην παρούσα ενότητα θα μελετήσουμε πως εντάσσεται η προτεινόμενη αρχιτεκτονική ασφαλείας σε ένα γενικό πλαίσιο διαλειτουργικότητας. Θεωρούμε ένα Instrumentation Grid σε συνεργασία με ένα Computational Grid, που ακολουθούν τις αρχές SOA και των τεχνολογιών Web Services.



Σχήμα 15: Διαστρωμάτωση των πρωτοκόλλων Ασφάλειας της προτεινόμενης Αρχιτεκτονικής

Στο Σχήμα 15 παρουσιάζεται η στοίβα των πρωτοκόλλων μεταφοράς και ασφαλείας που χρησιμοποιεί η αρχιτεκτονική. Τα πρωτόκολλα αυτά επιτρέπουν την λειτουργία της και ορίζουν την διαλειτουργικότητα με άλλες υποδομές Computational Grid. Το βέλος στο σχήμα καθορίζει τη κατεύθυνση των αιτήσεων. Στην ανάλυσή μας θεωρούμε τρεις συμμετέχοντες: Τον **Client**, το **Instrument Element** (παρέχει τις λειτουργίες ελέγχου ενός οργάνου) και το **Computational Grid Service** (παρέχει Υπηρεσία ενός κλασικού Grid, όπως το GridFtp). Ως **Instrumentation Security Domain** ορίζεται η περιοχή ασφάλειας που προστατεύεται από την αρχιτεκτονική ασφαλείας μας. Το Computational Grid Service είναι εκτός αρχιτεκτονικής και προστατεύεται με Grid Security Infrastructure (GSI). Τα επίπεδα των πρωτοκόλλων της αρχιτεκτονικής είναι (από κάτω προς τα πάνω):

- **HTTP**: Είναι το πρωτόκολλο μεταφοράς πάνω στο οποίο μεταφέρονται τα μηνύματα SOAP από τον Client στο Instrument Element
- **SOAP Layer:** Αποτελεί το επίπεδο που παρέχει την λειτουργία της ανταλλαγής δομημένης πληροφορίας με την μορφή αυτόνομων μηνυμάτων. Ορίζει την μορφή του μηνύματος που περιέχει τον Φάκελο (SOAP:Envelope) που περιέχει μία επικεφαλίδα (SOAP:Header) και το σώμα του μηνύματος (SOAP:Body). Στο σώμα του μηνύματος SOAP μπαίνει η αίτηση του πελάτη προς το Instrument Element
- **WS Security Layer (Kerberos Token Profile):** Ορίζει μια επικεφαλίδα ασφαλείας (Security Header) που ενσωματώνεται στη επικεφαλίδα του μηνύματος SOAP και προδιαγράφει πως τα διαπιστευτήρια του πρωτοκόλλου Kerberos μεταφέρονται μέσα στα μηνύματα SOAP και πως η πληροφορία μέσα στο σώμα του μηνύματος κρυπτογραφείται και υπογράφεται. Επίσης μέσω της ενσωμάτωσης του DN του αντίστοιχου X.509 διευκολύνει τον μηχανισμό αντιπροσώπευσης διαπιστευτηρίων, όπως είδαμε στην παράγραφο 4.4.5. Οι λειτουργίες αυτού του επιπέδου παρέχονται στη πλευρά του πελάτη από τον KrbClient και στην πλευρά του Instrument Element από τον ACM
- **Kerberos Security Layer – Mapping Mechanisms (X.509 to Kerberos):** Επιτρέπει την ταυτοποίηση του κάθε μηνύματος SOAP χρησιμοποιώντας το πρωτόκολλο Kerberos. Σε αυτό το επίπεδο βρίσκεται

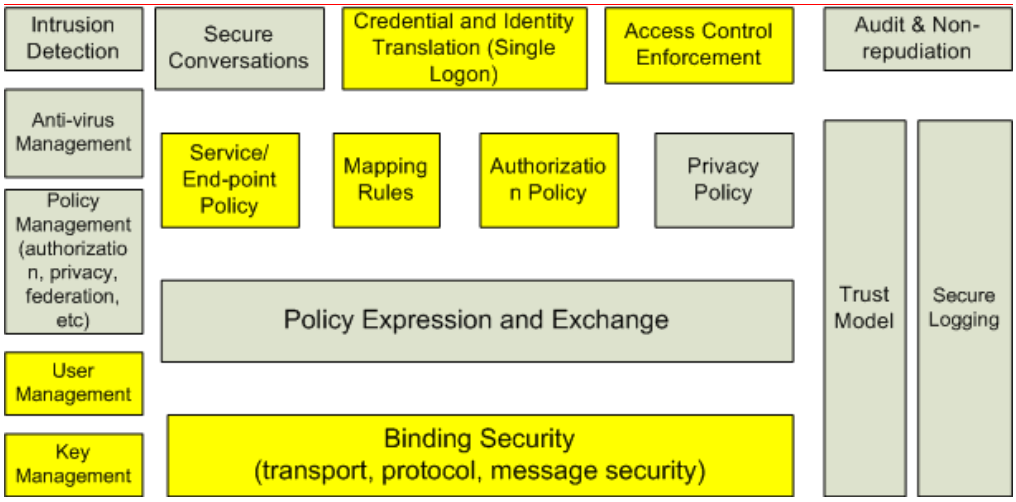
η διανομή των κλειδιών μέσω των εισιτηρίων υπηρεσιών που εκδίδει το KDC. Οι μηχανισμοί αντιστοίχισης (Mapping Mechanisms) παρέχουν την μηχανισμό αντιστοίχισης ταυτότητας από X.509 σε Kerberos Principal. Οι λειτουργίες αυτού του επιπέδου υλοποιούνται από τα KrbClient και ACM σε συνεργασία με το Kerberos KDC

- **GSI, X.509 Security Layer**: Ορίζει το επίπεδο GSI που εκτείνεται σε όλο το Grid, συμπεριλαμβανομένου και του Instrumentation Security Domain της αρχιτεκτονικής μας. Καθορίζεται από το πιστοποιητικό ή το proxy X.509 του χρήστη που χρησιμοποίησε μέσω του PKINIT για την αρχική ταυτοποίηση στην αρχιτεκτονική ασφαλείας που αφορά στο Instrumentation Grid όπως είδαμε με λεπτομέρεια στην παράγραφο 4.4.1
- **Delegation of Proxy Certificates**: Είναι το επίπεδο όπου ένα πιστοποιητικό μεταβιβάζεται από μια υπηρεσία σε μια άλλη επιτρέποντας της να δρα εκ μέρους του χρήστη του Instrumentation Grid σε υπηρεσίες του Computational Grid. Παρέχεται από το Delegation Service όπως είδαμε στην παράγραφο 4.4.5.

Στην συνέχεια εξετάζουμε την συμβατότητα της αρχιτεκτονικής μας με τις αρχές του πλαισίου αναφοράς OGSA που αναφερθήκαμε στην ενότητα 2.4.3. Το **Open Grid Services Architecture (OGSA)** ορίζει τα συστατικά των Computational Grids ως τμήματα μιας Υπηρεσιοστρεφούς Αρχιτεκτονικής (SOA) με την χρήση Web Services ως την κύρια υποδομή. Σημειώνουμε πως στα πλαίσια του OGSA δεν ορίζονται μηχανισμοί σε χαμηλότερα επίπεδα (π.χ. Kerberos, PKI κτλ.), αλλά παρέχεται μια υψηλής εποπτείας ανάλυση των τμημάτων και των λειτουργιών που πρέπει να προσφέρει η αρχιτεκτονική, ώστε να διευκολυνθεί η ανάπτυξη τέτοιων συστημάτων.

Η βασική θεωρία για το πλαίσιο OGSA τέθηκε από το I. Foster [Fost02a]. Η τρέχουσα έκδοσή της [Fost06b] ορίζει μια ομάδα από επτά βασικές λειτουργικότητες (core capabilities) που απαιτούνται για να υποστηριχθούν Σύστημα Υπολογιστικού Πλέγματος και εφαρμογών: *Infrastructure Services, Execution Management Services, Data Services, Resource Management Services, Security Services, Self-Management Services και Information Services*. Κάθε μία από τις βασικές λειτουργικότητες καλύπτει μια συγκεκριμένη περιοχή της αρχιτεκτονικής ενός Computational Grid.

Όσον αφορά τις Υπηρεσίες Ασφαλείας (Security Services) [Naga08], ορίζονται συγκεκριμένες λειτουργικές δυνατότητες που περιλαμβάνουν την *Ταυτοποίηση (Authentication), Αντιστοίχηση Ταυτότητας (Identity Mapping), Έλεγχος Πρόσβασης (Authorization), Μετατροπή Διαπιστευτηρίων (Credential Conversion), Έλεγχος Διαδικασιών Ασφαλείας (Audit) και Ασφαλής Καταγραφή Γεγονότων (Secure Logging)*. Η αρχιτεκτονική OGSA καθορίζει ένα περιβάλλον διαλειτουργικότητας μεταξύ διαφορετικών περιοχών ασφαλείας (security domains), οι οποίες βασίζονται σε διαφορετικές αρχές ασφάλειας, π.χ. πιστοποιητικά τύπου X.509 και Kerberos Tokens, και περιγράφει τις υπηρεσίες που επιτρέπουν αυτήν την λειτουργικότητα.



Σχήμα 16: Γενικά Στοιχεία ενός Μοντέλου Ασφάλειας Υπολογιστικού Πλέγματος [Fost06b]

Στο Σχήμα 16 παρουσιάζονται τα στοιχεία του μοντέλου Ασφαλείας του Computational Grid, όπως ορίζονται στην OGSA. Έχουμε επισημάνει με κίτρινο χρώμα τα μέρη της OGSA που αντιστοιχούν σε λειτουργίες της αρχιτεκτονικής μας. Συγκεκριμένα:

- **Credential and Identity Translation (Single Logon) (Μετατροπή Διαπιστευτηρίων και Ταυτότητας):** Πραγματοποιείται με την χρήση του συστήματος ταυτοποίησης του Kerberos και του μηχανισμού PKINIT, οι οποίοι ταυτοποιούν και παρέχουν εισιτήρια υπηρεσιών σε χρήστες που αναγνωρίζονται με πιστοποιητικό. Μπορούν στην συνέχεια χρησιμοποιώντας τα εισιτήρια να έχουν πρόσβαση σε διάφορους κατανεμημένους πόρους μέσα στο Instrumentation Grid, χωρίς να χρειάζεται να επανασυνδεθούν (re-login), δηλαδή υλοποιούν την διαδικασία Single-Sign-On (SSO).

- **Mapping Rules (Κανόνες Απεικόνισης):** Επιτρέπουν την αντιστοίχιση GSI πιστοποιητικών και X.509 proxy certificates που συμμορφώνονται με το RFC 3820 σε Kerberos Principals. Τους κανόνες τους παρουσιάσαμε στην παράγραφο 4.4.4. Η αντίθετη διαδικασία διευκολύνεται από την αρχιτεκτονική μας με την ενσωμάτωση του DN του X.509 στην επικεφαλίδα ασφαλείας του μηνύματος SOAP.
- **Access Control Enforcement (Επιβολή Μηχανισμών Πρόσβασης):** Πραγματοποιείται στον ACM μέσω των τοπικών κανόνων του κάθε πόρου (resource) που δημοσιεύονται στο Policy Repository.
- **Service End-point Policy (Πολιτική Υπηρεσίας):** Περιγράφεται μέσω του συνόλου των τοπικών κανόνων πρόσβασης.
- **Authorization Policy (Πολιτική Πρόσβασης):** Επιτρέπει μόνο σε χρήστες καθορισμένων ρόλων να χειρίζονται τις λειτουργίες των οργάνων – συσκευών. Υλοποιείται μέσω των τοπικών κανόνων πρόσβασης.
- **User and Key Management (Διαχείριση Χρηστών και Κλειδιών):** Πραγματοποιείται στο Kerberos KDC. Μέσω της υπηρεσίας TGS και του μηχανισμού του service ticket, μοιράζονται τα κλειδιά στους συναλλασσόμενους.
- **Bindings Security (Δεσμεύσεις Μηχανισμών Ασφαλείας):** Πραγματοποιούνται με την υιοθέτηση του πρωτοκόλλου WS Security Kerberos Token Profile, για την μεταφορά των διαπιστευτηρίων (Kerberos service tickets) και την παροχή ασφάλειας στα μηνύματα SOAP που ανταλλάσσονται μεταξύ του πελάτη και της υπηρεσίας.

Επίσης, πρέπει να επισημάνουμε πως το πλαίσιο OGSA δεν ορίζει την υποχρεωτική υιοθέτηση όλων των αρχών του, αλλά ένα υποσύνολο είναι αρκετό για να χαρακτηρίσει ένα Grid ως συμβατό με το OGSA. Αρκεί να χρησιμοποιεί τις βασικές αρχές όπως ορίζονται στα OGSA/SOA. Η αρχιτεκτονική ασφαλείας της διατριβής ικανοποιεί ένα επαρκές υποσύνολο και μπορεί εύκολα να συνυπάρξει με ένα Computational Grid που ακολουθεί τις αρχές OGSA.

5. Υλοποίηση Αρχιτεκτονικής

5.1. Τεχνολογίες που χρησιμοποιήσαμε

Στο προηγούμενο κεφάλαιο παρουσιάσαμε την Αρχιτεκτονική Ασφαλείας που προτείνεται στην διατριβή και εξηγήσαμε την λειτουργία της. Στο παρόν κεφάλαιο θα παρουσιάσουμε μια πρότυπη υλοποίηση της αρχιτεκτονικής ασφαλείας, η οποία βασίζεται σε Java SE 1.4. Ο κώδικας βρίσκεται αναρτημένος στο αποθετήριο κώδικα που χρησιμοποιεί το σύστημα CVS [CVS]:

http://ulisse.elettra.trieste.it/cgi-bin/viewcvs.cgi/gridcc/wp2/security/

Η ανάπτυξή του έχει γίνει με την χρήση του περιβάλλοντος ανάπτυξης ανοιχτού λογισμικού Eclipse IDE [ECLIPSE].

Η αρχιτεκτονική ασφαλείας που προτείνεται στην διατριβή, η οποία παρουσιάστηκε στο προηγούμενο κεφάλαιο, αποτελείται από τα εξής τμήματα:

- **Kerberos KDC**
- **KrbClient**
- **ACM**
- **Policy Repository**

Ως **KDC** χρησιμοποιήσαμε τον **Heimdal Kerberos** [HEIMDAL], εξαιτίας του γεγονότος ότι ήταν ο μοναδική υλοποίηση ανοιχτού λογισμικού που υποστήριζε, όταν ξεκινήσαμε την υλοποίηση, την λειτουργία ταυτοποίησης με την χρήση ηλεκτρονικού πιστοποιητικού, δηλαδή το πρωτόκολλο PKINIT [Zhu06]. Ωστόσο η λειτουργία αυτή υποστηρίζεται πλέον και από την πιο διαδεδομένη υλοποίηση ανοιχτού λογισμικού, αυτή του **MIT Kerberos** [MITKERB]. Η επιλογή του KDC γενικά δεν επηρεάζει την αρχιτεκτονική αρκεί να ακολουθεί το πρότυπο Kerberos και τα αντίστοιχα RFC για πιστοποιητικά X.509 και proxy certificates κατά RFC 3820 [Tuec04].

Τα υπόλοιπα τμήματα της αρχιτεκτονικής υλοποιήθηκαν σε γλώσσα προγραμματισμού Java SE 1.4. Θα παρουσιάσουμε τις βασικές βιβλιοθήκες, οι οποίες δίνουν την απαραίτητη υποδομή της αρχιτεκτονικής:

- **Axis 1.3 [AXIS]**: Παρέχει τις λειτουργίες και το υπόβαθρο Web Services στο λογισμικό που αναπτύξαμε. Επιτρέπει ο κώδικας που γράφουμε να μπορεί να χρησιμοποιηθεί από ένα Web Service. Επιπλέον παρέχει όλες τις αναγκαίες συναρτήσεις στην επεξεργασία των μηνυμάτων SOAP.
- **WSS4J [WSS4J]**: Είναι η βασική βιβλιοθήκη που παρέχει την λειτουργίες του πρωτοκόλλου Web Services Security SOAP Message Security 1.1.
- **BouncyCastle [BONCA]**: Ακολουθεί το Java Cryptography Architecture (JCA) και το Java Cryptography Extensions (JCE) και παρέχει την υλοποίηση όλων των κρυπτογραφικών αλγορίθμων συμμετρικού και δημοσίου κλειδιού.

Εκτός από τις παραπάνω βιβλιοθήκες χρησιμοποιήσαμε ένα μεγάλο πλήθος άλλων βιβλιοθηκών οι οποίες εξαρτώνται από αυτές. Για λόγους απλότητας τις παραλείψαμε, ωστόσο μπορεί να βρεθούν στο αποθετήριο κώδικα CVS.

Η υλοποίηση ήταν εφικτή με την χρήση μιας επιπλέον βιβλιοθήκης, της Kerberos15.jar. Αυτήν την δημιουργήσαμε εμείς από το πηγαίο κώδικα της Java SE 1.5. Ο λόγος που έγινε αυτό είναι για να έχουμε πλήρη πρόσβαση στο Generic Security Services Application Program Interface – **GSS-API** [Wray00], το οποίο είναι ένα **γενικό API για ταυτοποίηση πελάτη/εξυπηρετητή**. Από την έκδοση 5 του πρωτοκόλλου Kerberos, υποστηρίζεται το GSS-API [Zhu05], επομένως οποιοδήποτε λογισμικό που το υποστηρίζει, υποστηρίζει ταυτόχρονα και την ταυτοποίηση μέσω του πρωτοκόλλου Kerberos. Το πρόβλημα είναι πως το GSS-API όπως περιέχεται στην Java, δεν εμπεριέχει το πρωτόκολλο PKINIT. Επομένως υπήρχε η ανάγκη να καλέσουμε κάποιο εξωτερικό λογισμικό και συγκεκριμένα την init που παρέχει ο Heimdal Kerberos, για να πραγματοποιήσουμε αρχική ταυτοποίηση (βήμα a. στην παράγραφο 3.4.2) και αφού λάβουμε το TGT στην συνέχεια να χρησιμοποιήσουμε τις υπόλοιπες κλήσεις από το GSS-API. Ωστόσο το GSS-API όπως υλοποιείται στην Java δεν επιτρέπει την πρόσβαση σε επιμέρους λειτουργίες (με χρήση obfuscated code, δηλαδή αλλάζοντας τα κατανοητά ονόματα συναρτήσεων με τυχαίο τρόπο). Κάνοντας πάλι build τον πηγαίο κώδικα της Java, είχαμε στην διάθεση μας τις αναγκαίες λειτουργίες του GSS-API.

Στην συνέχεια του κεφαλαίου θα δώσουμε περισσότερες λεπτομέρειες για υλοποίηση της αρχιτεκτονικής ασφαλείας και συγκεκριμένα για τα τμήματα

λογισμικού server - ACM, client - KrbClient και Policy Repository. Όλα αυτά έχουν ενσωματωθεί, για μεγαλύτερη ευκολία χρήσης από άλλα συστήματα, σε μια βιβλιοθήκη Java την gridcc-sec.jar. Η υλοποίηση υιοθέτησε την λογική των handlers ώστε να υπάρχει ευελιξία στην ανάπτυξη (modular, component based architecture) και να αποφευχθεί μια μονολιθική αρχιτεκτονική.

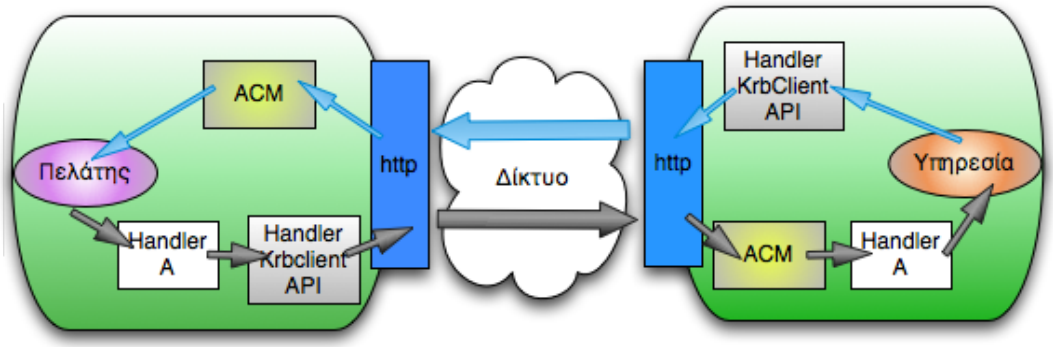
5.2. Handlers

Οι Handlers που ονομάζονται και Interceptors στα διάφορα προγραμματιστικά περιβάλλοντα (frameworks), χρησιμοποιούνται για να χτίζονται υπηρεσίες. Χρησιμοποιήθηκαν Web Service handlers του περιβάλλοντος AXIS 1.x. Παρόμοιοι handlers υπάρχουν στις επόμενες εκδόσεις του AXIS [AXIS2] και στο νεότερο Apache CXF [CXF]. Αν και δεν είναι απόλυτα συμβατοί μεταξύ τους, η έννοια των handlers/interceptors παραμένει κοινή [ICXF]. Αποτελούν το βασικό δομικό συστατικό στην δημιουργία μιας υπηρεσίας και επιτρέπουν την κατασκευή επαναχρησιμοποιήσιμων τμημάτων λογισμικού, που επιτελούν μια συγκεκριμένη επεξεργασία σε ένα εισερχόμενο ή εξερχόμενο μήνυμα μιας υπηρεσίας. Διακρίνονται σε κοινούς Java handlers που υλοποιούν συγκεκριμένη προγραμματιστική λογική, και handlers πρωτοκόλλων όπως του SOAP, όπου επεξεργάζονται μέρος τους μηνύματος, σύμφωνα με το πρωτόκολλο. Συνδέονται ο ένας μετά τον άλλο σε μορφή αλυσίδας. Δηλαδή η έξοδος ενός handler είναι η είσοδος στον επόμενο στην αλυσίδα.

Σε μια υπηρεσία που χρησιμοποιεί το πρωτόκολλο http για την μεταφορά των μηνυμάτων SOAP, ο τελευταίος handler στον πελάτη και ο πρώτος handler στον εξυπηρετητή στην αποστολή/λήψη ενός μηνύματος SOAP είναι ο http handler. Στην μεριά του πελάτη, ο http handler λαμβάνει το μήνυμα και το ενσωματώνει σε ένα http request, προσθέτοντας την επικεφαλίδα http. Στην μεριά του εξυπηρετητή κάνει την αντίθετη διαδικασία, παραδίδοντας το μήνυμα SOAP στον επόμενο handler, παράγοντας παράλληλα αντικείμενο Java που περιέχει την επικεφαλίδα του http request.

Στην συνέχεια στο Σχήμα 17 παρουσιάζεται ένα υποθετικό παράδειγμα όπου έχουμε μια ανταλλαγή μηνυμάτων SOAP (request-response), κατά την οποία η υπηρεσία και ο πελάτης επικοινωνούν με ασφάλεια χρησιμοποιώντας τους handlers ACM και KrbClient API που υλοποιούν τις λειτουργίες των ACM και KrbClient που περιγράψαμε στο προηγούμενο κεφάλαιο. Εκτός από αυτούς τους δύο handlers που

προσφέρουν τις ζητούμενες λειτουργίες ασφάλειας, συμμετέχει στην όλη επεξεργασία και ένα Handler A. Αυτός επιτελεί μια άλλη λειτουργία στο μήνυμα SOAP, η οποία μας είναι αδιάφορη για το παράδειγμά μας. Γενικά οι handlers που είναι υπεύθυνοι για την ασφάλεια ενός μηνύματος SOAP τοποθετούνται τελευταίοι στην μεριά του πελάτη και πρώτοι στην μεριά του εξυπηρετητή, εξαιρώντας πάντα τον http handler.



Σχήμα 17: Παράδειγμα ενός πελάτη με μια Υπηρεσία με την χρήση των handlers της Αρχιτεκτονικής Ασφαλείας

Ο λόγος που ακολουθείται η συγκεκριμένη σειρά στους security handlers (τελευταίος handler στον client – πρώτος handler στον server), όπως βλέπουμε και στο Σχήμα 17, είναι ότι ένας security handler μπορεί να αλλάξει δραστικά ένα μήνυμα ώστε να μην διαβάζονται τα στοιχεία του μηνύματος (xml elements), όταν παραδείγματος χάριν κρυπτογραφεί το σώμα του μηνύματος SOAP. Μετά την κρυπτογράφηση, κανένας handler δεν θα μπορεί να τα επεξεργαστεί τα κρυπτογραφημένα στοιχεία του μηνύματος. Ο σημαντικότερος λόγος όμως είναι πως εάν ο security handler εφαρμόζει ηλεκτρονική υπογραφή στο μήνυμα, τότε δεν μπορεί να γίνει καμία αλλαγή μετά την εφαρμογή της στο μήνυμα από κανέναν και επομένως πρέπει να είναι ο τελευταίος handler της αλυσίδας στην πλευρά του client και πρώτος στην πλευρά του server. Αλλιώς θα ακυρωθεί η εγκυρότητα της υπογραφής αφού το μήνυμα θα έχει αλλάξει και επομένως θα απορριφθεί από τον παραλήπτη.

5.3. Υλοποίηση του ACM

Η υλοποίηση του **Access Control Manager (ACM)** της αρχιτεκτονικής μας πραγματοποιήθηκε ως Axis Handler. Ένας Handler είναι ουσιαστικά ένας

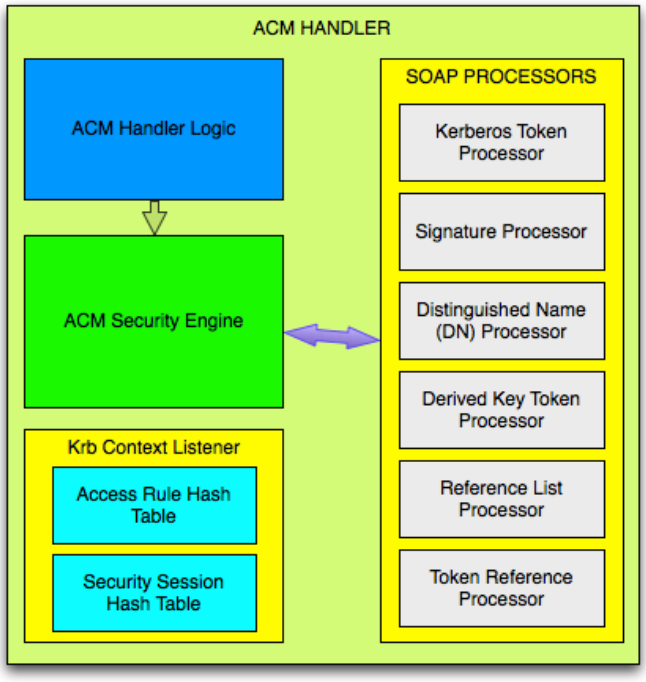
προεπεξεργαστής ενός μηνύματος SOAP. Υπενθυμίζεται ότι ο ACM υλοποιεί το server τμήμα της αρχιτεκτονικής μας.

Ο ACM handler έχει ακολουθήσει την δομή του Web Services Security for Java (WSS4J) security handler [WSS4J]. Χρησιμοποιεί την βιβλιοθήκη WSS4J, η οποία υποστηρίζει το πρότυπο WSS όπως είδαμε στην παράγραφο 3.5.2 και συγκεκριμένα τα profiles:

- Web Services Security: SOAP Message Security 1.1
- Username Token Profile 1.1
- X.509 Certificate Token Profile 1.1

Ο WSS4J handler δεν υποστηρίζει το πρωτόκολλο Kerberos Token Profile του WSS. Στην υλοποίηση του ACM, επεκτείναμε την βιβλιοθήκη WSS4J και προσθέσαμε την απαραίτητη υποστήριξη του Kerberos Token Profile.

Στο Σχήμα 18 παρουσιάζεται σχηματικά η αρχιτεκτονική του ACM handler που υλοποιήσαμε.



Σχήμα 18: Αρχιτεκτονική του ACM

Ο ACM handler υλοποιεί την κεντρική λογική (ACM Handler Logic) όπως αυτή παρουσιάστηκε στο Σχήμα 13 στην παράγραφο 4.4.2.3. Ταυτοποιεί το μήνυμα,

ελέγχει την πρόσβαση σύμφωνα με τους κανόνες πρόσβασης. Χρησιμοποιεί τα εξής υποσυστήματα:

- **KrbContext Listener**: Το αντικείμενο αυτό ορίζει μέσα σε Java Servlet Engine που λειτουργεί ως container για το Web Service (στην υλοποίηση μας χρησιμοποιήσαμε για container τον Tomcat 5.28 [TOMCAT]) την κοινή πληροφορία που μοιράζονται όλα τα στιγμιότυπα του ACM handler που μπορεί να τρέχουν ταυτόχρονα (ένα για κάθε εισερχόμενο μήνυμα). Αρχικοποιεί τον ACM handler, π.χ. διαβάζει το Kerberos Key του ACM, διαβάζει όλες τις μεταβλητές περιέχονται στο ACMHandler.properties, διαβάζει τους τοπικούς κανόνες και τους αποστέλλει στο Policy Repository και δημιουργεί τα αντικείμενα *Security Session Hash Table* και *Access Rule Hash Table*. Τα δύο αυτά αντικείμενα είναι πίνακες κατακερματισμού (hash tables). Ο πρώτος πίνακας περιέχει όλη την πληροφορία για την κατάσταση (state) κάθε συνόδου, όπως το Kerberos Principal, το sessionKey και την ημ/νια λήξης του ticket. Ο δεύτερος πίνακας αφορά το authorization και περιέχει όλους τους κανόνες (Access Rules) που φυλάσσονται στο αρχείο rules.properties τύπου Java properties file. Τέλος η αποστολή των τοπικών κανόνων στο Policy Repository γίνεται με την χρήση αντίστοιχου Web Service (δες παράγραφο 5.5).
- **ACM Security Engine**: Είναι η κύρια μηχανή επεξεργασίας του μηνύματος SOAP. Ακολουθεί το προγραμματιστικό μόρφημα (Programming Pattern) Singleton. Το μόρφημα Singleton, όταν ακολουθείται από ένα αντικείμενο, ορίζει πως μπορεί να υπάρχει μόνο ένα στιγμιότυπο του αντικειμένου σε μια Εικονική Μηχανή Java (Java Virtual Machine - JVM). Διασφαλίζει ότι κάθε μήνυμα θα το επεξεργαστεί το ίδιο αντικείμενο, δηλαδή η μηχανή ACM Security Engine στην περίπτωσή μας. Επομένως διατηρείται η κατάσταση σε διαδοχική αποστολή μηνυμάτων και αυτά παραδίδονται στην υπηρεσία με την σειρά άφιξης τους. Η επεξεργασία των διαφόρων λειτουργιών στο μήνυμα SOAP, γίνεται από τους διάφορους SOAP Security Processors. Η μηχανή φροντίζει την ορθή σειρά εκτέλεσής τους, ελέγχοντας την επικεφαλίδα ασφαλείας του μηνύματος SOAP.

- **SOAP Security Processors**: εκτελούν συγκεκριμένες λειτουργίες στο μήνυμα SOAP. Συγκεκριμένα:
  - ***Kerberos Token Processor***: Ελέγχει αν υπάρχουν Kerberos Tokens στο μήνυμα SOAP, πιστοποιεί την ταυτότητα του χρήστη με βάση το token (εισιτήριο - Kerberos ticket στην περίπτωση αυτή) και αποθηκεύει στο Security Session Hash Table το session key και τα στοιχεία του εισιτηρίου. Με αυτόν το τρόπο ορίζεται μια σύνοδος Kerberos στο Security Session Hash Table με βάση το κλειδί της συνόδου και στα επόμενα μηνύματα δεν είναι αναγκαία η αποστολή του Kerberos token.
  - ***Signature Processor***: Ελέγχει το είδος υπογραφής (digital signature ή MAC) για ένα στοιχείο του μηνύματος και καλεί τον αντίστοιχο processor για να επιβεβαιώσει την υπογραφή.
  - ***Distinguished Name (DN) Processor***: Βρίσκει το DN του χρήστη που αντιστοιχεί σε ένα ηλεκτρονικό πιστοποιητικό X.509. Χρησιμοποιείται για την εκπροσώπηση του χρήστη μέσω του Delegation Service (βλέπε βήμα 4, παράγραφο 4.4.1).
  - ***Derived Key Token Processor***: Ανάλογα με το είδος του processor (Kerberos Token, Token Reference Processor, κτλ.) εξάγει το κλειδί. Στην περίπτωση του Kerberos είναι το κλειδί της συνόδου που βρίσκεται είτε στο Kerberos Token είτε, εάν έχει δημιουργηθεί η σύνοδος, βρίσκεται από το Security Session HashTable.
  - ***Reference List Processor***: Ελέγχει όλες τις αναφορές στα διάφορα στοιχεία (XML elements) του μηνύματος SOAP, με βάση αυτές ανακτά τα αντίστοιχα κλειδιά κρυπτογράφησης και αποκρυπτογραφεί το μήνυμα.
  - ***Token Reference Processor***: Υποστηρίζει αναφορές σε Kerberos Tokens και με βάση αυτές βρίσκει το κλειδί συνόδου από το Security Session Hash Table.

Ο αλγόριθμος που υλοποιεί ο ACM Handler (**ACM Handler Logic**) συνοψίζεται ως εξής:

1. Αρχικά αποκωδικοποιεί (parses) το εισερχόμενο μήνυμα SOAP. Η διαδικασία αυτή επιστρέφει το μήνυμα SOAP ως ένα αντικείμενο Java.
2. Στην συνέχεια καλείται η ACM Security Engine για να επεξεργαστεί την επικεφαλίδα ασφαλείας του μηνύματος. Αυτή καλεί ανάλογα με τα στοιχεία του μηνύματος τους διάφορους processors. Εφόσον η επεξεργασία είναι επιτυχής, με το πέρας αυτού του βήματος έχει ταυτοποιηθεί και αποκρυπτογραφηθεί το μήνυμα.
3. Στην συνέχεια τροποποιεί το σώμα του αρχικού μηνύματος SOAP αντικαθιστώντας με το αποτέλεσμα που έλαβε από την ACM Security Engine. Για παράδειγμα, αντικαθιστά τα κρυπτογραφημένα στοιχεία με τα αντίστοιχα αποκρυπτογραφημένα.
4. Κατασκευάζει νέα επικεφαλίδα για το μήνυμα SOAP. Σε αυτή δεν περιέχεται ο αρχικός Security Header, αφού η επεξεργασία ασφαλείας έχει ολοκληρωθεί.
5. Ελέγχει αν έχει παρέλθει το έγκυρο χρονικό διάστημα σε περίπτωση που υπάρχει χρονοσήμανση (timestamp) στο μήνυμα SOAP.
6. Τέλος ελέγχει αν υπάρχει κανόνας πρόσβαση (message authorization) διαβάζοντας τον Access Rule Hash Table και επιτρέπει την πρόσβαση εφόσον βρεθεί ο αντίστοιχος κανόνας. Για περισσότερες λεπτομέρειες για την διαδικασία ελέγχου πρόσβασης δείτε την παράγραφο 4.4.3.

Σε αυτό το σημείο τελειώνει η παρουσίαση της υλοποίησης του ACMHandler.

5.4. Υλοποίηση του Kerberos KrbClient API

Το KrbClient API της αρχιτεκτονικής, ενθυλακώνεται σε μια κλάση Java, ονομαζόμενη krbClient, η οποία παρέχει και την υλοποίηση του. Μπορεί εύκολα να ενσωματωθεί σε οποιαδήποτε αρχιτεκτονική ασφαλείας client (π.χ. user portal ή αυτόνομη client εφαρμογή) σύμφωνα με τις ιδιομορφίες διαφόρων υλοποιήσεων client. Δεν υλοποιήσαμε handlers, όπως στην περίπτωση του server, δίνοντας έμφαση στην υλοποίηση του API, απαραίτητου για την ένταξη διαφόρων clients στην αρχιτεκτονική μας. Η πιθανή υλοποίηση handler θα πρέπει να είναι προσαρμοσμένη στα προγραμματιστικά περιβάλλοντα υλοποίησης clients, τα οποία ίσως να θέλουν ειδική διαχείριση, π.χ. clients βασισμένοι σε portlets [JCR168], τα οποία δεν

μοιράζονται state information μεταξύ τους. Πάντως κατάλληλοι handlers θα μπορούσαν να ενσωματωθούν στην προτεινόμενη αρχιτεκτονική client ώστε να διευκολυνθεί ο προγραμματιστής του client software στην χρήση της αρχιτεκτονικής

Οι ρυθμίσεις του krbClient, αποθηκεύονται στο clientConfig.properties. Τέτοιες ρυθμίσεις είναι το REALM/VO του χρήστη και το DNS name του KDC. Βασική δυνατότητα του krbClient είναι να διατηρεί την κατάσταση του χρήστη, λόγου χάρη τα εισιτήρια των υπηρεσιών (service tickets), το TGT κ.α.

Οι δημόσιοι μέθοδοι (public methods) του krbClient που επιτρέπουν την ταυτοποίηση (authentication) εξωτερικών χρηστών στον KDC (login) και την διαχείριση των διαπιστευτηρίων τους (tickets) είναι:

- init_krb: Εκτελεί το login ενός χρήστη με ένα username/password στο KDC και αποθηκεύει ένα TGT. Χρησιμοποιεί εξωτερικό kinit πρόγραμμα
- init_java: Εκτελεί το login ενός χρήστη με ένα username/password στο KDC και αποθηκεύει ένα TGT. Χρησιμοποιεί την kinit που είναι ενσωματωμένη στην Java και δεν υποστηρίζει το πρωτόκολλο PKINIT
- init_X509: Εκτελεί το login ενός χρήστη με ένα πιστοποιητικό X.509 και με το αντίστοιχο ιδιωτικό κλειδί και αποθηκεύει ένα TGT. Χρησιμοποιεί την init που παρέχει ο Heimdal Kerberos που υποστηρίζει το πρωτόκολλο PKINIT
- init_proxy: Εκτελεί το login ενός χρήστη με ένα πιστοποιητικό πληρεξουσίου X.509 (proxy certificate) και αποθηκεύει ένα TGT. Χρησιμοποιεί την init που παρέχει ο Heimdal Kerberos που υποστηρίζει το πρωτόκολλο PKINIT
- getTGTPath: Επιστρέφει τη θέση (directory path) του TGT στο σύστημα αρχείων (file system)
- destroy: Καταστρέφει το TGT και όλα τα service tickets
- setREALM: Εγγράφει την τιμή της μεταβλητή Kerberos REALM του χρήστη
- setKDC: Εγγράφει τη διεύθυνση (IP address ή DNS name) του ενεργού KDC που χρησιμοποιείται

- getDN: Επιστρέφει το Distinguished Name του χρήστη σε περίπτωση που χρησιμοποίησε πιστοποιητικό ή πιστοποιητικό πληρεξουσίου για να κάνει login στο KDC

Οι ακόλουθες μέθοδοι χρησιμοποιούνται για την απόκτηση εισιτηρίων υπηρεσιών (service tickets) και διαχείριση αυτών:

- requestServiceTicket: Δέχεται ως όρισμα το όνομα μιας υπηρεσίας και επιστρέφει ένα εισιτήριο (service ticket) για την χρήση της. Χρησιμοποιεί την υπηρεσία TGS του KDC
- renewServiceTicket: Δέχεται σαν όρισμα το όνομα μιας υπηρεσίας και ανανεώνει το εισιτήριο της (service ticket). Χρησιμοποιεί την υπηρεσία TGS του KDC
- getRequestedServices: Επιστρέφει όλα τα ονόματα των υπηρεσιών για τις οποίες έχουν εκδοθεί εισιτήρια (service tickets) και διατηρούνται μέσα στο αντικείμενο krbClient
- getServiceTicket: Επιστρέφει το εισιτήριο (service ticket) για ένα όνομα υπηρεσίας που είναι αποθηκευμένο στο αντικείμενο krbClient

Η κυρίως λειτουργία του API προσφέρεται από την παρακάτω μέθοδο:

- callKrbService: Δέχεται ως ορίσματα: (1) ένα αντικείμενο κλήσης (AXIS call object) προς μια υπηρεσία (έχει προηγηθεί η ανάκτηση εισιτηρίου της υπηρεσίας), (2) αν θα γίνει κρυπτογράφηση ή/και ηλεκτρονική υπογραφή στο σώμα του μηνύματος και (3) τα δεδομένα (data) κλήσης που θέλουμε να στείλουμε με το μήνυμα. Εναλλακτικά αντί των δεδομένων του μηνύματος (3), μπορεί να δεχτεί ένα XML Document ή έναν SOAP Envelope (δηλαδή ένα ολόκληρο SOAP μήνυμα). Όταν δέχεται ως παράμετρο τα data της κλήσης, παράγει εσωτερικά το μήνυμα SOAP που στέλνει. Στις άλλες περιπτώσεις δεν χρειάζεται.

Παρακάτω στον Πίνακα 4 ακολουθεί ένα απλό παράδειγμα χρήσης του KrbClient API.

Πίνακας 4: Απλό Παράδειγμα χρήσης του KrbClient API

```
public class testKrbAPI {
        boolean bodyEncryption=true, bodySigning=true;
```

```
        String userPrincipal= "username@REALM", servicePrincipal= "serviceName@REALM";
        Object[] callparam = { "this is the data"};

        public static void main(String[] args) {
                krbClient krbclient = new krbClient;
                krbclient.init_krb(userprincipal, password);
                krbclient.requestServiceTicket(servicePrincipal);
                testClient client = new testClient;
                Call call = client.callService();
                Object obj = krbclient.callKrbService(callparam, call, servicePrincipal,
bodyEncryption, bodySigning);
        }
}
```

5.5. Υλοποίηση του Policy Repository

Το Policy Repository έχει υλοποιηθεί ως ένα Web Service, το οποίο παράγεται από αντίστοιχο αρχείο WSDL. Εφόσον γνωρίζαμε εκ των προτέρων την πληροφορία που θέλουμε να αποθηκεύεται, ορίσαμε μέσα σε ένα WSDL file τις λειτουργίες που θέλαμε να παρέχει στην αρχιτεκτονική. Στην συνέχεια μέσω της WSDL2Java παράγαμε τον κώδικα Java stub για το Web Service Policy Repository. Ο κώδικας συμπληρώθηκε στα κατάλληλα σημεία με τις εντολές αποθήκευσης σε Βάση Δεδομένων MySQL [MYSQL].

Συγκεκριμένα ορίσαμε δύο λειτουργίες (Web Service Operations):

- getRules
- setRules

Ως παράμετρο λαμβάνουν μια υπηρεσία και επιστρέφουν μια λίστα από εγγραφές RuleTuple. Στον Πίνακα 5, παρουσιάζεται ο τύπος του RuleTuple, σε XML Schema.

Πίνακας 5: Ορισμός του RuleTuple σε XML Schema

```
<complexType name="RuleTuple">
    <sequence>
     <element name="command" nillable="true" type="xsd:string"/>
     <element name="instrumentManagerID" nillable="true" type="xsd:string"/>
     <element name="operation" nillable="true" type="xsd:string"/>
     <element name="parameterName" nillable="true" type="xsd:string"/>
     <element name="srv_princ" nillable="true" type="xsd:string"/>
     <element name="subgroup" nillable="true" type="xsd:string"/>
     <element name="username" nillable="true" type="xsd:string"/>
```

```
      <element name="vo" nillable="true" type="xsd:string"/>
    </sequence>
</complexType>
```

Οι τύπος RuleTuble περιγράφει το είδος της πληροφορίας που περιγράψαμε στην ενότητα 4.4.3, τροποποιημένο όμως, ώστε να καλύπτει τις ανάγκες ενός Instrument Element που περιγράψαμε στην ενότητα 2.3.

Η πληροφορία αυτή διατηρείται σε μια βάση δεδομένων. Χρησιμοποιήσαμε MySQL και δημιουργήσαμε έναν πίνακα με τα στοιχεία του RuleTuple. Η επικοινωνία με την βάση γίνεται με την χρήση της βιβλιοθήκης JDBC.

Η πρόσβαση στο Policy Repository ελέγχεται με ACM handler. Για το λόγο αυτό ορίσαμε το Policy Repository ως υπηρεσία στο KDC. Έτσι οι ACMs στο server για να αποθηκεύσουν τους τοπικούς τους κανόνες με ασφάλεια χρησιμοποιούν το KrbClient API, ζητώντας εισιτήριο (service ticket) για την υπηρεσία του Policy Repository και καλούν την λειτουργία setRules. Την ίδια διαδικασία ακολουθεί και το λογισμικό του πελάτη (client) για να καλέσει την λειτουργία getRules. Το Policy Repository διαθέτει τοπικούς κανόνες πρόσβασης, που επιτρέπουν ώστε οι λειτουργίες εγγραφής να επιτρέπονται μόνο από το τα ACMs, ενώ δικαιώματα ανάγνωσης παρέχονται σε όλους τους ταυτοποιημένους χρήστες.

6. Πειραματική Επαλήθευση και Αξιολόγηση της Προτεινόμενης Αρχιτεκτονικής Ασφαλείας

6.1. Εισαγωγή

Στα Κεφάλαια 4 και 5 της διατριβής παρουσιάσαμε την προτεινόμενη αρχιτεκτονική ασφαλείας, η οποία έχει σκοπό να βελτιώσει την απόδοση των λειτουργιών ασφαλείας ενός Instrumentation Grid. Η αρχιτεκτονική που σχεδιάσαμε και υλοποιήσαμε καλύπτει όλες τις βασικές αρχές του πλαισίου OGSA (ενότητα 4.5) και μπορεί εύκολα να συνεργαστεί με οποιαδήποτε Computational Grid ακολουθεί αυτές τις αρχές.

Στο παρόν κεφάλαιο θα επαληθεύσουμε πειραματικά την αρχιτεκτονική εξομοιώνοντας τη προστασία μιας υπηρεσίας (ενός Web Service). **Συγκεκριμένα θα μετρήσουμε την απόδοση του ACM handler (ενότητα 5.3) που ακολουθεί το Kerberos Token Profile στην επεξεργασία ασφαλών μηνυμάτων και θα το συγκρίνουμε με αυτή του WSS4J handler – X.509 Token Profile.** Να σημειώσουμε πως σε ένα Instrumentation Grid η στενωπός της αρχιτεκτονικής είναι το Instrument Element σαν υπηρεσία που μπορεί να αντιπροσωπεύει πολλά όργανα-συσκευές και να εξυπηρετεί πολλούς χρήστες.

Στην συνέχεια του κεφαλαίου θα δούμε το δοκιμαστικό περιβάλλον (testbed) που δημιουργήσαμε, τα σενάρια που χρησιμοποιήσαμε, την μεθοδολογία που ακολουθήσαμε και τα αποτελέσματα που εξάγαμε.

6.2. Δοκιμαστικό Περιβάλλον και Σενάρια

Δημιουργήσαμε ένα δοκιμαστικό περιβάλλον για να μπορέσουμε να μετρήσουμε και να εκτιμήσουμε την απόδοση της προτεινόμενης αρχιτεκτονικής ασφαλείας και συγκεκριμένα των μηχανισμών μεταφοράς προστατευμένων μηνυμάτων. Οι διαδικασία login (μέσω του pkinit) στο KDC δεν μετρήθηκε, γιατί δεν παίζει κανένα ρόλο κατά την αλληλεπίδραση του πελάτη με την υπηρεσία. Συμβαίνει συνήθως προκαταβολικά της συνόδου ελέγχου (π.χ. αν το λογισμικό του πελάτη είναι ένα portal κατά την είσοδο του στο portal και όχι κατά την αλληλεπίδραση του με την υπηρεσία).

Στο δοκιμαστικό περιβάλλον συγκρίναμε την απόδοση της υλοποίησης **ACM Kerberos Token Profile handler**, με την υλοποίηση του **WSS4J X.509 Token Profile handler**. Η δεύτερη υλοποίηση εκπροσωπεί την απόδοση μιας εναλλακτικής λύσης που θα έβρισκε χρήση σε ένα Instrumentation Grid βασισμένο στο GSI. Ισχύει πάντα η προϋπόθεση ότι χρησιμοποιείται ασφάλεια στο επίπεδο του μηνύματος (SOAP) και όχι στο επίπεδο της μεταφοράς (http).

6.2.1. Δοκιμαστικό Περιβάλλον

Το δοκιμαστικό περιβάλλον που δημιουργήσαμε αποτελείται από ένα εξυπηρετητή, όπου τρέχει μια απλή υπηρεσία "echo" Web Service και οι ACM/WSS4J handlers με 16 πελάτες (χρήστες) στο ίδιο τοπικό δίκτυο. Ο εξυπηρετητής έχει έναν διπύρηνο AMD64 X2 4.400 (2200MHz), 2GB φυσικής μνήμης, 2x120GB σκληρούς δίσκους σε διάταξη Raid 1. Το λειτουργικό σύστημα είναι Linux Debian 4.0 και χρησιμοποιεί τον πυρήνα συμμετρικής πολυεπεξεργασίας (SMP kernel) 64 bit.

Το λογισμικό του πελάτη έχει εγκατασταθεί σε ένα μικτό περιβάλλον από 16 υπολογιστές με Windows XP ή Linux. Υλοποιήθηκε σε περιβάλλον Java 1.5. Οι πελάτες έχουν την δυνατότητα να στέλνουν μηνύματα echo μέσω πρωτοκόλλου SOAP με καθορισμένο ρυθμό αποστολής και μέγεθος μηνύματος. Επίσης έχουν τις δυνατότητες να κρυπτογραφούν το σώμα του μηνύματος SOAP χρησιμοποιώντας τον προτυποποιημένο αλγόριθμο κρυπτογράφησης AES128 [Daem02] και να προσθέτουν ηλεκτρονική υπογραφή του σώματος του μηνύματος στην επικεφαλίδα ασφαλείας του. Χρησιμοποιούν φυσικά το KrbClient API που είδαμε στις ενότητες 4.4.2.1 και 5.4.

Το λογισμικό της υπηρεσίας echo Web Service που χρησιμοποιούμε για να δοκιμάσουμε την αρχιτεκτονική υλοποιείται με Apache Axis 1.3 [AXIS], εκτελείται σε ένα Java 1.5 64bit Edition Java Virtual Machine (JVM) και είναι εγκατεστημένο σε Jakarta Tomcat 5.28 Servlet Server.

Ο κώδικας του ACM, που χρησιμοποιούμε, έχει υλοποιηθεί ως Axis Handler, όπως είδαμε στην ενότητα 5.3. Οι Handlers μπορούν να εγκατασταθούν κατά την επεξεργασία ενός μηνύματος SOAP στον εξυπηρετητή πριν ή μετά από το ίδιο Web Service, τροποποιώντας το μήνυμα SOAP κατά περίπτωση. Ο ACM χειρίζεται ό,τι έχει σχέση με την ταυτοποίηση και τον έλεγχο πρόσβασης σε κάθε μήνυμα SOAP

σύμφωνα με την αρχιτεκτονική μας. Υλοποιεί για την μεταφορά των μηνυμάτων το WSS Kerberos Token Profile και χρησιμοποιεί την βιβλιοθήκη Apache WSS4J. Για την ακρίβεια επεκτείνει τον υπάρχον WSS4J Handler που παρέχει η βιβλιοθήκη.

Η σύγκριση γίνεται με τον παρεχόμενο handler από την βιβλιοθήκη WSS4J , ο οποίος υλοποιεί το WSS X.509 Token Profile. Για τις κρυπτογραφικές λειτουργίες χρησιμοποιήσαμε και στις δύο υλοποιήσεις την ίδια βιβλιοθήκη κρυπτογράφησης την BouncyCastle JCE [BONCA], που υλοποιεί το πρότυπο Java Cryptography Extension (JCE) [JCE02],. Η χρήση των ίδιων βιβλιοθηκών μας εξασφαλίζει ότι θα έχουν την ίδια επίδοση όσον αφορά τις κρυπτογραφικές τους λειτουργίες.

6.2.2. Σενάρια Μετρήσεων

Τα σενάρια που θα παρουσιάσουμε έχουν σκοπό να μετρήσουν την απόδοση της επεξεργασίας των λειτουργιών ασφαλείας στην πλευρά της υπηρεσίας χρησιμοποιώντας **ασφάλεια στο επίπεδο μηνύματος**, τόσο υπό χαμηλό όσο και υπό υψηλό υπολογιστικό φορτίο (άνω του 90%). Ως ασφάλεια στο επίπεδο του μηνύματος ορίζουμε τις εξής λειτουργίες:

1. **Ταυτοποίηση Μηνύματος (Message Authentication)**
2. **Ακεραιότητα Μηνύματος (Message Integrity)**
3. **Εμπιστευτικότητα Μηνύματος (Message Confidentiality)**

Η ταυτοποίηση του μηνύματος αφορά στην επιβεβαίωση της ταυτότητας του αποστολέα. Μαζί με τον έλεγχο ακεραιότητας των μηνυμάτων παρέχεται μέσω ηλεκτρονικών υπογραφών ή κωδικών HMAC. Η εμπιστευτικότητα του μηνύματος διασφαλίζεται μέσω της κρυπτογράφησης του περιεχομένου.

Η απόδοση εκτιμήθηκε μετρώντας το μέσο ρυθμό μηνυμάτων (SOAP messages/sec), τον ρυθμό μετάδοσης πληροφορίας (message payload) και των επιμέρους χρόνων επεξεργασίας των λειτουργιών ασφαλείας. Οι χρόνοι αυτοί ορίζονται σε επόμενη ενότητα 6.3. Είναι σημαντικό να μετρηθούν/ποσοτικοποιηθούν οι σχετικές διαφορές στην απόδοση μεταξύ των λειτουργιών της ηλεκτρονικής υπογραφής και κρυπτογράφησης, αν και αναμένουμε ίδια συμπεριφορά αφού βασίζονται στις ίδιες κρυπτογραφικές αρχές. Τέτοια αποτελέσματα θα ήταν πολύ χρήσιμα στο να αποφασιστεί ποια μέθοδος (Πιστοποίηση – Ακεραιότητα – Εμπιστευτικότητα) θα μπορούσε να χρησιμοποιηθεί στις διάφορες εφαρμογές

Instrumentation Grid ώστε σύμφωνα με τις απαιτήσεις τους να βρεθεί η χρυσή τομή ανάμεσα σε απόδοση και ασφάλεια.

Τα σενάρια που θα εξετάσουμε, τα χωρίζουμε σε δύο κύριες κατηγορίες:

A) **Υπό αυξανόμενο αριθμό πελατών (που αναλογεί σε αναλογικά αυξανόμενο υπολογιστικό φορτίο του echo Web Service)**

   a. **SC1**[2]: Μελέτη απόδοσης των λειτουργιών ασφαλείας, προσφέροντας μόνο **Εμπιστευτικότητα Μηνύματος (μόνο κρυπτογράφηση)**, αυξάνοντας τον αριθμό πελατών και κρατώντας σταθερό το μέγεθος των μηνυμάτων και τον ρυθμό τους ανά πελάτη

B) Υπό αυξανόμενο μέγεθος μηνυμάτων SOAP σε υψηλό συνολικό φορτίο echo Web Service από σταθερό αριθμό πελατών

   a. **SC2**: Μελέτη απόδοσης λειτουργιών ασφαλείας, προσφέροντας **Ταυτοποίηση, Ακεραιότητα και Εμπιστευτικότητα Μηνύματος (κρυπτογράφηση και ηλεκτρονικές υπογραφές)** για αυξανόμενα μεγέθη μηνύματος.

   b. **SC3**: Μελέτη απόδοσης λειτουργιών ασφαλείας, προσφέροντας **Ταυτοποίηση και Ακεραιότητα Μηνύματος (μόνο ηλεκτρονικές υπογραφές ή HMAC)** για αυξανόμενα μεγέθη μηνύματος

Το υπολογιστικό φορτίο που αναφέρουμε δεν προκαλείται από εξωγενείς παράγοντες, αλλά από την ίδια την λειτουργία της υπηρεσίας. Αυξάνοντας το ρυθμό ή/και τους πελάτες που επικοινωνούν με την υπηρεσία, αυξάνεται το υπολογιστικό φορτίο. **Η Υπηρεσία για τα πειράματα μας έπρεπε να προκαλεί μικρό σταθερό υπολογιστικό φορτίο, ώστε να έχουμε καθαρή εικόνα για την απόδοση των δύο υπό σύγκριση μηχανισμών. Έτσι επιλέξαμε την πιο απλή υπηρεσία που είναι ένα "echo" Web Service, δηλαδή μια υπηρεσία που απλά επιστρέφει στον πελάτη το μήνυμα που έλαβε**.

Επίσης, δεν έχουμε συμπεριλάβει στα σενάρια μας την λειτουργία ελέγχου πρόσβασης (authorization) που προσφέρει ο ACM, αφού ουσιαστικά θα ήταν κοινός

[2] Οι μετρήσεις στο SC1 έγιναν χωρίς να είναι ενεργοποιημένος ο SMP kernel, για αυτό παρουσιάζουν μειωμένη απόδοση σε σχέση με τα SC2 και SC3. Πάντως, η χρήση του SMP kernel δεν επηρεάζει την συγκριτική επίδοση των δύο handlers που είναι και το ζητούμενο

παράγοντας στις συγκρίσεις. Ο χρόνος για να ανακληθεί η πολιτική από την μνήμη RAM (στην υλοποίηση μας) σε κάθε σενάριο θα είναι ο ίδιος και επομένως ο έλεγχος πρόσβασης θα επηρέαζε την απόδοση κατά ένα σταθερό παράγοντα. Εξάλλου ο βασικός μας σκοπός είναι να δώσουμε έμφαση στην απόδοση των ασφαλών ανταλλαγών μηνυμάτων, αφού επιβαρύνουν την απόδοση της επεξεργασίας κατά σημαντικό βαθμό. Μια εκτίμηση της διαφοράς των κρυπτογραφικών αλγορίθμων κρυπτογράφησης συμμετρικού και δημοσίου κλειδιού είδαμε στην παράγραφο 3.3.2. Ωστόσο αναφέρονται καθαρά στους αλγόριθμους και δεν αντικατοπτρίζουν τις διαφορές σε ένα κατανεμημένο περιβάλλον ανταλλαγής μηνυμάτων.

Λεπτομερής περιγραφή του σεναρίων παρουσιάζονται στον Πίνακα 6.

Πίνακας 6: Μεθοδολογία σεναρίων υπό μεταβαλλόμενο υπολογιστικό φορτίο

Σενάριο	Περιγραφή
SC1 **(υπό αυξανόμενο αριθμό πελατών, σταθερό ρυθμό αποστολής ανά πελάτη, αυξανόμενο υπολογιστικό φορτίο, σταθερό μέγεθος μηνύματος)**	Στο SC1, οι πελάτες στέλνουν μηνύματα σταθερού μεγέθους με σταθερό ρυθμό (21 μηνύματα/sec). Το μέγεθος του μηνύματος έχει ορισθεί σε 60 bytes. Το σώμα του μηνύματος (message payload) κρυπτογραφείται. Εκτελούμε τις μετρήσεις, αρχικά με ένα πελάτη και σε κάθε επανάληψη προσθέτουμε ένα πελάτη, μέχρι να έχουμε 6 πελάτες να στέλνουν ταυτόχρονα. Ο αριθμός των μηνυμάτων που στέλνουν είναι 10.000 ο καθένας και ξεκινούν ταυτόχρονα.
SC2, SC3 **(αυξανόμενο μέγεθος μηνύματος, υπό σταθερό αριθμό πελατών, μέγιστος ρυθμός αποστολής μηνυμάτων από του πελάτες, υψηλό υπολογιστικό φορτίο)**	Η μεθοδολογία που ακολουθήσαμε στα σενάρια SC2 και SC3, αφορά 16 πελάτες που στέλνουν με τον μέγιστο ρυθμό τους. Κάθε πελάτης στέλνει συνολικά 10.000 μηνύματα (συνολικά 160.000). Όλοι οι πελάτες ξεκινούν την αποστολή ταυτόχρονα. Οι μετρήσεις επαναλαμβάνονται με μέγεθος μηνύματος 60, 400, 800, 1600 και 3200.

Σε σενάρια απομακρυσμένου ελέγχου ενός Instrumentation Grid που στοχεύει η αρχιτεκτονική μας, έχουμε γενικά μικρά μηνύματα ελέγχου (<1000 bytes), επομένως δεν υπάρχει κατάτμηση σε πακέτα IP. Ωστόσο, συμπεριλάβαμε στις μετρήσεις και μεγαλύτερα μεγέθη (π.χ. 1600 και 3200 bytes), τα οποία υπόκεινται σε κατάτμηση. Με αυτό τον τρόπο θα μπορούσαμε να παρατηρούμε αν υπάρχει επίπτωση της κατάτμηση πακέτων στην χρονική καθυστέρηση και να δούμε την εν γένει συμπεριφορά καθώς αυξάνει το μέγεθος των μηνυμάτων. Επισημάνουμε ότι αναφερόμαστε το μέγεθος του μηνύματος πριν την κρυπτογράφηση και όχι στο μέγεθος του μηνύματος SOAP που είναι σαφώς μεγαλύτερο, διότι συμπεριλαμβάνει την επικεφαλίδα του μηνύματος (συμπεριλαμβανομένης και της επικεφαλίδας ασφαλείας).

Για να εκτιμήσουμε την απόδοση της υπηρεσίας echo Web Service που προστατεύεται από την αρχιτεκτονική μας, μετράμε κατά την διάρκεια του πειράματος τον αριθμό των μηνυμάτων SOAP και τους αντίστοιχους υπολογιστικούς χρόνους για κάθε handler (βλέπε 6.2.1) που υλοποιεί το κάθε πρωτόκολλο (Kerberos Token Profile και Certificate Token Profile). Μόνο η κατεύθυνσης λήψης του εξυπηρετητή προστατεύεται καθώς μας ενδιαφέρει να δούμε την απόδοση σε αυτό το σημείο και όχι στους πελάτες.

Ακολουθεί στην συνέχεια η παράθεση των λεπτομερειών ανά μηχανισμό που υιοθετήθηκαν σε κάθε σενάριο.

6.2.3. X.509 Certificates

Στο **σενάριο SC1**, όπου προσφέρεται Εμπιστευτικότητα Μηνύματος (κρυπτογραφώντας το σώμα του μηνύματος SOAP) αυξάνοντας τον αριθμό πελατών, σε κάθε αποστολή μηνύματος, ο client:

α) Κρυπτογραφεί το σώμα του μηνύματος SOAP με τυχαία επιλογή συμμετρικού κλειδιού ανά μήνυμα (χρησιμοποιώντας τον αλγόριθμο AES128),

β) Δημιουργεί επικεφαλίδα WS Security, κρυπτογραφεί το συμμετρικό κλειδί με τον αλγόριθμο κρυπτογράφησης δημοσίου κλειδιού RSA15, χρησιμοποιώντας το δημόσιο κλειδί του WSS4J X.509 Token Profile handler της υπηρεσίας,

γ) Προσθέτει στην κεφαλίδα WS Security το κρυπτογραφημένο συμμετρικό κλειδί μαζί με την αναφορά στον εκδότη και στο σειριακό αριθμό του πιστοποιητικού X.509 που χρησιμοποιήθηκε κατά την κρυπτογράφηση του κλειδιού

δ) Αντικαθιστά στο σώμα του μηνύματος το αρχικό κείμενο (cleartext) με το κρυπτογραφημένο (ciphertext).

Στην πλευρά του server, ο WSS4J X.509 Token Profile handler:

α) Ανακτά από την τοπική αποθήκη κλειδιών το ιδιωτικό κλειδί της υπηρεσίας και με βάση αυτό

β) Αποκρυπτογραφεί το συμμετρικό κλειδί και

γ) Αποκρυπτογραφεί με το συμμετρικό κλειδί το σώμα του μηνύματος. Τέλος αντικαθιστά το κρυπτογραφημένο σώμα με το αποκρυπτογραφημένο, αφαιρεί την επικεφαλίδα ασφαλείας και το προωθεί στην υπηρεσία.

Στο **σενάριο SC2**, το οποίο προσφέρει Ταυτοποίηση, Ακεραιότητα και Εμπιστευτικότητα Μηνύματος (μέσω κρυπτογράφησης και υπογραφής του σώματος του μηνύματος) για αυξανόμενα μεγέθη μηνύματος, κάθε πελάτης:

α) Αρχικά κρυπτογραφεί με ένα τυχαίο συμμετρικό κλειδί το σώμα του μηνύματος SOAP χρησιμοποιώντας τον αλγόριθμο κρυπτογράφησης AES128, υλοποιώντας το πρότυπο XML-Enc που περιγράφεται στην παράγραφο 3.5.3,

β) Υπογράφει το σώμα μήνυμα ακολουθώντας το πρότυπο XML Digital Signature (δες την παράγραφο 3.5.4). Το τυχαίο συμμετρικό κλειδί κρυπτογράφησης μεταφέρεται μέσα στην επικεφαλίδα ασφαλείας μηνύματος κρυπτογραφημένο με τον αλγόριθμο δημοσίου κλειδιού RSA15 με την χρήση του δημοσίου κλειδιού της υπηρεσίας. Το μήνυμα σε αυτήν την περίπτωση αποτελείται από το κρυπτογραφημένο σώμα και που περιέχει μια επικεφαλίδα WS Security, η οποία περιέχει μια αναφορά στο εκδότη και στο σειριακό αριθμό του πιστοποιητικού X.509 της υπηρεσίας. Και τέλος

γ) Προσθέτει μια ηλεκτρονική υπογραφή στην επικεφαλίδα ασφαλείας μαζί με την αντίστοιχη αναφορά στον εκδότη και στο σειριακό αριθμό του πιστοποιητικού X.509 του πελάτη που χρησιμοποιήθηκε για την υπογραφή. Η

κρυπτογράφηση προσφέρει Εμπιστευτικότητα, ενώ η ψηφιακή υπογραφή Ταυτοποίηση και Ακεραιότητα.

Στην πλευρά του εξυπηρετητή, στο σενάριο SC2, όταν λαμβάνεται ένα μήνυμα, ο WSS4J handler:

α) Ανακτά το ιδιωτικό κλειδί της υπηρεσίας και το δημόσιο κλειδί του πελάτη από την τοπική αποθήκη κλειδιών Υποδομής Δημοσίου Κλειδιού -ΥΔΚ (PKI keystore). Στην ΥΔΚ αποθηκεύονται όλα τα κλειδιά της υπηρεσίας αλλά και των πελατών που είναι εξουσιοδοτημένοι να έχουν πρόσβαση στην υπηρεσία. Επόμενο βήμα, αφού ανακτήσει τα κλειδιά,

β) Χρησιμοποιεί το ιδιωτικό κλειδί του για να αποκρυπτογραφήσει το συνημμένο συμμετρικό κλειδί στο μήνυμα,

γ) Επαληθεύει την ηλεκτρονική υπογραφή με το δημόσιο κλειδί του πελάτη, επομένως ταυτοποιείται ο μήνυμα και τέλος

δ) Αποκρυπτογραφεί με το συμμετρικό κλειδί το σώμα του μηνύματος. Το μήνυμα μεταβιβάζεται αποκρυπτογραφημένο και απογυμνωμένο από όλη την πληροφορία ασφαλείας (κεφαλίδα ασφαλείας, ηλεκτρονικές υπογραφές κτλ) στην τελική υπηρεσία που το εξυπηρετεί.

Όσον αφορά το **σενάριο SC3**, όπου προσφέρεται Ταυτοποίηση και Ακεραιότητα Μηνύματος για αυξανόμενα μεγέθη μηνύματος, ο handler ασφαλείας του πελάτη:

α) Κρυπτογραφεί το σώμα του SOAP μηνύματος,

β) Δημιουργεί μια επικεφαλίδα WS Security όπου προσθέτει την υπογραφή με αναφορά στον εκδότη και στον σειριακό αριθμό του X.509 πιστοποιητικού του πελάτη. Και τέλος

γ) Όταν το μήνυμα ληφθεί στον εξυπηρετητή, ο WSS4J handler της υπηρεσίας ανακτά από το δημόσιο κλειδί του πελάτη από την τοπική αποθήκη κλειδιών της ΥΔΚ και εγκρίνει ή απορρίπτει το μήνυμα ανάλογα με το αποτέλεσμα ελέγχου της ηλεκτρονικής υπογραφής.

6.2.4. Kerberos Tokens

Στην περίπτωση του Kerberos Token Profile μετά από αρχική ταυτοποίησης ενός χρήστη χρησιμοποιούμε αποκλειστικά συμμετρική κρυπτογραφίας και στα τρία σενάρια. Το σενάριο SC3 που απαιτεί και ηλεκτρονικές υπογραφές ανά μήνυμα που υλοποιούνται με την χρήση κωδικών **HMAC (Hashed Message Authentication Code)** που περιγράψαμε στην ενότητα 3.3.1**.** Οι κώδικες HMAC στην περίπτωσή μας παράγονται χρησιμοποιώντας τον αλγόριθμο κατακερματισμού SHA1 στο σώμα του μηνύματος SOAP και κρυπτογραφώντας το αποτέλεσμα με τον συμμετρικό αλγόριθμο AES 128. Στο σενάριο SC2 λόγω του γεγονότος ότι στο πρωτόκολλο Kerberos ο πελάτης και ο εξυπηρετητής μοιράζονται κοινό κλειδί και περιμένουμε σε Instrumentation Grids δεδομένα καθορισμένης μορφής (π.χ. μια εντολή) ο HMAC παραλείπεται (βλέπε παράγραφο 4.4.2.3).

Και στα τρία σενάρια στην περίπτωση που χρησιμοποιείται ο ACM handler, όταν λαμβάνει το πρώτο μήνυμα SOAP, δημιουργείται μια σύνοδος στο επίπεδο του WS Security. Η σύνοδος ορίζεται από το session key που βρίσκεται στο εισιτήριο της υπηρεσίας (service ticket) που στέλνεται στο πρώτο μήνυμα και διαρκεί όσο διαρκεί αυτό (περισσότερα είδαμε στην παράγραφο 4.3). Στα επόμενα μηνύματα της συνόδου γίνεται απλή αναφορά του session key μέσα στην επικεφαλίδα των μηνυμάτων.

Το λογισμικό του πελάτη και στα τρία σενάρια SC1, SC2 και SC3 καλεί σε επαναληπτικό βρόχο την υπηρεσία echo Web Service χρησιμοποιώντας το KrbClient API (βλέπε ενότητα 5.4), το οποίο αποτελεί αναπόσπαστο μέρος της προτεινόμενης αρχιτεκτονικής. Η διαδικασία ακολουθεί τα εξής βήματα:

α) Ο client αρχικοποιεί ένα αντικείμενο krbClient, παρέχοντας του το πιστοποιητικό X.509 του πελάτη,

β) Με βάση το πιστοποιητικό του πελάτη, το αντικείμενο krbClient ταυτοποιεί τον πελάτη στο Kerberos KDC και ως αποτέλεσμα λαμβάνει ένα TGT όπως είδαμε στην παράγραφο 4.4.2.1. Υλοποιείται με αυτό τον τρόπο η λειτουργία Single Sign On - SSO,

γ) Στην συνέχεια ο krbClient εξετάζει το URL της υπηρεσίας και ανακτά το εισιτήριο της υπηρεσίας (service ticket), χρησιμοποιώντας το TGT και καλώντας την υπηρεσία TGS του Kerberos KDC.

δ) Ο krbClient δημιουργεί επικεφαλίδα ασφαλείας WS Security μέσα στο μήνυμα SOAP και προσθέτει το εισιτήριο της υπηρεσίας echo Web Service. Μέσα στην επικεφαλίδα ασφαλείας του μηνύματος SOAP, το εισιτήριο ονομάζεται Kerberos Token. Τέλος,

ε) Ο krbClient χρησιμοποιεί το κλειδί της συνόδου (session key) που επιστρέφει η υπηρεσία TGS μαζί με το εισιτήριο της υπηρεσίας για να κρυπτογραφήσει ή να υπογράψει τα SOAP μηνύματα στα τρία σενάρια (SC1, SC2 και SC3).

Ειδικότερα, στο **σενάριο SC1**, το αντικείμενο krbClient κρυπτογραφεί το σώμα του μηνύματος με την χρήση του κλειδιού της συνόδου που λαμβάνει μαζί με την λήψη του εισιτηρίου της υπηρεσίας. Το ίδιο κλειδί εμπεριέχεται και μέσα στο εισιτήριο της υπηρεσίας. Υπενθυμίζουμε (βλέπε παράγραφο 3.4.2) πως το εισιτήριο της υπηρεσίας (service ticket) είναι μια δομή, που περιλαμβάνει μεταξύ των άλλων (πελάτης, διάρκεια ζωής του εισιτήριου κτλ.) το κλειδί συνόδου. Το εισιτήριο της υπηρεσίας είναι κρυπτογραφημένο με το συμμετρικό κλειδί της υπηρεσίας echo, επομένως μόνο η ίδια η υπηρεσία μπορεί να το διαβάσει. Η ενσωμάτωση του εισιτηρίου μέσα στην επικεφαλίδα ασφαλείας γίνεται μόνο για το πρώτο μήνυμα. Στα επόμενα μηνύματα και για την διάρκεια ζωής του εισιτηρίου στέλνεται μια αναφορά στο εισιτήριο της υπηρεσίας, όπως ορίζεται στο πρωτόκολλο WS Security Kerberos Token Profile. Η τιμή της αναφοράς προκύπτει από την χρήση της συνάρτησης κατακερματισμού (hashing function) SHA1 πάνω στο εισιτήριο της υπηρεσίας.

Στην πλευρά του server, ο ACM handler:

α) Αρχικά εξάγει το εισιτήριο από το πρώτο μήνυμα του πελάτη και το αποθηκεύει τοπικά στον πίνακα κατακερματισμού Session Hash Table (δες παράγραφο 5.3), χρησιμοποιώντας ως κλειδί στον πίνακα κατακερματισμού την τιμή κατάτμησης (hash value) του εισιτηρίου,

β) Στην συνέχεια και για το πρώτο μήνυμα με το ενσωματωμένο στο εισιτήριο κλειδί συνόδου αποκρυπτογραφεί το σώμα του μηνύματος. Για τα επόμενα μηνύματα, το κλειδί συνόδου δεν στέλνεται μέσα στο εισιτήριο, αλλά στέλνεται μια αναφορά σε αυτό. Η αναφορά βρίσκεται στην επικεφαλίδα ασφαλείας του μηνύματος SOAP και παραπέμπει στο τοπικά αποθηκευμένο κλειδί συνόδου. Τέλος,

γ) Ο ACM handler αντικαθιστά το κρυπτογραφημένο σώμα του μηνύματος με την αποκρυπτογραφημένη τιμή του, αφαιρεί την κεφαλίδα ασφαλείας και προωθεί το μήνυμα στην τελική υπηρεσία. Η κρυπτογράφηση ακολουθεί το πρότυπο XML-Enc που είδαμε στην παράγραφο 3.5.3.

Στο **σενάριο SC2**, το αντικείμενο KrbClient, κατά την αποστολή των μηνυμάτων, κρυπτογραφεί και υπογράφει το σώμα του μηνύματος SOAP με το κλειδί συνόδου που λαμβάνει κατά την ανάκτηση του εισιτηρίου της υπηρεσίας. Ως αλγόριθμο κρυπτογράφησης χρησιμοποιεί τον AES128 και για υπογραφές προσθέτει ένα Hashed Message Authentication Code – HMAC (δες παράγραφο 3.3.1). Όπως και στο SC1 στο πρώτο μήνυμα προσθέτει το εισιτήριο της υπηρεσίας στην επικεφαλίδα ασφαλείας, ενώ στα επόμενα μηνύματα, όπως περιγράψαμε και στην προηγούμενη παράγραφο, στέλνει αναφορά προς το εισιτήριο την τιμή κατακερματισμού του εισιτηρίου. Επιπλέον στις επικεφαλίδες όλων των μηνυμάτων προστίθεται ο κώδικας HMAC για ταυτοποίηση και έλεγχο ακεραιότητάς τους.

Στο **σενάριο SC3**, ο πελάτης προσθέτει μόνο την ηλεκτρονική υπογραφή. Η επεξεργασία του μηνύματος είναι πανομοιότυπη του σεναρίου SC2, αφαιρώντας την λειτουργία της κρυπτογράφησης του σώματος του μηνύματος.

6.3. Παράμετροι Αξιολόγησης

Τα σενάρια και η μεθοδολογία που περιγράψαμε παραπάνω έχουν ως σκοπό την συγκριτική μελέτη των δύο μηχανισμών (handlers) . Η περίπτωση της χρήσης του ACM handler αντικατοπτρίζει την απόδοση της προτεινόμενης αρχιτεκτονικής κατά τον έλεγχο των συσκευών/οργάνων σε ένα Instrumentation Grid, ενώ η περίπτωση του WSS4J handler, που χρησιμοποιεί ηλεκτρονικά πιστοποιητικά X.509 αντιστοιχεί στην χρήση κρυπτογραφίας δημοσίου κλειδιού που θα χρησιμοποιούσαμε σε ένα Computational Grid βασισμένο σε GSI για τον έλεγχο των συσκευών/οργάνων, αν δεν παρέμβαινε η αρχιτεκτονική ασφαλείας που προτείνεται στην διατριβής.

Στις μετρήσεις τροποποιήσαμε ελαφρώς τους handlers, ώστε να καταγράφονται οι χρόνοι έναρξης και λήξης επεξεργασίας των διαφόρων λειτουργιών του handler ώστε. Οι λειτουργίες που καταγράψαμε για τους δύο handlers (ACM και WSS4J) είναι:

- **SOAP Processing**: Αξιολογείται με τον χρόνο που χρειάζεται ένας WSS handler για να λάβει το μήνυμα SOAP από τον HTTP handler και να το

αποκωδικοποιήσει (de-serialize). Η λειτουργία της αποκωδικοποίησης περιλαμβάνει την επεξεργασία των δεδομένων XML και την μετατροπή των ακολουθιών χαρακτήρων (strings) σε δομές της γλώσσας προγραμματισμού που χρησιμοποιείται. Στην περίπτωση μας παράγονται Java objects. Αποτελεί σημαντική στενωπό απόδοσης όπως έχει δειχτεί στην σχετική βιβλιογραφία [Chiu02], [Ghaz05].

- **Security Request Processing**: Αξιολογείται με τον χρόνο που απαιτείται για να επεξεργαστεί ο handler όλα τα security tokens (π.χ. Kerberos Ticket, X.509 Certificate), να εξάγει από αυτά τα κλειδιά κρυπτογράφησης, να επαληθεύσουν τις ηλεκτρονικές υπογραφές και να αποκρυπτογραφήσουν το μήνυμα (εφόσον είναι κρυπτογραφημένο)
- **Request to Axis**: Αξιολογείται με τον χρόνο που απαιτείται για να ξαναδημιουργηθεί το μήνυμα, αφαιρώντας όλα τα στοιχεία ασφάλειας και (εφόσον το αρχικό μήνυμα ήταν κρυπτογραφημένο) να αντικατασταθεί το κρυπτογραφημένο μήνυμα (ciphertext) με το αποκρυπτογραφημένο (cleartext)
- **Verify header, cert, timestamp**: Αξιολογείται με τον χρόνο που απαιτείται για την τελική επαλήθευση της επικεφαλίδας, του πιστοποιητικού και της χρονοσήμανσης (timestamp) εάν υπάρχει
- **Total Handler Processing**: Αξιολογείται με τον συνολικό χρόνο επεξεργασίας του WS Security που πραγματοποιείται σε κάθε handler. Αυτός ο χρόνος είναι το άθροισμα των χρόνων.

Εκτός από τους παραπάνω χρόνους, για να μπορέσουμε να αποκτήσουμε μια καλύτερη κατανόηση πως ο κάθε μηχανισμός αποδίδει, μετράμε την **Μέση Ρυθμαπόδοση (Average_Throughput)** των μηνυμάτων SOAP (messages/sec) που εξυπηρετεί η Υπηρεσία καθόλη την διάρκεια του πειράματος. Το μέγεθος αυτό είναι σημαντικό για εφαρμογές ευαίσθητες σε καθυστερήσεις όπως αυτές του Instrumentation Grid: Οι συνόδοι (sessions) που δημιουργούνται μεταξύ των πελατών και της υπηρεσίας, συνήθως, απαιτούν ένα μεγάλο αριθμό μικρών μηνυμάτων με σκοπό να μειώσουν τον μέσο χρόνο απόκρισης (average latency) (latency) και την διασπορά της καθυστέρησης (delay jitter) μιας υπό ελέγχου συσκευής. Η Μέση Ρυθμαπόδοση υπολογίζεται μετρώντας τον αριθμό των εισερχομένων μηνυμάτων

SOAP και διαιρώντας τον με τον χρόνο που απαιτήθηκε για να ολοκληρωθεί το πείραμα.

Βασιζόμενοι στη Μέση Ρυθμαπόδοση, υπολογίζουμε την **Ενεργή Ρυθμαπόδοση (Effective Throughput)**, η οποία ορίζεται ως:

Effective Throughput = (Average_Throughput)*(SOAP_Body_Size)

Όπου **SOAP_Body_Size** αναφέρεται στο μέγεθος της πληροφορίας που μεταδίδεται πριν την κρυπτογράφησή της.

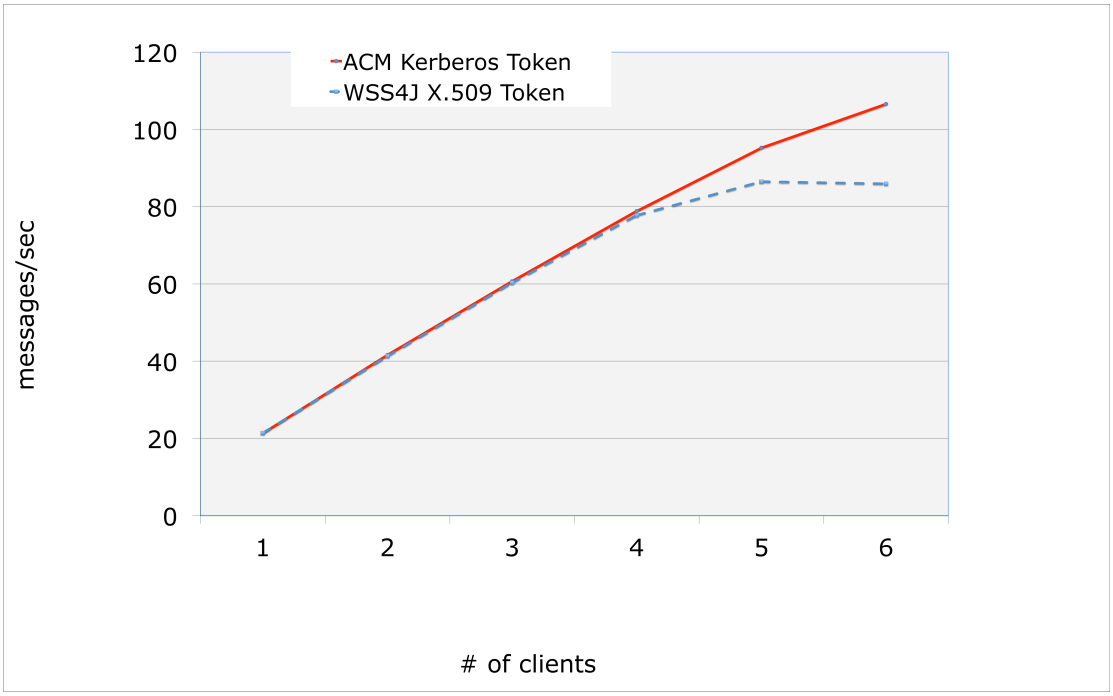
Το μέγεθος του Effective Throughput (σε bytes/sec), όπως το ορίσαμε παραπάνω, εκφράζει τον μέσο ρυθμό των πραγματικών δεδομένων που στέλνονται από τον πελάτη στην υπηρεσία, εξαιρώντας τα σταθερά κόστη μεταφοράς (overheads). Στα κόστη μεταφοράς η επικεφαλίδα ασφαλείας, η επικεφαλίδα του http και τέλος τις επικεφαλίδες των πρωτοκόλλων TCP και IP. Ωστόσο, αυτές οι επικεφαλίδες μπορούν για λόγους απλότητας να θεωρηθούν σχεδόν ιδίου μεγέθους για τους δύο υπό εξέταση μηχανισμούς (Kerberos και X.509 Token Profiles), και επομένως να αγνοηθούν κατά την σύγκριση των δύο handlers.

Τέλος ένα ακόμα μέγεθος που αξιολογήσαμε είναι ο **Μέσος Χρόνος Εξυπηρέτησης** (σε sec) που προκύπτει από την μέση τιμή του Total Handler Processing Time και εκφράζει τον χρόνο (μέση τιμή) που χρειάζεται ένας handler να εξυπηρετήσει μια αίτηση, διεκπεραιώνοντας όλες τις λειτουργίες ασφαλείας όπως περιγράφηκαν.

6.4. Μετρήσεις και Αποτελέσματα

6.4.1. Σενάριο SC1 – Εμπιστευτικότητα μηνύματος υπό αυξανόμενο αριθμό πελατών

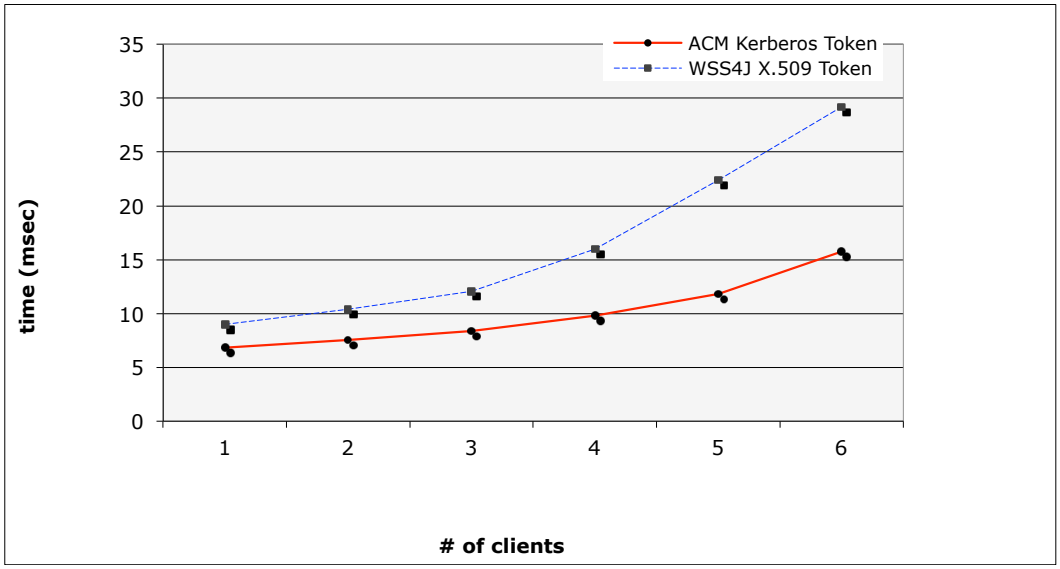
Στο σενάριο SC1 διερευνούμε πως οι δύο handlers ασφαλείας συμπεριφέρονται καθώς αυξάνεται ο αριθμός των πελατών (χρηστών) και επομένως αυξάνεται και υπολογιστικό φορτίο στον handler. Όπως αναφέραμε, κάθε πελάτης αποστέλλει μηνύματα με σταθερό ρυθμό (21 μηνύματα/sec) και με σταθερό μέγεθος του σώματος του μηνύματος SOAP (60 bytes).



Σχήμα 19: Μέση Ρυθμαπόδοση για αυξανόμενο αριθμό πελατών

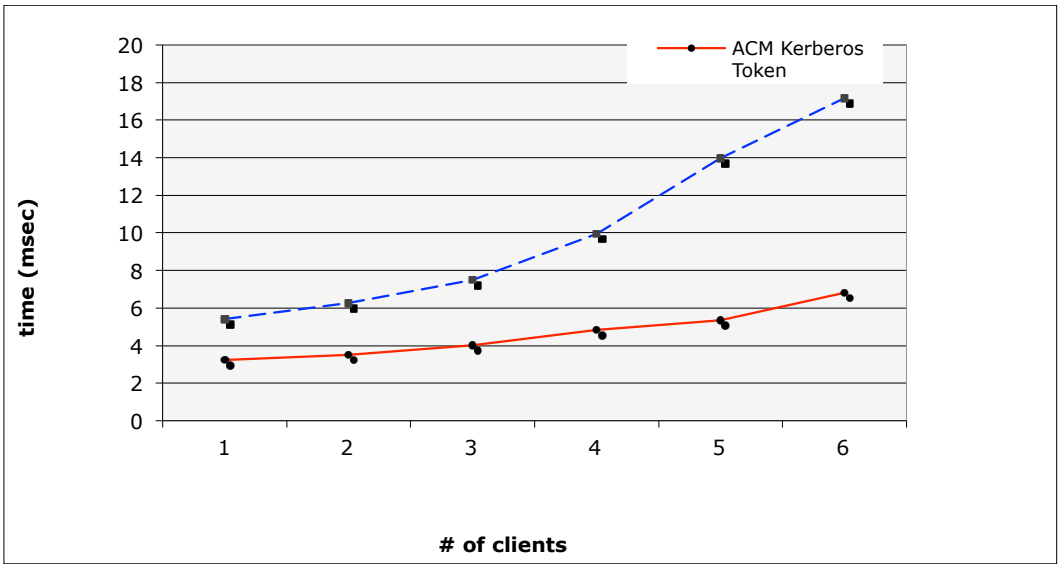
Στο Σχήμα 19, παρουσιάζεται ο μέσος ρυθμός μηνυμάτων για τους δύο handlers σε συνάρτηση με την αύξηση των πελατών (χρηστών) που συμμετέχουν.

Όπως παρατηρούμε, και οι δύο handlers (ACM Kerberos Token handler και WSS4J X.509 Token handler) που υλοποιούν τα πρωτόκολλα WS Kerberos Token Profile και X.509 Certificate Token Profile αντίστοιχα, παρουσιάζουν ίδια συμπεριφορά μέχρι τους τέσσερις πελάτες. Παρατηρώντας την χρησιμοποιούμενη υπολογιστική χρήση κατά την εκτέλεση των πειραμάτων, διαπιστώσαμε ότι ο X.509 handler χρησιμοποιούσε περισσότερη υπολογιστική ισχύ σε σχέση με τον Kerberos handler. Συγκεκριμένα όταν χρησιμοποιήσαμε τέσσερις πελάτες, η επεξεργαστική χρήση ήταν ~90% στην περίπτωση του X.509, ενώ στην περίπτωση του Kerberos ήταν ~65%. Η διαφορά στην επεξεργαστική χρήση οφείλεται ουσιαστικά στον διαφορετικό τρόπο εξαγωγής του ιδιωτικού κλειδιού κρυπτογράφησης, αφού όλοι οι άλλοι παράγοντες (αλγόριθμοι κρυπτογράφησης σώματος μηνύματος, αποκωδικοποίηση μηνύματος SOAP κτλ.) είναι κοινοί. Ο αλγόριθμος Δημοσίου Κλειδιού RSA που χρησιμοποιεί ο X.509 handler για να εξάγει το συμμετρικό κλειδί, όπως είδαμε και στο 6.2.3, συμβάλει στην υψηλή επεξεργαστική χρήση. Η διαφορά στην υπολογιστική χρήση επιτρέπει στον Kerberos handler να παρέχει βελτιωμένη απόδοση εξυπηρετώντας πλήρως μέχρι 6 πελάτες και φτάνοντας το Μέση Ρυθμαπόδοση στα 107 μηνύματα/sec.



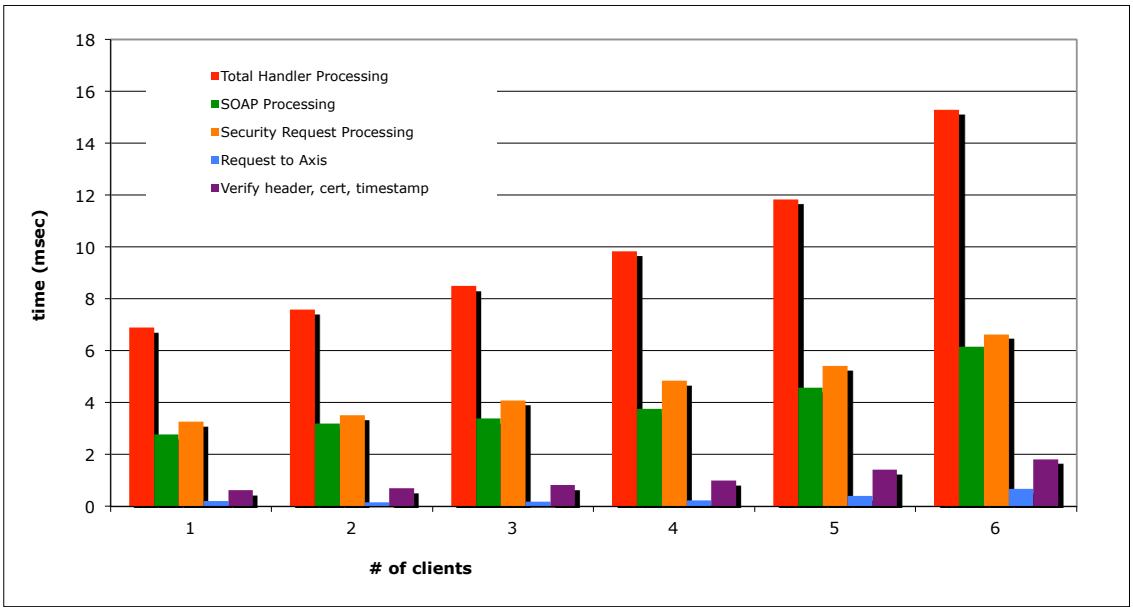
Σχήμα 20: Μέσος χρόνος εξυπηρέτησης στον handler μια αίτησης (Total Handler Processing)

Στο Σχήμα 20 παρουσιάζουμε την επεκτασιμότητα (scalability) της λύσης που προτείνουμε σε συνάρτηση με τον αριθμό των πελατών. Αν και μέχρι τους 4 πελάτες η ρυθμαπόδοση είναι σχεδόν πανομοιότυπη για τις δύο προσεγγίσεις, όπως φαίνεται στο Σχήμα 19, στο Σχήμα 20 αποκαλύπτεται ότι ο ACM handler παρουσιάζει σημαντικά μικρότερους χρόνους εξυπηρέτησης και επομένως καλύτερους χρόνος απόκρισης της υπηρεσίας. Ακόμα και σε περιπτώσεις που η υπολογιστική ισχύς είναι αρκετή, ο μικρότερος χρόνος απόκρισης που παρουσιάζει η λύση που προτείνουμε στην αρχιτεκτονική, κάνει την προσέγγιση μας καλύτερη για λύσεις όπου ο χρόνος απόκρισης είναι σημαντικός, όπως στα Instrumentation Grids. Στην συνέχεια θα δούμε του χρόνους που απαιτούν οι λειτουργίες ασφαλείας στο σενάριο SC1.



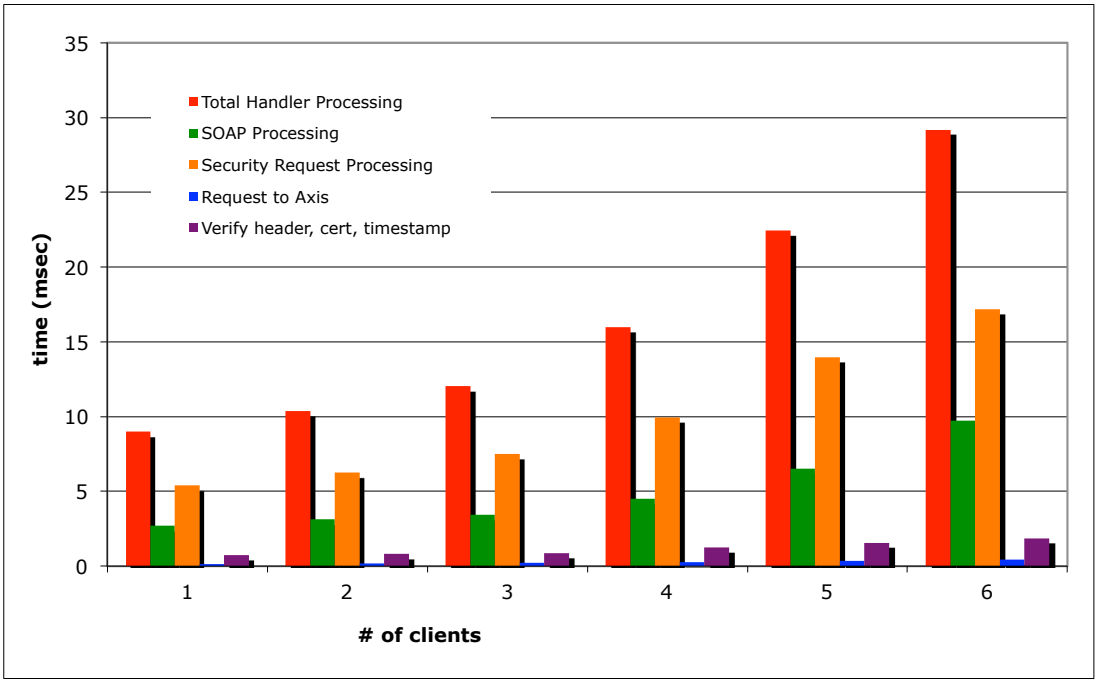
Σχήμα 21: Μέσος Χρόνος Επεξεργασίας Λειτουργιών Ασφαλείας Αίτησης (Security Request Processing)

Ο Μέσος Χρόνος Επεξεργασίας που απαιτούν οι λειτουργίες ασφάλειας σε συνάρτηση του αριθμού πελατών παρουσιάζεται στο Σχήμα 21 με χρήση των τροποποιήσεων που κάναμε στους handlers (δες παράγραφο 6.3). Από το σχήμα αυτό προκύπτει πως οι χρόνοι των λειτουργιών ασφαλείας (εξαγωγή συμμετρικού κλειδιού και αποκρυπτογράφηση του μηνύματος) είναι οι κύριοι υπεύθυνοι για την διαφορά στην συνολική επεξεργασία του handler που είδαμε στο Σχήμα 20. Επίσης βλέπουμε πως αυτοί οι χρόνοι αυξάνουν με την αύξηση των πελατών στο σύστημα και μάλιστα στην περίπτωση του X.509 handler η αύξηση γίνεται μεγαλύτερη, διευρύνοντας την διαφορά επεξεργασίας των δύο μηχανισμών.



Σχήμα 22: Μέσες τιμές Χρόνων Επεξεργασίας του ACM Kerberos Token handler

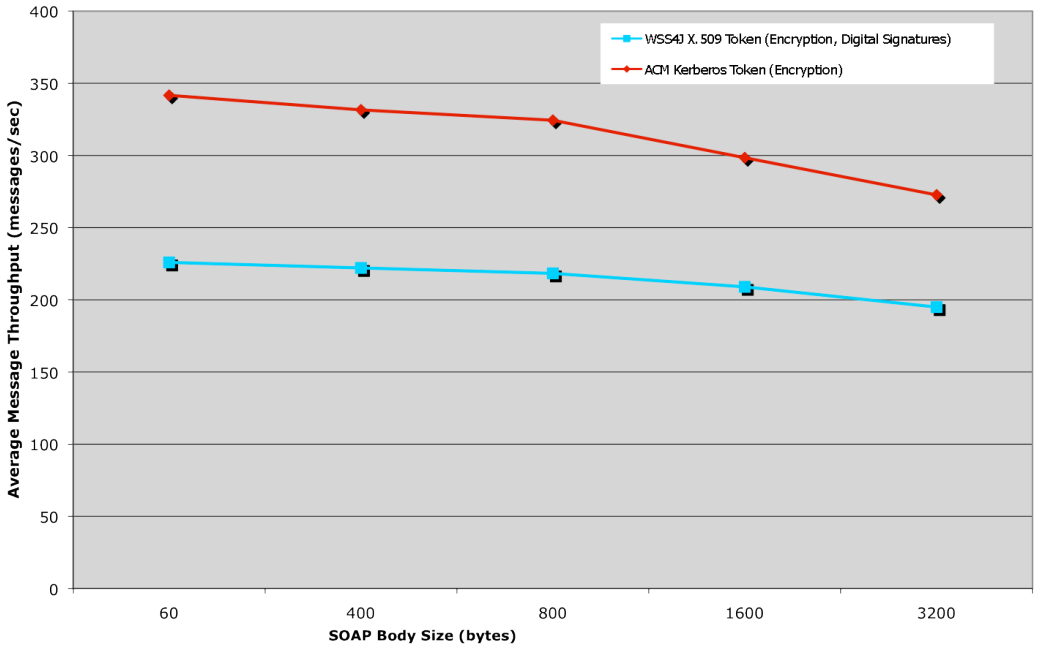
Στα Σχήματα 22 και 23 παρουσιάζονται ο συνολικός και οι επιμέρους μέσοι χρόνοι επεξεργασίας των δύο handlers σαν συνάρτηση του αριθμού των πελατών. Επιβεβαιώνεται ότι ο χρόνος SOAP Processing που εκφράζει την αποκωδικοποίηση του μηνύματος SOAP είναι σημαντικός όπως άλλωστε έχει αναφερθεί στην σχετική βιβλιογραφία [Chiu02], [Ghaz05]. Παρατηρούμε επίσης πως οι ο χρόνος Security Request Processing είναι συγκρίσιμος εξίσου σημαντικός: Στην περίπτωση του WSS4J X.509 Token handler είναι περίπου διπλάσιος από τον SOAP Processing, ενώ στην προτεινόμενη αρχιτεκτονική (ACM Kerberos Token handler) είναι περίπου ίσος.



Σχήμα 23: Μέσες τιμές Χρόνων Επεξεργασίας του WSS4J X.509 Token handler

6.4.2. Σενάριο SC2 – Ταυτοποίηση, Ακεραιότητα και Εμπιστευτικότητα Μηνύματος υπό αυξανόμενο μέγεθος μηνυμάτων SOAP σε υψηλό υπολογιστικό φορτίο

Στην παρούσα ενότητα, θα παρουσιάσουμε τα συγκριτικά αποτελέσματα σύμφωνα με το σενάριο SC2. Οι λειτουργίες της ταυτοποίηση-ακεραιότητα-εμπιστευτικότητα μηνύματος σε κάθε handler προσφέρονται μέσω των μηχανισμών ψηφιακής υπογραφής και κρυπτογράφησης, όπως είδαμε στις παραγράφους 6.2.3 και 6.2.4. Επίσης επισημάνουμε πως ο υπολογιστικός φόρτος στους handlers δημιουργείται κυρίως από επεξεργασία των μηνυμάτων που αποστέλλουν οι 16 πελάτες. Η απλή υπηρεσία “echo” που επιλέξαμε επιβαρύνει ελάχιστα στον υπολογιστικό φόρτο.



Σχήμα 24: Μέση Ρυθμαπόδοση για ACM Kerberos Token Profile και WSS4J X.509 Certificate Token Profile handlers για διαφορετικά μεγέθη του σώματος του SOAP μηνύματος

Στο Σχήμα 24 παρουσιάζεται η μεταβολή της Μέσης Ρυθμαπόδοσης χρησιμοποιώντας κρυπτογράφηση και ηλεκτρονικές υπογραφές ως συνάρτηση του μεγέθους του σώματος του μηνύματος. Παρατηρούμε ότι υπάρχει φθίνουσα απόδοση και των δύο λύσεων καθώς το μέγεθος αυξάνει. Η λύση του Kerberos που ενσωματώνει η αρχιτεκτονική της διατριβής παρουσιάζει σημαντικά μεγαλύτερη απόδοση όπως φαίνεται και στον Πίνακα 7.

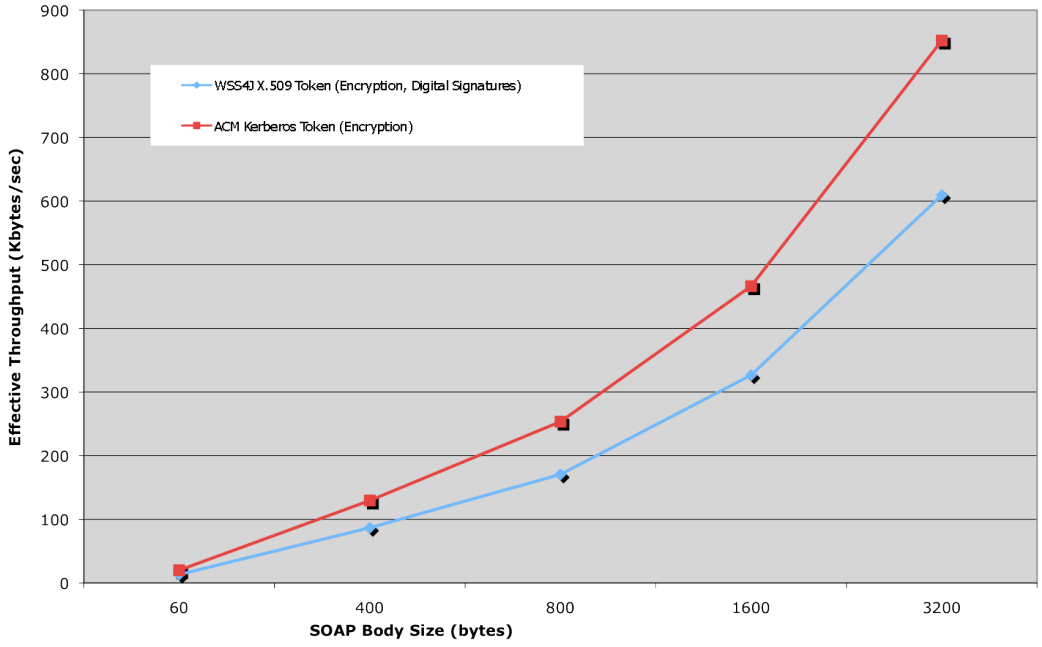
Πίνακας 7: Μέση Ρυθμαπόδοση με χρήση κρυπτογράφησης και ηλεκτρονικών υπογραφών για Kerberos και X.509 Token Profiles

SOAP Body Size (bytes)	***60***	***400***	***800***	***1600***	***3200***
X509 (messages/sec)	226	222	218	209	195
Kerberos (messages/sec)	342	332	324	298	273
Gain % ***(|X509 - Kerberos|/X509)*100***	51	49	48	43	40

Ο πίνακας παρουσιάζει αριθμητικά τα αποτελέσματα που απεικονίζονται στο Σχήμα 24 μαζί με το υπολογισμένο κέρδος απόδοσης επί τις % του Kerberos handler σε σχέση με τον X.509 handler.

Όπως παρατηρούμε στο Σχήμα 24 και στον Πίνακα 7, η προσέγγιση του ACM

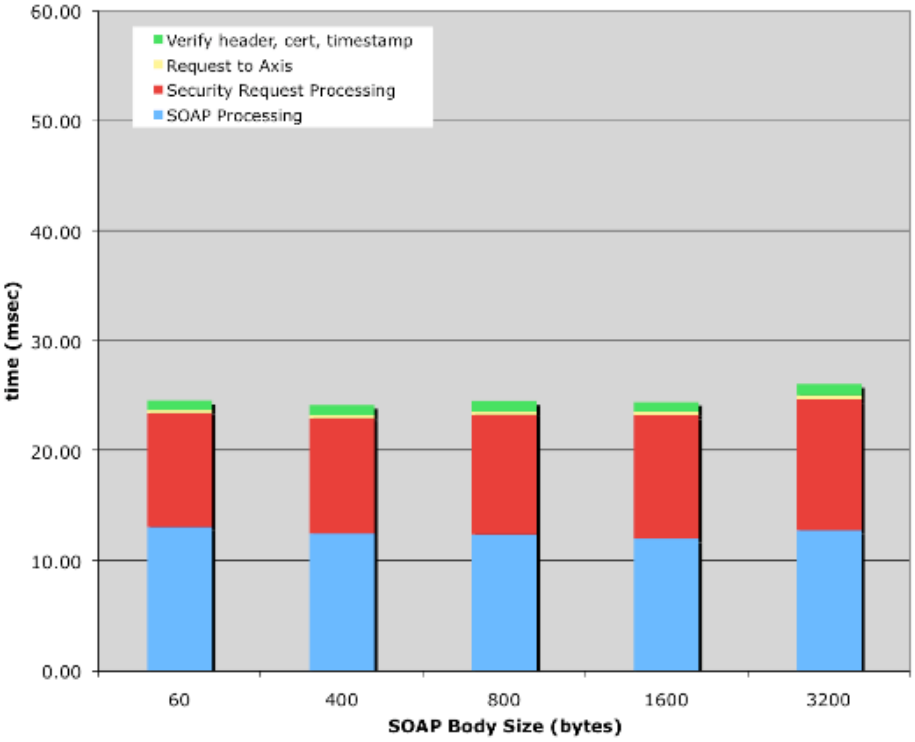
που υλοποιεί το πρωτόκολλο Kerberos προσφέρει σημαντικά βελτιωμένη απόδοση της Μέση Ρυθμαπόδοση ειδικά για μικρές τιμές μεγέθους σώματος μηνύματος που αναμένουμε σε ένα Instrumentation Grid. Ειδικότερα, η προσέγγιση Kerberos πετυχαίνει 51% βελτίωση πάνω από την προσέγγιση X.509 σε μέγεθος σώματος μηνύματος 60 bytes. Αυτή η βελτίωση οφείλεται στην ταχύτερη και αποδοτικότερη εξαγωγή συμμετρικών κλειδιών που εμφανίζει η υλοποίηση του μηχανισμού Kerberos, διότι ακολουθείται συμμετρική κρυπτογραφία. Αντίθετα, στην περίπτωση του X.509 handler υιοθετείται ασύμμετρη κρυπτογραφία για την εξαγωγή του συμμετρικού κλειδιού. Οι διαφορές ανάμεσα στους δύο handlers μειώνονται καθώς αυξάνεται το μέγεθος του σώματος του μηνύματος (π.χ. 40% για μέγεθος 3200 bytes), καθώς και οι δύο προσεγγίσεις (Kerberos και X.509) χρησιμοποιούν συμμετρική κρυπτογραφία για την αποκρυπτογράφηση του σώματος του μηνύματος: Καθώς το μέγεθος του κρυπτογραφημένου μηνύματος μεγαλώνει, αυξάνονται οι επεξεργαστικές απαιτήσεις της αποκρυπτογράφησης του σώματος σε σχέση με την λειτουργία εξαγωγής των συμμετρικών κλειδιών.



Σχήμα 25: Effective Throughput για διαφορετικά μεγέθη Σώματος SOAP μηνύματος

Παρόμοιες παρατηρήσεις και αποτελέσματα εξάγονται, όπως είναι αναμενόμενο, εξετάζοντας την συμπεριφορά του Effective Throughput ως συνάρτηση του μεγέθους μηνύματος (Σχήμα 25). Παρατηρούμε ότι αν και ο αριθμός των μηνυμάτων μειώνεται, η πληροφορία που μεταδίδεται αυξάνεται. Και σε αυτήν την

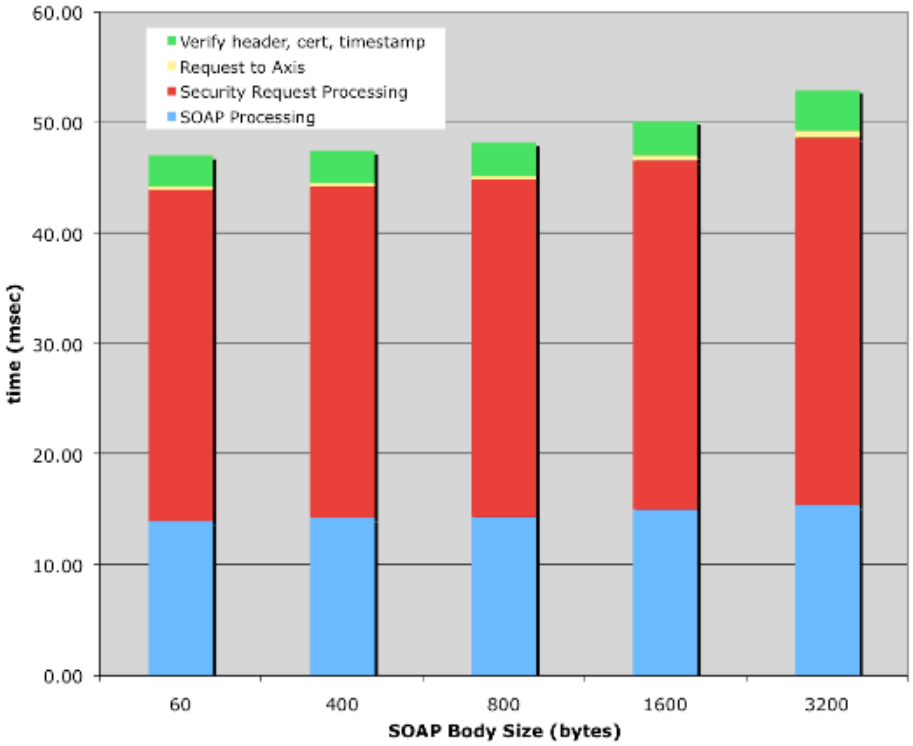
περίπτωση ο μηχανισμός του Kerberos εμφανίζει αυξημένη απόδοση λόγο των περισσότερων μηνυμάτων που μπορεί να εξυπηρετεί. Επίσης παρατηρούμε από τα Σχήματα 24 και 25 ότι η όλη διαδικασία και στους δύο handlers περιορίζεται κυρίως από την Ρυθμαπόδοση μηνυμάτων (μηνύματα/sec), και όχι από τον όγκο της πληροφορίας που μεταδίδεται (bytes/sec). Άλλωστε, στις περιπτώσεις που μας ενδιαφέρουν, δηλαδή σε περιβάλλον Instrumentation Grid, το μέγεθος την Ρυθμαπόδοσης μηνυμάτων παίζει το βασικό ρόλο.



Σχήμα 26: Υπολογιστικοί Χρόνοι χρησιμοποιώντας Kerberos Security ταυτοποίηση-ακεραιότητα-εμπιστευτικότητα μηνύματος

Στα Σχήματα 26 και 27 απεικονίζονται οι μέσες τιμές των διαφόρων υπολογιστικών χρόνων, όπως τους ορίσαμε στην ενότητα 6.3, για τον Kerberos και X.509 Certificate handler αντίστοιχα. Παρατηρούμε πως ο συνολικός υπολογιστικός χρόνος στην περίπτωση του WSS4J X.509 Token handler είναι σχεδόν διπλάσιος από αυτόν του ACM Kerberos Token. Αυτή η διαφορά στην απόδοση οφείλεται στους λόγους που αναφέραμε στο SC1.

Υπολογίσαμε επίσης την **Τυπική Απόκλιση (Standard Deviation *σ*)** στα δεδομένα των δύο μετρήσεων (X.509-Kerberos) και βρήκαμε ότι είναι περίπου 25ms για τον Kerberos handler και περίπου 38ms για τον X.509 handler για όλα τα μεγέθη σώματος SOAP μηνύματος.

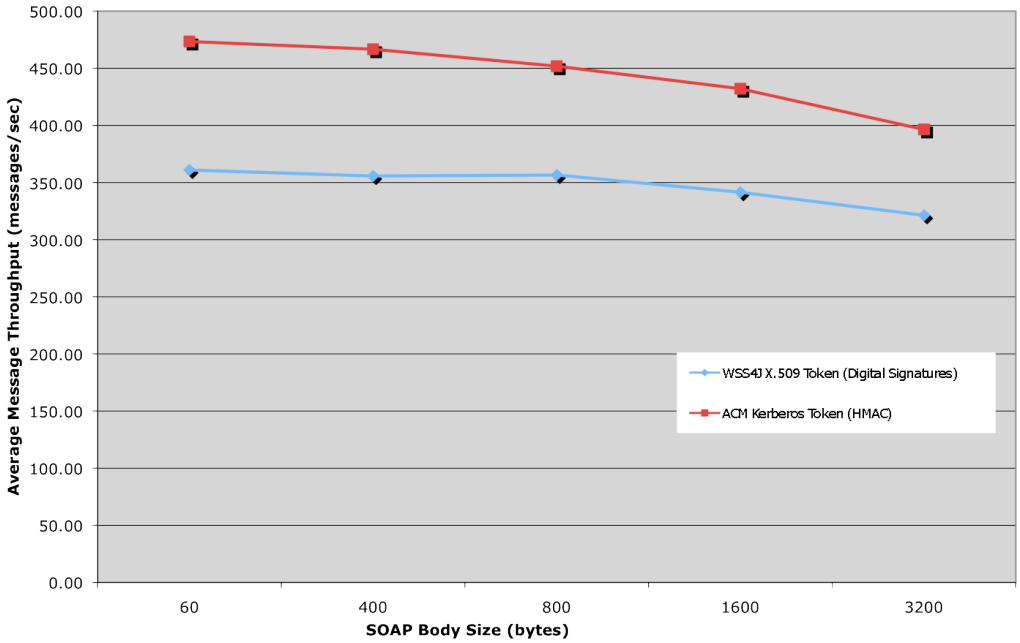


Σχήμα 27: Υπολογιστικοί Χρόνοι χρησιμοποιώντας τον WSS4J X.509 Token handler για ταυτοποίηση-ακεραιότητα-εμπιστευτικότητα μηνύματος

Το στατιστικό μέγεθος της Τυπικής Απόκλισης αντικατοπτρίζει την διαφοροποίηση της καθυστέρησης (delay jitter). Είναι σημαντικό μέγεθος σε Instrumentation Grids, αφού εξασφαλίζει μεγαλύτερη σταθερότητα στην καθυστέρηση μεταξύ μηνυμάτων και επομένως αυξημένη δυνατότητα ομαλότερου ελέγχου στο χρήστη (better user experience).

6.4.3. Σενάριο SC3 – Ταυτοποίηση και Ακεραιότητα Μηνύματος υπό αυξανόμενο μέγεθος μηνυμάτων SOAP σε υψηλό υπολογιστικό φορτίο

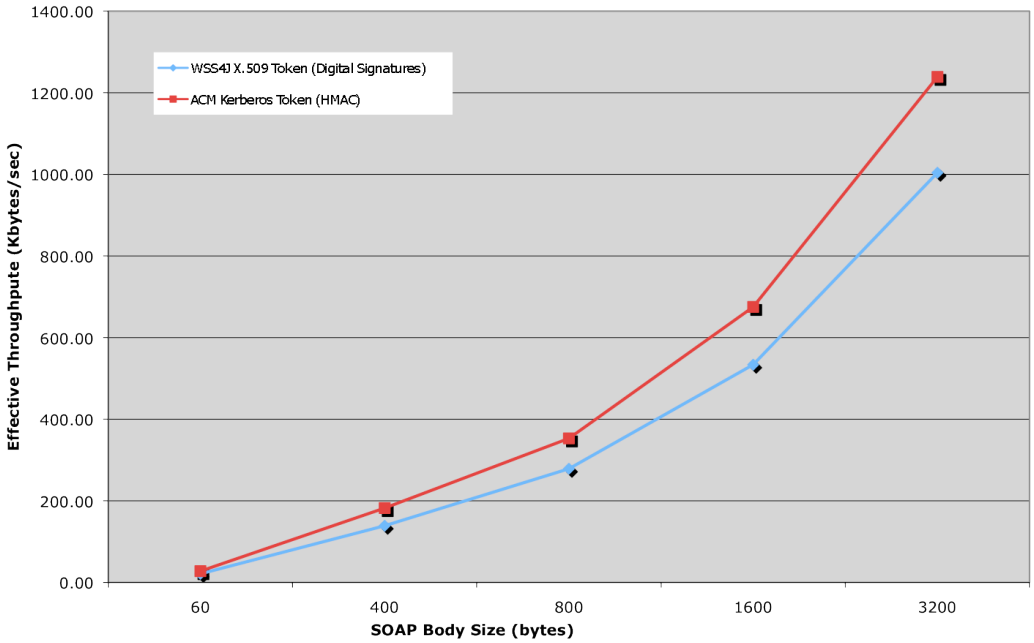
Τα Σχήματα 28 και 29 απεικονίζουν την βελτίωση της απόδοσης της υλοποίησης μας του ACM Kerberos Token handler σε σχέση με αυτήν του WSS4J X.509 Certificate handler στο σενάριο SC3. Όπως είναι αναμενόμενο, τα αποτελέσματα ακολουθούν την ίδια τάση σε σχέση με το SC2. Υπενθυμίζουμε πως η ακεραιότητα του μηνύματος εξασφαλίζεται εφαρμόζοντας στο σώμα του μηνύματος SOAP στην μεν περίπτωση των πιστοποιητικών X.509 με ηλεκτρονικές υπογραφές, στη δε περίπτωση του πρωτοκόλλου Kerberos με κωδικούς HMAC.



Σχήμα 28: Μέση Ρυθμαπόδοση για τους WSS4J X.509 Token και ACM Kerberos Token handlers με χρήση ψηφιακών υπογραφών στο μήνυμα (ταυτοποίηση-ακεραιότητα μηνύματος)

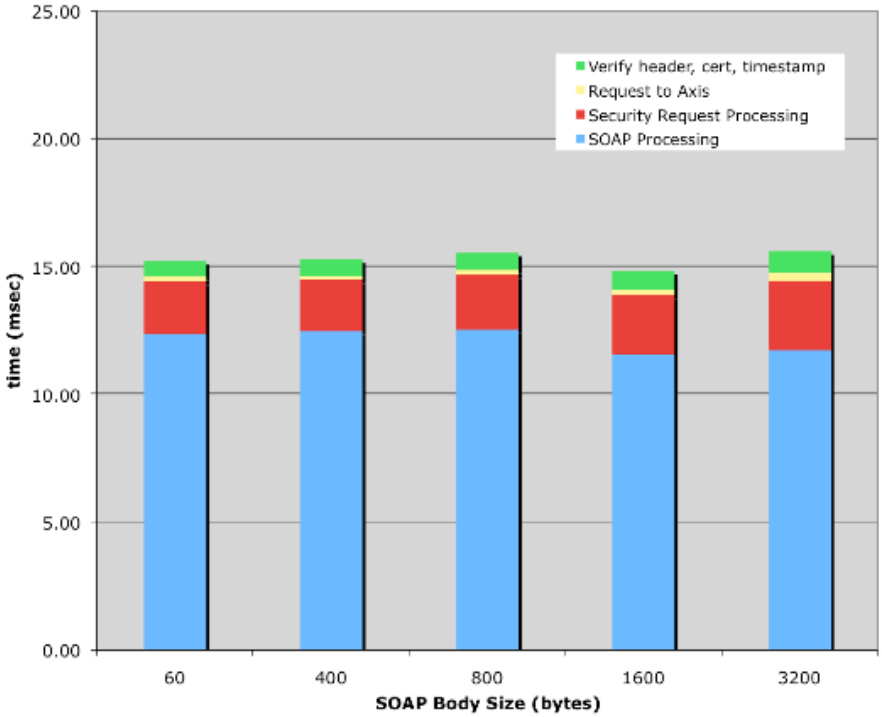
Το πλεονέκτημα των HMAC που υιοθετεί ο ACM Kerberos Token handler κατά την ανταλλαγή μηνυμάτων, για την παροχή ταυτοποίησης και ακεραιότητας μηνύματος είναι εμφανές, σε σχέση με τις ηλεκτρονικές υπογραφές που χρησιμοποιεί ο WSS4J X.509 Token handler. Οι κωδικοί HMAC βασίζονται σε συμμετρική κρυπτογραφία, ενώ οι ηλεκτρονικές υπογραφές (Digital Signatures) βασίζονται σε κρυπτογραφία Δημοσίου Κλειδιού.

Επίσης, παρατηρούμε ότι στο SC3 έχουμε εμφανή βελτίωση σε σχέση με τα αποτελέσματα του σεναρίου SC2. Στο SC3 η Μέση Ρυθμαπόδοση του ACM handler αγγίζει τα 473 μηνύματα/sec ενώ στο SC2 έχουμε 342 μηνύματα/sec. Η βελτίωση κατά 38% στην Μέση Ρυθμαπόδοση επιτυγχάνεται στο SC3 θυσιάζοντας την εμπιστευτικότητα μηνύματος που παρέχεται μέσω της κρυπτογράφησης του σώματος στο SC2. Επομένως επαφίεται στον σχεδιαστή μιας εφαρμογής ενός Instrumentation Grid να επιλέξει την ασφάλεια των μηνυμάτων που επιθυμεί.

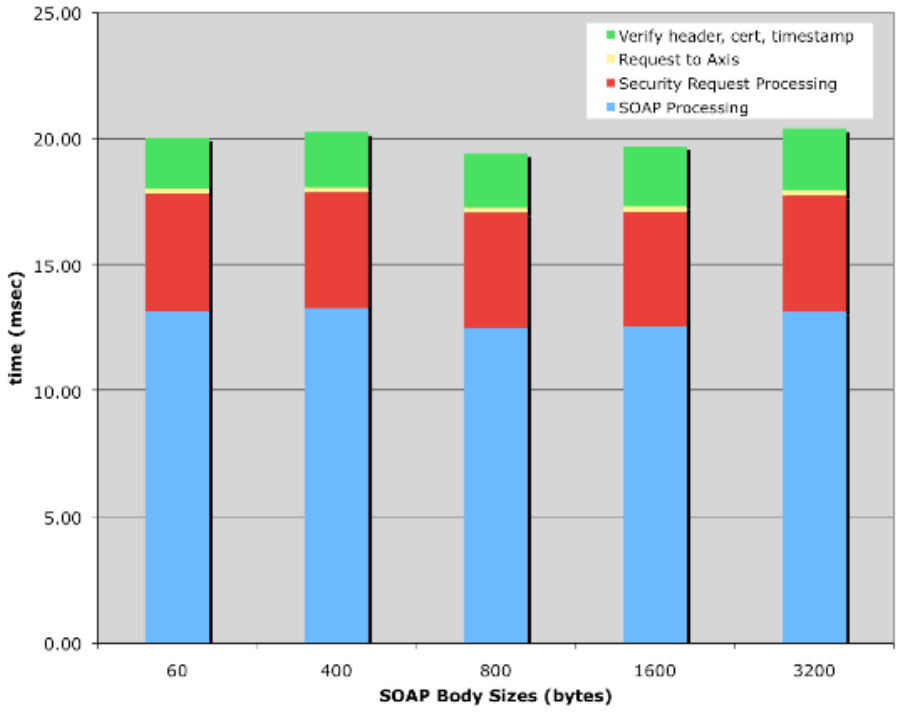


Σχήμα 29: Effective Throughput για τα δύο handlers με την χρήση ηλεκτρονικών υπογραφών για ταυτοποίηση-ακεραιότητα μηνύματος

Τα Σχήματα 30 και 31 παρουσιάζουν αναλυτικά τους αντίστοιχους χρόνους επεξεργασίας των handlers όπως και στα προηγούμενα σενάρια. Οι χρόνοι βρίσκονται σε συμφωνία με τους χρόνους στο σενάριο SC2. Ωστόσο παρουσιάζουν μικρότερους συνολικούς χρόνους που οφείλονται στο μικρότερο χρόνο επεξεργασίας των λειτουργιών ασφαλείας (Security Request Processing), καθώς δεν απαιτείται χρόνος για να αποκρυπτογραφήσουν το μήνυμα, παρά μόνο να επαληθεύσουν την ψηφιακή υπογραφή.



Σχήμα 30: Χρόνοι Επεξεργασία του ACM Kerberos Token handler με χρήση ηλεκτρονικών υπογραφών (HMAC)



Σχήμα 31: Χρόνοι Επεξεργασίας του WSS4J X.509 Certificate handler με την χρήση ηλεκτρονικών υπογραφών (Digital Signature)

7. Συμπεράσματα

7.1. Σύνοψη

Στην παρούσα διατριβή προτάθηκε και υλοποιήθηκε μια αρχιτεκτονική ασφαλείας που έχει ως στόχο δώσει λύση στο πρόβλημα της απόδοσης του συστήματος ασφαλείας ενός Instrumentation Grid, βελτιώνοντας τους χρόνους απόκρισης των υπηρεσιών. Ένα Instrumentation Grid είναι ένα κατανεμημένο σύστημα μετρήσεων και ελέγχου οργάνων που ακολουθεί την αρχιτεκτονική και το ενδιάμεσο λογισμικό ενός Computational Grid. Στην εκδοχή του Instrumentation Grid που θεωρήσαμε, τα όργανα μετρήσεων και ελέγχου αντιμετωπίζονται ως πόροι και οι λειτουργίες είναι προσβάσιμες στο υπόλοιπο Grid μέσω κατάλληλων υπηρεσιών (π.χ. Instrument Elements - IEs) ακολουθώντας τις γενικές αρχές των SOA και των Web Services. Στο περιβάλλον των Instrumentation Grid, ο έλεγχος των οργάνων επιβάλλει αμεσότητα στο έλεγχο και επομένως μικρούς χρόνους στην απόκριση των ενεργειών ελέγχου των επιστημονικών οργάνων.

Η αρχιτεκτονική ασφαλείας που σχεδιάσαμε και υλοποιήσαμε, ως προσανατολισμένη σε τεχνολογίες Web Services, προσφέρει ταυτοποίηση, εμπιστευτικότητα και ακεραιότητα των μηνυμάτων SOAP που ελέγχουν τις λειτουργίες του οργάνων. Τα μηνύματα αποστέλλονται από τους πελάτες (χρήστης – ροές εργασίας) προς τις υπηρεσίες που αντιπροσωπεύουν τα όργανα. Οι λειτουργίες ασφαλείας στο μήνυμα (authentication, integrity, confidentiality) προσφέρονται υιοθετώντας το πρωτόκολλο WS Security. Βασίζεται στο πρωτόκολλο Kerberos για να παρέχει Single Sign On (SSO) και ασφαλή ανταλλαγή κλειδιών (μέσω service tickets) μεταξύ υπηρεσιών και πελατών, ορίζοντας μια ασφαλή σύνοδο (session) μεταξύ τους που να ικανοποιεί τις αυξημένες απαιτήσεις χρόνου απόκρισης ενός Instrumentation Grid. Για το λόγο αυτό χρησιμοποιεί κρυπτογραφία συμμετρικού κλειδιού για όλες τις λειτουργίες ασφαλείας στην ανταλλαγή μηνυμάτων από χρήστες σε οι οποίοι εξ' αρχής έχουν ταυτοποιηθεί στο Kerberos KDC. Μόνο η αρχική ταυτοποίηση του χρήστη γίνεται με βάση την ασύμμετρη κρυπτογραφία Δημοσίου Κλειδιού μέσω του πρωτοκόλλου PKINIT. Η αρχιτεκτονική μας είναι σχεδιασμένη να λειτουργεί ως μέρος μιας ευρύτερης αρχιτεκτονικής Grid (OGSA) και προσφέρει

διαλειτουργικότητα με την defacto υποδομή ασφαλείας για Grid, το GSI που βασίζεται σε PKI. Την υποστήριξη του GSI την υλοποιήσαμε με την χρήση του πρωτοκόλλου PKINIT που επιτρέπει το login στο Kerberos KDC με την χρήση πιστοποιητικών X.509, ορίζοντας κανόνες και μηχανισμούς αντιστοίχισης ταυτότητας Kerberos Principal σε X.509 Certificate και υποστήριξης της εκπροσώπευσης χρηστών (proxy delegation). Τέλος η αρχιτεκτονική μας υποστηρίζει έλεγχο πρόσβασης (authorization) με την χρήση απλών κανόνων πρόσβασης και με βάση τον ρόλο του χρήστη για τον έλεγχο των συσκευών.

Η αρχιτεκτονική που υλοποιήσαμε αποτελείται από 4 υποσυστήματα: Kerberos KDC, KrbClient, ACM και Policy Repository.

Το Kerberos KDC αποτελεί την Third Trusted Party – TTP της αρχιτεκτονικής, η οποία ταυτοποιεί χρήστες αρχικά (SSO) και στην συνέχεια τους παρέχει εισιτήρια (service tickets) για να καλέσουν τις υπηρεσίες και να γίνει η αμοιβαία ταυτοποίηση χρηστών και υπηρεσιών. Επίσης μέσω του μηχανισμού των service tickets μοιράζει κρυπτογραφικά κλειδιά για την ασφαλή και ταχεία επικοινωνία μεταξύ τους.

Ο KrbClient σχεδιάστηκε και υλοποιήθηκε ώστε να παρέχει στον προγραμματιστή του λογισμικού πελάτη (client software) τις λειτουργίες της αρχιτεκτονικής ασφαλείας, μέσω ενός API. Επιτρέπει την ταυτοποίηση του χρήστη μέσω πιστοποιητικού GSI X.509, αιτείται αυτόματα την έκδοση Kerberos service ticket για την υπηρεσία που θέλει να έχει πρόσβαση ο χρήστης, ενσωματώνει σε κάθε αίτηση του χρήστη (SOAP Request) τα διαπιστευτήρια ασφαλείας (service tickets) για το πρώτο μήνυμα SOAP (στα επόμενα στέλνει αναφορά σε αυτά) και κρυπτογραφεί/υπογράφει το σώμα του μηνύματος. Η επεξεργασία ασφαλείας που εφαρμόζει στο μήνυμα ακολουθεί το πρωτόκολλο WS Security Kerberos Token Profile. Τέλος ο KrbClient διατηρεί την κατάσταση ασφαλείας του χρήστη, δηλαδή το πιστοποιητικό του και τα εισιτήρια (TGT και service tickets) του. Υλοποιήθηκε ως κλάση Java ώστε να ενσωματώνεται εύκολα σε οποιαδήποτε αρχιτεκτονική client.

Ο Access Control Manager (ACM) είναι το βασικό υποσύστημα της αρχιτεκτονικής που σχεδιάσαμε για να μειώσουμε τους χρόνους απόκρισης των υπηρεσιών του Instrumentation Grid, μέσω της βελτίωσης στην απόδοση των λειτουργιών ασφαλείας. Προστατεύει, όπως είδαμε στο 4^{ο} και 5^{ο} Κεφάλαιο, την υπηρεσία, παρέχοντας ταυτοποίηση (authentication), ακεραιότητα, εμπιστευτικότητα

και έλεγχο πρόσβασης (authorization) σε επίπεδο του μηνύματος. Υποστηρίζει το πρωτόκολλο Kerberos για τον ορισμό συνόδου ανταλλαγής μηνυμάτων με βάση το κλειδί του εισιτηρίου. Επίσης ο ACM αρνείται ή επιτρέπει την πρόσβαση στο όργανο μέσω των τοπικών κανόνων που εκφράζουν την τοπική πολιτική που θέτει ο διαχειριστής του πόρου/οργάνου πίσω από την υπηρεσία. Ο ACM υλοποιήθηκε σε περιβάλλον Java με χρήση του AXIS και βιβλιοθήκης WSS4J. Δημιουργήθηκε ως handler που εμπλέκεται στην επεξεργασία του μηνύματος, εκτελώντας όλες τις λειτουργίες ασφαλείας. Τέλος δημιουργήθηκαν οι απαραίτητες κλάσεις για να υποστηριχθεί το WSS Kerberos Token Profile, αφού δεν υπήρχε η απαραίτητη υποστήριξη από την βιβλιοθήκη WSS4J.

Το Policy Repository σχεδιάστηκε ως αποθετήριο των τοπικών κανόνων πρόσβασης (authorization) λειτουργώντας πληροφοριακά. Προσφέρει στο λογισμικό του πελάτη την συνολική εικόνα πρόσβασης στο Instrumentation Grid, δηλαδή τι επιτρέπεται να εκτελέσει σε κάθε υπηρεσία ο χρήστης. Έχει υλοποιηθεί ως Web Service με αποθήκευση των κανόνων σε βάση δεδομένων.

Τα αποτελέσματα των εξομοιώσεων (emulation) που παρουσιάσαμε στο 6^{ο} Κεφάλαιο επιβεβαίωσαν την προσδοκώμενη από την αρχιτεκτονική βελτίωση στην απόκριση της υπηρεσίας. Χρησιμοποιώντας διάφορα σενάρια παραγωγής κίνησης (παραγόμενη από διαφορετικούς υπολογιστές σε τοπικό δίκτυο TCP/IP) συγκρίναμε την απόδοση του ACM και του WSS4J handler (εγκατεστημένων σε έναν εξυπηρετητή στο τοπικό δίκτυο). Οι μετρήσεις των handlers αντικατοπτρίζουν την απόδοση των λειτουργιών ασφαλείας και συγκεκριμένα των πρωτοκόλλων WSS Kerberos Token Profile και WSS X.509 Certificate Token Profile. Στα σενάρια υψηλής κίνησης μηνυμάτων προέκυψε σημαντική διαφορά (μέχρι και 50%) στην ρυθμαπόδοση και τους χρόνους εξυπηρέτησης μηνυμάτων. Η υλοποίηση μας επίσης συμπεριφέρεται καλύτερα, προσφέροντας μικρότερους χρόνους επεξεργασίας και επομένως μικρότερους χρόνους απόκρισης της υπηρεσίας και σε σενάρια χαμηλής κίνησης.

7.2. Ανοιχτά Θέματα – Μελλοντικές Κατευθύνσεις

Από τα θέματα της διατριβής που παρουσιάστηκαν ανακύπτουν μερικές μελλοντικές κατευθύνσεις και επεκτάσεις όπως:

- Μια επέκταση της αρχιτεκτονικής είναι η υποστήριξη του πρωτοκόλλου WS-Policy (βλέπε παράγραφο 3.5.6), ώστε να ορίζεται στο αρχείο WSDL η πολιτική ασφαλείας της υπηρεσίας που προστατεύεται από τον ACM. Έτσι ενημερώνονται τρίτοι χρήστες για τους κανόνες χρήσης υπηρεσιών του Instrumentation Grid και διευκολύνεται η «κατανάλωση» της υπηρεσίας από ευρύτερο κοινό ή από χρήστες διαφορετικών περιοχών ασφαλείας σε ομόσπονδο (federated) περιβάλλον. Εξετάζοντας το αρχείο WSDL θα μπορούσε ένας οποιοσδήποτε εξωτερικός client μιας προστατευμένης υπηρεσίας να ανακαλύψει τις παραμέτρους ασφαλείας της ώστε να προβεί σε όλες τις απαραίτητες ενέργειες για να αποκτήσει διαπιστευτήρια συμβατά με την συγκεκριμένη υπηρεσία.
- Η αρχιτεκτονική όπως παρουσιάστηκε χρησιμοποιεί ως Third Trusted Party (TTP) το Key Distribution Center (KDC) του Kerberos. Μια πιθανή επέκταση θα ήταν η χρήση της υπηρεσία Security Token Service (STS) του πρωτοκόλλου WS-Trust μπροστά από το KDC. Αυτή, όπως είδαμε στην ενότητα 3.5.6, δρα ως TTP μεταξύ διαφορετικών περιοχών ασφαλείας (security domain). Το πρωτόκολλο όπως ορίζεται απαιτεί η υπηρεσία να επικοινωνεί με το STS για να ταυτοποιήσει τον χρήστη. Με την υιοθέτηση του STS θα διευκολύναμε την ταυτοποίηση του πελάτη σε ένα federated περιβάλλον. Πιθανές επεκτάσεις αφορούν στην διαλειτουργικότητα με υλοποιήσεις SAML π.χ. Shibboleth [Scav05].
- Η υλοποίηση της αρχιτεκτονικής είναι βασισμένη σε AXIS 1.4. Μια λογική επέκταση της αρχιτεκτονικής είναι η ανάπτυξη του λογισμικού ACM βασισμένο σε νεότερα frameworks όπως το Apache CXF. Μια τέτοια υλοποίηση θα έκανε την αρχιτεκτονική εύκολα ενσωματώσιμη σε βιβλιοθήκες τρίτων που θέλουν να υιοθετήσουν την αρχιτεκτονική μας σε συστήματα που αναπτύσσονται με υπηρεσίες τους σε Java.
- Η αρχιτεκτονική ασφαλείας είναι προσανατολισμένη στην παρούσα φάση σε SOAP Web Services. Ωστόσο όπως είδαμε και στην ενότητα 2.5.1 υπάρχουν και τα RESTful Web Services που βασίζονται στο πρωτόκολλο http. Σε κάποιες περιπτώσεις τέτοιες υπηρεσίες θα μπορούσαν να καλύψουν τις ανάγκες απλών πόρων, όπου οι λειτουργίες (operations) που προσφέρει το http αρκούν. Μια λογική επέκταση της διατριβής θα ήταν η

υποστήριξη και των REST Web Services. Τα κύρια στοιχεία της αρχιτεκτονικής θα ήταν ίδια (ACM, KrbClient, KDC, Policy Repository). Ωστόσο, μια τέτοια προσέγγιση θα απαιτούσε αλλαγές στον κώδικα του πελάτη (KrbClient) και δημιουργία ενός διαφορετικού ACM, ο οποίος θα λειτουργούσε σαν http handler. Το λογισμικό του πελάτη μπορεί να ορίσει στατικά ή ακόμα και δυναμικά (εφόσον υπάρχουν οι απαραίτητοι μηχανισμοί ανακάλυψης του τύπου της υπηρεσίας) αν η υπηρεσία που καλείται είναι τύπου SOAP ή REST. Ο KrbClient θα εκτελούσε τον αντίστοιχο κώδικα που αντιστοιχεί στον τύπου του ACM (SOAP ή REST) για την περαιτέρω επικοινωνία. Πάντως να τονίσουμε ότι ένα πιθανό μειονέκτημα των RESTful Web Services είναι η υποστήριξη ασφάλειας point-to-point και όχι end-to-end, όπως είδαμε στην ενότητα 3.5.1.

- Η αρχιτεκτονική ασφαλείας που προτείνουμε θα μπορούσε να ενσωματωθεί σε υπηρεσίες Instrumentation Grid υλοποιημένες με μοντέρνες τεχνικές του Software Engineering, όπως είναι ένα Enterprise Service Bus (ESB) [ESB]. Τα ESBs προσφέρουν μια αρχιτεκτονική δομή που δρα ως ενδιάμεσος (middleware) μεταξύ των διαφόρων συστημάτων και λειτουργεί ως δίαυλος (bus) επικοινωνίας μεταξύ τους, υποστηρίζοντας τα πρότυπα πρωτόκολλα messaging (SOAP, REST, RMI κτλ.) και τους μετασχηματισμούς μεταξύ τους. Τα ESBs επιτρέπουν την δημιουργία υπηρεσιών SOA. Η αρχιτεκτονική μας θα μπορούσε να ενσωματωθεί ως ένα στοιχείο του ESB πχ. χρησιμοποιώντας τεχνολογίες όπως το Java Business Integration [JBI]. Το JBI θα μπορούσε να προσφέρει μια υλοποίηση του ACM που να μπορεί να ενσωματωθεί σε διαφορετικές υλοποιήσεις του ESB, όπως το Apache ServiceMix [SERMIX] και το OpenESB [OESB].

8. Βιβλιογραφία

[Adam97] C.M. Adams, "Constructing Symmetric Ciphers Using the CAST Design Procedure", Designs, Codes, and Cryptography, 12(3), pp. 283–316, 1997

[Adam99] C. Adams, S. Farrell, "Internet X.509 Public Key Infrastructure: Certificate Management Protocols", IETF RFC 2510, March 1999

[Agra07] A. Agrawal, et al, “WS-BPEL Extension for People (BPEL4People), Version 1.0”, proposed by (Active Endpoints Inc., Adobe Systems Inc., BEA Systems Inc., International Business Machines Corporation, Oracle Inc., and SAP AG), June 2007

[Akki05] R. Akkiraju, et al, “Web Services Semantics – WSDL-S”, W3C Member Submission, 7 November 2005

[Alfi03] R. Alfieri et al, “VOMS, an Authorization System for Virtual Organizations”, Presented at the 1st European Across Grids Conf., Santiago de Compostela, Spain, Feb. 14, 2003

[Alfi05] R. Alfieri, R Cecchini, V. Ciaschini, F. Spataro, L. Dell'Agnello, A. Frohner, K. Lorentey, “From gridmap-file to VOMS: managing authorization in a Grid environment”, Future Generation Computer Systems. Vol. 21, no. 4, pp. 549-558. Apr. 2005

[Allc03] W. Allcock, editor, “GridFTP Protocol Specification”, Global Grid Forum Recommendation GFD.20, March 2003.

[Andr09] S. Andreozzi, S. Burke, F. Ehm, L. Field, G. Galang, B. Konya, M. Litmaath, P. Millar, JP Navarro, “Glue Specification v.2.0”, Open Grid Forum Specification, March 3 2009

[ATHENA] The Athena Project - http://ist.mit.edu/services/athena

[Avel04] G. Avellino, S. Beco, B. Cantalupo, A. Maraschini, F. Pacini, M. Sottilaro, A. Terracina, D. Colling, F. Giacomini and E. Ronchieri, et al, “The DataGrid Workload Management System: Challenges and Results”, Journal of Grid Computing, Volume 2, Number 4, 353-367, 2004

[AXIS] Apache Axis - http://ws.apache.org/axis/

[AXIS2] Apache Axis 2, Retreived on 26 November 2010 - http://axis.apache.org/axis2/java/core/

[Bake02] P. Hallam-Baker, E. Maler et al, “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)”, OASIS Standard, 5 November 2002

[Ball04] K. Ballinger, D. Ehnebuske, C. Ferris, M. Gudgin, C. Kevin, M. Nottingham, P. Yendluri, “WS-I Basic Profile Version 1.1”, Web Services Interoperability Organization Standard, 24 August 2004

[Bank06] T. Banks, “Web Services Resource Framework (WSRF) – Primer v1.2”, Oasis Committee Draft 02, 23 May 2006

[Bart08] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, E. Simon, “XML Signature Syntax and Processing (Second Edition)”, W3C Recommendation, 10 June 2008

[Baud03] JP Baud, B. Couturier, C. Curran, JD Durand, E. Knezo, S. Occhetti, O. Barring, “CASTOR status and evolution”, CoRR, cs.oh/0305047, informal publication, 28 May 2003

[BONCA] Bouncy Castle Crypto API - http://www.bouncycastle.org/

[Boot04] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard, “Web Services Architecture”, W3C Working Group Note 11, February 2004

[Boye08] J. Boyer, G. Marcy, “Canonical XML Version 1.1”, W3C Recommendation, 2 May 2008

[Brai06] J. Brainard, A. Juels, R. L. Rivest, M. Szydlo, M. Yung, “Fourth-factor authentication: somebody you know”, In Proceedings of the 13th ACM Conference on Computer and Communications Security (Alexandria, Virginia, USA, October 30 - November 03, 2006). CCS '06. ACM, New York, NY, 168-178. DOI= http://doi.acm.org/10.1145/1180405.1180427

[Butl00] Butler, R., Engert, D., Foster, I., Kesselman, C., Tuecke, S., Volmer, J. and Welch, V. Design and Deployment of a National-Scale Authentication Infrastructure. IEEE Computer, 33(12):60-66. 2000

[Case90] J. Case, M. Fedor, M. Schoffstall, J. Davin, “A Simple Network Management Protocol (SNMP)”, IETF RFC 1157, May 1990

[CCMP] Counter Mode with Cipher Block Chaining Message Authentication Code Protocol - http://en.wikipedia.org/wiki/CCMP

[Chap06] D. Chappell, L. Liu, “Web Services Brokered Notifications 1.3 (WS-BrokeredNotification)”, OASIS Standard, 1 October 2006

[Chin03] R. Chinnici, M. Gudgin, J-J Moreau, S. Weerawarana, “ Web Services Description Language (WSDL 1.2)”, W3C Working Draft, 11 June 2003

[Chiu02] Kenneth Chiu, Madhusudhan Govindaraju, Randall Bramley, "Investigating the Limits of SOAP Performance for Scientific Computing," hpdc, pp.246, 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11 '02), 2002

[Chri01] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, “Web Services Description Language (WSDL) 1.1”, W3C Note, March 2001

[Chris01] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, “Web Services Description Language (WSDL 1.1)”, W3C Note, 14 March 2001

[Cias06] V. Ciaschini, V. Venturi, A. Ceccanti, “The VOMS Attribute Certificate Format”, OGF Draft, 2006

[Clem04] L. Clement, A. Hately, C. Riegen, T. Rogers, “UDDI Version 3.0.2”, UDDI Spec Technical Committee Draft, 2004

[CMS] Compact Muon Solenoid - http://cms.web.cern.ch/cms/index.html

[CMS08] “The CMS experiment at the CERN LHC”, Chapter 9, Published by the Institute of Physics Publishing and SISSA, August 2008 - http://www.iop.org/EJ/article/1748-0221/3/08/S08004/jinst8_08_s08004.pdf

[CONDOR] Condor High Throughput Computing - http://www.cs.wisc.edu/condor/

[CORBA] Common Object Resource Broker Architecture - http://www.omg.org/spec/CORBA/3.1/

[CruxToolkit] Crux Toolkit - http://confluence.globus.org/display/whi/Crux+Toolkit+-

[CVS] Concurrent Versions System - http://cvs.nongnu.org/

[CXF] Apache CXF, Retrieved 26 November 2010 - http://cxf.apache.org/

[Daem02] Joan Daemen, Vincent Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard." Springer, 2002.

[Daem99] J. Daemen and V. Rijmen, AES proposal: Rijndael (1999)

[DCOM] Distributed Component Object Model (DCOM) Remote Protocol Specification - http://msdn.microsoft.com/library/cc201989.aspx

[DELEG] Delegation Protocol http://www.gridsite.org/wiki/Delegation_protocol

[Dier06] T. Dierks, E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.1 (RFC 4346)”, IETF RFC, 2006

[Diff76a] Whitfield Diffie and Martin Hellman, "Multi-user cryptographic techniques", AFIPS Proceedings 45, pp109–112, 8 June 1976

[Diff76b] W. Diffie, M. Hellman, “New Directions in Cryptography”, IEEE Transactions on Information Theory, Vol.22, No.6 (November 1976), p.644.

[DORI] Deployment or Remote Instrumentation Infrastructure – DORII - http://www.dorii.eu/

[ECLIPSE] Eclipse IDE - http://www.eclipse.org/

[EGEE] Enabling Grids for E-sciencE – http://www.eu-egee.org

[EGEE10] EGEE Summary - http://www.eu-egee.org/fileadmin/documents/publications/EGEEIII_Publishable_summary.pdf

[EGI] EGI - http://www.egi.eu/

[ELLETRA] Synchrotron Radiation Sotorage Ring Elettra - http://www.elettra.trieste.it/index.php

[ESB] Enterprise Service Bus, Retrieved 13 January 2011 - http://en.wikipedia.org/wiki/Enterprise_service_bus

[Este09] J. Estefan, K. Laskey, F. McCabe, D. Thornton, “Reference Architecture Foundation for Service Oriented Architecture, Version 1.0”, Committee Draft 02, 14 October 2009

[EUGridPMA] The EUGridPMA - coordinating grid authentication in e-Science - http://www.eugridpma.org/

[Fall04] D. C. Fallside, P. Walmsley, “XML Schema Part 0: Primer Second Edition”, W3C Recommendation, 28 October 2004

[Ferg10] N. Ferguson, B. Schneier, T. Kohno, “Cryptography Engineering”, Book by Wiley Publishing, 2010

[Ferr04] C. Ferris, A. Karmarkar, C. K. Liu, “Attachment Profile 1.1”, Web Services Interoperability (WS-I) Final Material, 24 August 2004

[Fiel00] R. T. Fielding, “Representational state transfer (REST)”, Chapter 5 in Architectural Styles and the Design of Network-based Software Architectures, Ph.D. Thesis, University of California, Irvine, CA, 2000.

[Fiel99] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach, Berners-Lee, “RFC 2616: Hypertext Transfer Protocol – HTTP/1.1”, IETF Standard, June 1999

[FIPS180-2] “Secure Hash Standard (SHS)”, Federal Information Processing Standards Publication 180-2, 1 August 2002

[FIPS197] “Advanced Encryption Standard (AES)”, Federal Information Processing Standards Publication 197, 26 November 2001

[FIPS46-3] “Data Encryption Standard (DES)”, Federal Information Processing Standards Publication 46-3, 25 October 1999

[Fluh01] S. Fluhrer, I. Mantin, A. Shamir, “Weaknesses in the Key Scheduling Algorithm of RC4”, Lecture Notes in Computer Science, 2001, Volume 2259/2001, 1-24

[Fost01] I. Foster, C. Kesselman, S. Tuecke: “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, International J. Supercomputer Applications, 15(3), 2001

[Fost02a] I. Foster, C. Kesselman, J.M. Nick, S. Tuecker, "The Physiology of the Grid", white paper, 2002

[Fost02b] Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke, "Grid Services for Distributed System Integration," Computer, vol. 35, no. 6, pp. 37-46, June 2002, doi:10.1109/MC.2002.1009167

[Fost02c] I. Foster, "What is the Grid? A Three Point Checklist." 2002, http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf.

[Fost05] I. Foster, “A Globus Toolkit Primer”, 2005

[Fost06a] I. Foster, “Globus Toolkit Version 4: Software for Service-

Oriented Systems", IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006

[Fost06b] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramanian, J. Treadwell, J. Von Reich, "Open Grid Services Architecture, Version 1.5 (OGSA)", GGF International Document, 24 July 2006, http://www.ogf.org/documents/GFD.80.pdf

[Fost97] I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", Intl J. Supercomputer Applications, 11(2):115-128, 1997.

[Fost99] I. Foster, C. Kesselman. Chapter 2 of "The Grid: Blueprint for a New Computing Infrastructure", Morgan-Kaufman, 1999

[Friz06] E. Frizziero, M. Gulmini, F. Lelli, G. Maron, A. Oh, S. Orlando, A. Petrucci, S. Squizzato, S. Traldi, "Instrument Element: A New Grid component that Enables the Control of Remote Instrumentation," ccgrid, p. 52, Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06), 2006

[GANG] Ganga - http://ganga.web.cern.ch/ganga/

[Ghaz05] Nayef Abu-Ghazaleh, Michael J. Lewis, "Differential Deserialization for Optimized SOAP Performance," sc, pp.21, Proceedings of the 2005 ACM/IEEE conference on Supercomputing, 2005

[gLite] gLite Grid Middleware Project - http://glite.web.cern.ch/glite/

[GLOBUS] Globus Toolkit - http://www.globus.org/toolkit/

[GlobusAlliance] Globus Alliance - http://www.globus.org/

[Grah06a] S. Graham, D. Hull, B. Murray, "Web Services Base Notification 1.3 (WS-BaseNotification), OASIS Standard, 1 October 2006

[Grah06b] S. Graham, A. Karmarkar, J. Mischkinsky, I. Robinson, I. Sedukhin, "Web Services Resource 1.2 (WS-Resource)", OASIS Standard, 1 April 2006

[GRIDCC] GRIDCC Project – http://en.wikipedia.org/wiki/GridCC

[Grim06] A. Grimshaw, D. Snelling, "WS-Naming Specification" GGF Recommentation (GFD-R-P.109), March 2006

[GriPhyN] Grid Physics Network - http://www.griphyn.org

[GSPH] GridSphere Portal Framework - http://www.gridsphere.org/gridsphere/gridsphere

[Gudg06] M. Gudgin, M. Hadley, T. Rogers, "Web Services Addressing 1.0 – Core (WS-Addressing)", W3C Recommendation, 9 May 2006.

[Gudg07] M. Gudgin, et al, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", W3C Recommendation, 27 April 2007

[Gutm02] Peter Gutman, "PKI: It's Not Dead, Just Resting", Computer, v.35 n.8, p.41-49, August 2002, doi>10.1109/MC.2002.1023787

[Hass04] H. Hass, A. Brown, "Web Services Glossary", W3C Working Group Note 11, February 2004

[HEIMDAL] The Heimdal Kerberos: http://www.pdc.kth.se/heimdal/

[Hous02] R. Housley, W. Polk, W. Ford, D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", IETF RFC3280, April 2002

[Hugh05] J. Hughes et al, "Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0" OASIS Standard, March 2005

[ICXF] Apache CXF Interceptors, Retrieved 26 November 2010 - http://cxf.apache.org/docs/interceptors.html

[Imam02] T. Imamura, B. Dillaway, E. Simon, "XML Encryption Syntax and Processing", W3C Recommendation, 10 December 2002

[Iosu04] A.Iosup, N.Ţăpuş, S.Vialle, "A Monitoring Architecture for Control Grids", proc. Advances in Grid Computing - EGC 2005, LNCS, vol.3470/2005,pp.922-931, July 11, 2005

[Irvi04] M. Irving, G. Taylor, P. Hobson, "Plug in to grid computing", IEEE Power and Energy Magazine 2(2): 40-44p, Mar 2004

[ITU05] "X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", http://www.itu.int/rec/T-REC-X.509, 2005

[JBI] "JSR 312: Java Business Integration 2.0 (JBI 2.0)", Retrieved 14 January 2011 - http://jcp.org/en/jsr/detail?id=312

[JCE02] "Java Cryptography Architecture API Specification & Reference", 4 August 2002 -

http://java.sun.com/j2se/1.4.2/docs/guide/security/CryptoSpec.html

[JCR168] "JSR-000168 Portlet Specification" - http://jcp.org/aboutJava/communityprocess/review/jsr168/

[Kent96] S. Kent, R. Atkinson, "IPSec, Security Architecture for the Internet Protocol (RFC 2401)", IETF RFC, 1996

[Kiss09] B. Kissel, "OpenID 2009 Year in Review", 16 December 2009, retrieved from http://openid.net/2009/12/16/openid-2009-year-in-review/

[Kivi03] T. Kivinen, M. Kojo, "More Modular Exponential (MOPD) Diffie-Hellman groups for Internet Key Exchange (IKE)", IETF RFC 3526, May 2003

[Klen08] J. Klensin, "RFC 5123: Simple Mail Transfer Protocol", IETF Standard, October 2008

[Kohl93] J. Kohl, C. Neuman, "The Kerberos Network Authentication Service (V5)", IETF RFC 1510, September 1993

[Kohn78] L. Kohnfelder, "Towards a Practical Public-key Cryptosystem", MIT Bachelor's thesis, May 1978, available from http://theses.mit.edu/Dienst/UI/2.0/Composite/0018.mit.theses/1978-29/1

[Kreg09] H. Kreger, J. Estefan, "Navigating the SOA Open Standards Landscape Around Architecture", July 2009

[Lell07] Francesco Lelli, Eric Frizziero, Michele Gulmini, Gaetano Maron, Salvatore Orlando, Andrea Petrucci and Silvano Squizzato, "The many faces of the integration of instruments and the grid", International Journal of Web and Grid Services 2007 - Vol. 3, No.3 pp. 239 – 266

[LHC] Large Hardon Collider - http://lhc.web.cern.ch/lhc/

[LHCD] Worldwide LHC Computing Grid - http://public.web.cern.ch/public/en/lhc/Computing-en.html

[Linz09] AD Linz, "Instument Element", Presentation at the DORII Summer School, St. Stefan, Austria, September 9 2009

[Litz88] Michael Litzkow, Miron Livny, and Matt Mutka, "Condor - A Hunter of Idle Workstations", Proceedings of the 8th International Conference of Distributed Computing Systems, pages 104-111, June, 1988

[LSF] Platform LSF - http://www.platform.com/workload-management/high-performance-computing/lp

[MacK06] M. MacKenzie, K. Laskey, F. McCabe, P. Brown, R. Metz, B. Hamilton, “Reference Model for Service Oriented Architecture 1.0”, OASIS Standard, 12 October 2006

[MDS] GT Information Services: Monitoring & Discovery System (MDS) - http://www.globus.org/toolkit/mds/

[Mene97] A. Menezes, P. Van Oorschot, S. Vanstone, “Handbook of Applied Cryptography”, CRC Press Series on Discrete Mathematics and Its Applications, 1997

[Merk78] R. C. Merkle, "Secure Communications over Insecure Channels", Communications of the ACM 21 (4): 294–299, April 1978

[Mill87] S. P. Miller, B. C. Neuman, J. I. Schiller, J. H. Saltzer, “Section E.2.1: Kerberos Authentication and Authorization System”, M.I.T. Project Athena, Cambridge, Massachusetts, December 21, 1987

[Mite04] M. Mineter, “Grid developments and middleware components”, Presentation in The 2nd International Summer School on Grid Computing, Vico Equense, Italy, 2004

[MITKERB] Massachusetts Institute of Technology (MIT) Kerberos Implementation - http://web.mit.edu/kerberos/

[Mitr07] N. Mitra, Y. Lafon, “SOAP Version 1.2 Part 0: Primer (Second Edition)”, W3C Recommendation, 27 April 2007

[Monz06] R. Monzillo, C. Kaler, A. Nadalin, P. Hallem-Baker, “WS-SAML Token Profile 1.1”, OASIS Standard Specification, 1 February 2006

[MSAC] Active Directory, Retrieved 14 January 2011 - http://en.wikipedia.org/wiki/Active_Directory

[MYSQL] MySQL - http://www.mysql.com/

[Nada06a] A. Nadalin, C. Kaler, R. Monzillo, P. Hallam-Baker, “Web Services Security: SOAP Message Security 1.1 (WS-Security 2004”, OASIS Standard Specification, 1 February 2006

[Nada06b] A. Nadalin, C. Kaler, R. Monzillo, P. Hallam-Baker, “WS-Security X509 Token Profile 1.1”. OASIS Standard Specification, 1 February 2006

[Nada06c] A. Nadalin, C. Kaler, R. Monzillo, P. Hallam-Baker, “WS-Security Kerberos Token Profile 1.1”, OASIS Standard Specification, 1 February 2006

[Nada07] A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, H. Granqvist, “WS-Trust 1.3” OASIS Standard, 19 March 2007

[Nada09] A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, H. Granvist, “WS-SecureConverasation 1.4”, OASIS Standard, 2 February 2009

[Naga08] N. Nagaratnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster, S. Tuecke, “The Security Architecture for Open Grid Services”, OGSA-SEC-WG Document

[Neal03] M. O’ Neal, “We Know Web Services Need Security, But What Type?”, Web Services Journal, 3(3), 18-27, 2003

[Need78] R. Needham, M. Schroeder, “Using encryption for authentication in large networks of computers.”, Communications of the ACM 21 (12): 993–999. December 1978, doi:10.1145/359657.359659

[NEES] Network for Earthquake Engineering and Simulation – http://www.nees.org

[Nefe06] V. Nefedova, R. Jacob, I. Foster, Z. Liu, Y. Liu, E. Deelman, G Mehta, M. Su, K. Vahi. “Automating Climate Science: Large Ensemble Simulations on the TeraGrid with the GriPhyN Virtual Data System”, Presented at the eScience Conference in Amsterdam, December, 2006

[Neum05] B. C. Neuman, T. Yu, S. Hartman, K. Raeburn, “The Kerberos Network Authentication Service (V5)”, IETF RFC 4120, July 2005

[Neum05] C. Neuman, T. Yu, S. Hartman, K. Raeburn, “The Kerberos Network Authentication Service (V5)”, IETF RFC 4120, July 2005

[Neum94] B. C. Neuman, T. Ts'o, “Kerberos: An Authentication Service for Computer Networks”, IEEE Communications 32 (9): 33–8, September 1994, doi:10.1109/35.312841

[OASIS] The Oasis Consortium - http://www.oasis-open.org/

[OESB] Java Open Enterprise Service Bus, Retrieved 14 January 2011 - http://java.net/projects/open-esb

[OGF] Open Grid Forum – http://www.gridforum.org

[OGSOA] Open Group - http://opengroup.org/projects/soa/doc.tpl?gdid=10632

[OIDMEMBER] OpenID Sponsoring Members - http://openid.net/foundation/sponsoring-members/

[OMG] Object Management Group – http://www.omg.org

[OPENGRO] Opent Group - http://opengroup.org

[OpenID] OpenID Foundation - http://openid.net/

[OSA2010] OSA, Definition of Security Architecture, Visited on September 21 2010, http://www.opensecurityarchitecture.org/cms/definitions/it-security-architecture

[Park98] D. B. Parker, "Fighting Computer Crime", New York, NY: John Wiley & Sons, 1998

[PBS] Open PBS - http://www.openpbs.org/

[Pear02] L. Pearlman, V. Welch, I. Foster, K. Kesselman and S. Tuecke, "A Community Authorization Service for Group Collaboration", IEEE Workshop on Policies for Distributed Systems and Networks, 2002

[Perl99] R. Perlman, "Overview of PKI trust models", IEEE NETWORK. Vol. 13, no. 6, pp. 38-43. 1999, doi>10.1109/65.806987

[Post82] J. B. Postel, "RFC 821: Simple Mail Transfer Protocol", IETF Standard, August 1982

[Post85] J. Postel, J Reynolds, "RFC 959: File Transfer Protocol", IETF Standard, October 1985

[PrCert] Globus Security Certificate Types - http://dev.globus.org/wiki/Security/ProxyCertTypes, Retrieved 22/10/2010

[Rago08] N. Ragouzis et al., "Security Assertion Markup Language (SAML) V2.0 Technical Overview", OASIS Committee Draft, March 2008, Document ID sstc-saml-tech-overview-2.0-cd-02

[Rive78] R. L. Rivest, A. Shamir, L Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Commun. ACM 21, 2 (Feb. 1978), 120-126. DOI= http://doi.acm.org/10.1145/359340.359342

[Rive87] Ron Rivest, "RC4", Designed 1987, leaked 1994 -

https://secure.wikimedia.org/wikipedia/en/wiki/RC4

[Rive92a] R. Rivest, “The MD4 Message Digest Algorithm”, IETF RFC 1186, April 1992

[Rive92b] R. Rivest, “The MD5 Message Digest Algorithm, IETF RFC 1321, April 1992

[Rive94] R. L. Rivest, "The RC5 Encryption Algorithm", Proceedings of the Second International Workshop on Fast Software Encryption (FSE) 1994e. pp. 86–96.

[Rose01] M. Rose, “RFC 3080: The Blocks Extensible Exchange Protocol Core”, IETF Standard, March 2001

[RSAfaq] “2.1.7 What are Message Authentication Codes?”, RSA Labs, http://www.rsa.com/rsalabs/node.asp?id=2177

[RSAPerf] RSA Security, “How fast is the RSA Algorithm”, http://www.rsa.com/rsalabs/node.asp?id=2215, Retrieved October 13, 2010

[SCAD] Supervisory Control And Data Acquisition SCADA - http://en.wikipedia.org/wiki/SCADA

[Scav05] T. Scavo, S. Cantor, N. Dors, “Shibboleth Architecture, Technical Overview”, Shibboleth Project, Working Draft 02, 8 June 2005

[Schn94] B. Scheier, “Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)”, Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993), Springer-Verlag, 1994, pp. 191-204

[SERMIX] Apache ServiceMix 4, Retrieved 14 January 2011 - http://servicemix.apache.org/home.html

[Sing07] A. Singhal, T. Winograd, K. Scarfone, “Guide to Secure Web Services” NIST, Special Publication 800-95. Retrieved September 18, 2008, from http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf, 2007

[Slom94] M. Sloman, “Policy Driven Management for Distributed Systems”, Journal of Network and Systems Management, 2 (4):333–360, Plenum Press, 1994

[Snel04] Snelling, D. F., Van den Berghe, S. and Li, V. Q. (2004). Explicit Trust Delegation: Security for the Dynamic Grids, FUJITSU Sci. Tech. J.

[SPNEGO] SPNEGO - Simple and Protected GSSAPI Negotiation Mechanism, Retrieved 14 January 2011, http://en.wikipedia.org/wiki/SPNEGO

[Stap06] G. Staples, “TORQUE resource manager”, In Proceedings of the 2006 ACM/IEEE Conference on Supercomputing (Tampa, Florida, November 11 - 17, 2006). SC '06. ACM, New York,

[Stew07] GA Stewart, D. Cameron, GA Cowan, G. McCance, “Storage and data management in EGEE”, In Proceedings of the fifth Australasian symposium on ACSW frontiers - Volume 68 (ACSW '07), 2007

[Sura02] S. Suratti, M. Muckin, “HTTP-Based Cross-Platform Authentication by Using the Negotiate Protocol”, Microsoft Consulting Services, December 2002, Retrieved 14 January 2011 - http://msdn.microsoft.com/en-us/library/ms995329

[TOMCAT] Apache Tomcat - http://tomcat.apache.org/

[Tuec04] S. Tuecke, V Welch, D. Engert, L. Pearlman, M. Tompson, “Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile”, IETF RFC 3820, June 2004

[Vamb06] W. Vambenepe, S. Graham, P. Niblett, “Web Services Topics 1.3 (WS-Topics)”, OASIS Standard, 1 October 2006

[Veda07a] A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, U. Yalcinalp, “Web Services Policy 1.5 – Framework”, W3C Recommendation, 4 September 2007

[Veda07b] A. S. Vedamuthu, D. Orchard, F. Hirsch, M. Hondo, P. Yendluri, T. Boubez, U. Yalcinalp, “Web Services Policy 1.5 – Attachment”, W3C Recommendation, 4 September 2007

[VeriSign] VeriSign - http://www.verisign.com/

[Vlis99] John Vlessides, “An Introduction to Design Patterns”, IBM T. J. Watson Research Presentation, 1999 - John Vlissides, http://www.research.ibm.com/designpatterns/pubs/dp-tutorial.pdf

[W3C] World Wide Web Consortium (W3C) – http://www.w3c.org

[Wang04a] X. Wang, C. Wu, X. Hu, L. Liu, J. Z. Jisuanji, “Study of equipment grid based on simulation and modeling”, Computer Integrated Manufacturing System (China). Vol. 10, no. 9, pp. 1031-1035. Sept. 2004

[Wang04b] X. Wang, D. Feng, X. Lai, H. Yu, “Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD”,

Cryptology ePrint Archive, Report 2004/199, 17 August 2004

[Wang04c] Y Wang, L Liu, C Wu, W Ni, "Research on equipment resource scheduling in grids", Lecture Notes in computer science, 2004

[Welc04] V. Welch, I. Foster, C. Kesselman, O. Mulmo, L. Pearlman, S. Teucke, J. Gawor, S. Meder, F. Siebenlist, «X.509 proxy certificate for dynamic delegation», Proceeding of the 3rd Annual PKI R&D Workshop, 2004

[WEP] Wired Equivalent Privacy (WEP) - http://en.wikipedia.org/wiki/Wired_Equivalent_Privacy

[West01] A. Westerinen et al, "Terminology for Policy-Based Management", IETF RFC 3198, November 2001

[Woll96] A. Wollrath, R. Riggs, J. Waldo, "A Distributed Object Model for the Java System", Proceedings of the USENIX 1996 Conf. on Object-Oriented Technologies (COOTS), pp. 219-231, 1996

[WPA2] WPA2® (Wi-Fi Protected Access® 2) - http://www.wi-fi.org/knowledge_center/wpa2

[Wray00] J. Wray, "Generic Security Service API Version 2 : C-bindings", IETF RFC 2743, January 2000

[WS-I] Web Services Interoperability Organization – http://www.ws-i.org

[WSDL2.0] "WSDL 2.0", W3C Web Services Working Group - http://www.w3.org/2002/ws/desc/

[WSS4J] Apache WSS4J - http://ws.apache.org/wss4j/

[X.500] "The website of the X.500 Directory standard" - http://www.x500standard.com/

[XML] XML - http://www.w3.org/XML/

[Xuej91] L. Xuejia Lai, J. L. Massey, S. Murphy, "Markov ciphers and differential cryptanalysis", Advances in Cryptology — Eurocrypt '91, Springer-Verlag (1992), pp17–38.

[Zhan07] J. Zhang, "Position Paper for W3C Workshop on Next Steps for XML Signature and XML Encryption", 2007, Retrieved November 3, 2010, from http://www.w3.org/2007/xmlsec/ws/papers/06-zhang-ximpleware/

[Zhu05] L. Zhu, K. Jaganathan, S. Hartman, "The Kerberos Version 5

Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2", IETF RFC 4121, July 2005

[Zhu06] L. Zhu, B. Tung, "Public Key Cryptographyfo Initial Authentication in Kerberos (PKINIT)", IETF RFC 4556, June 2006