ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών
Τηλεπικοινωνιών και Δικτύων

ΘΕΜΑΤΑ ΑΣΦΑΛΕΙΑΣ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗΣ ΣΕ ΣΥΓΧΡΟΝΑ ΤΗΛΕΠΙΚΟΙΝΩΝΙΑΚΑ ΔΙΚΤΥΑ

Δημήτριος Ζησιάδης

Μια εργασία που εκπονήθηκε για τις απαιτήσεις
του Διδακτορικού Διπλώματος

Επιβλέπων Καθηγητής: Λέανδρος Τασιούλας

Πανεπιστήμιο Θεσσαλίας
Βόλος, 2010

Εισαγωγή

Η διαχείριση της ασφάλειας των προσωπικών επικοινωνιών είναι ένα ιδιαίτερα σημαντικό θέμα, τόσο για τις επικοινωνίες σημαντικών προσώπων (επιχειρηματίες, κυβερνητικά στελέχη, οικονομικοί παράγοντες κλπ) και οργανισμών (κρατικών ή ιδιωτικών), όσο και για την περίπτωση των απλών πολιτών που έχουν δικαίωμα στην ιδιωτικότητα των επικοινωνιών τους και την ασφαλή μεταφορά των δεδομένων τους πάνω από κοινόχρηστα τηλεπικοινωνιακά δίκτυα και κανάλια επικοινωνίας. Το θέμα της ασφάλειας είναι ιδιαίτερα κρίσιμο σε ασύρματες υποδομές, όπως τα δίκτυα κινητής τηλεφωνίας ή αυτά που είναι βασισμένα σε τεχνολογίες WiFi ή Bluetooth, όπου οι επικοινωνίες των χρηστών χρησιμοποιούν ανοιχτά κανάλια χωρίς κάποια εχέγγυα επαρκούς ασφάλειας από το δίκτυο. Οι επιθέσεις που λαμβάνουν χώρα σε τηλεπικοινωνιακά δίκτυα είναι ποικίλες. Η πιο συνηθισμένη επίθεση είναι η καταγραφή επικοινωνιών (συνομιλιών ή ανταλλασσόμενων δεδομένων). Ένα τέτοιο παράδειγμα είχαμε στην Ελλάδα κατά την διοργάνωση των Ολυμπιακών Αγώνων του 2004, όπου ο έλεγχος ενός κεντρικού κόμβου στο τηλεπικοινωνιακό δίκτυο γνωστής εταιρίας κινητής τηλεφωνίας έδωσε τη δυνατότητα της καταγραφής σημαντικών συνομιλιών επιχειρηματιών και κυβερνητικών. Η πιο επικίνδυνη επίθεση όμως είναι αυτή όπου ο εισβολέας ξεγελά τα μέρη μιας επικοινωνίας προσποιούμενος το ένα στο άλλο και παίρνει υπό τον πλήρη έλεγχο την επικοινωνία, καταγράφοντας αλλά και μεταβάλλοντας τα μεταφερόμενα δεδομένα.

Η διατριβή αυτή ασχολείται με τη διαχείριση της ασφάλειας των προσωπικών επικοινωνιών και την προστασία των χρηστών από κακόβουλες επιθέσεις, προτείνοντας λύσεις για τη διασφάλιση του απορρήτου των προσωπικών επικοινωνιών μέσα από την ασφαλή διασύνδεση συσκευών με την ενεργή συμμετοχή του χρήστη στη διαδικασία αυτή. Η διαδεδομένη λύση για την αντιμετώπιση τέτοιων εισβολών είναι η ασφάλεια που παρέχουν οι υποδομές δημόσιου κλειδιού (PKI), όπου γίνεται χρήση ασύμμετρης κρυπτογράφησης με τη βοήθεια Έμπιστων Τρίτων Οντοτήτων. Το πρόβλημα στην περίπτωση αυτή είναι ότι οι Αρχές Πιστοποίησης αποτελούν εν δυνάμει εισβολείς είτε κακόβουλα εκ των έσω είτε λόγω πιθανής απώλειας ελέγχου στις υποδομές τους από επίθεση τρίτων. Τα χρησιμοποιούμενα τηλεπικοινωνιακά δίκτυα για την επικοινωνία με την Α.Π. αποτελούν επίσης εν δυνάμει αδύνατα σημεία για τους ίδιους λόγους. Η λύση που μπορεί να διασφαλίσει τους τελικούς χρήστες τηλεπικοινωνιακών δικτύων και υποδομών είναι η χρήση συμμετρικής κρυπτογράφησης από άκρο σε άκρο, η οποία ελέγχεται πλήρως από αυτούς. Όμως, εγκατάσταση και χρήση συστήματος συμμετρικής κρυπτογράφησης έχει με τη σειρά της κάποια μειονεκτήματα, όπως για παράδειγμα η διατήρηση/ανανέωση των κλειδιών μεταξύ των μελών της επικοινωνίας και ο συγχρονισμός των χρηστών στο τρέχον κλειδί, ενώ προϋποθέτει και την ύπαρξη ξεχωριστού καναλιού επικοινωνίας ειδικά για το σκοπό αυτό. Στην παρούσα διατριβή παρουσιάζουμε μία μέθοδο για την ασφαλή εγκαθίδρυση συμμετρικού κλειδιού μίας χρήσης, που διασφαλίζει την συγκεκριμένη προσωπική επικοινωνία, χρησιμοποιώντας το ίδιο κανάλι επικοινωνίας δεδομένων για την ανταλλαγή του κλειδιού και εμπλέκοντας το χρήστη στη διαδικασία, δίνοντάς του τον πλήρη έλεγχό της. Η μέθοδος αυτή μπορεί να χρησιμοποιηθεί για την ασφαλή διασύνδεση συσκευών του ίδιου ή διαφορετικών χρηστών και την ασφαλή ανταλλαγή δεδομένων. Η μέθοδος συνίσταται στην ανταλλαγή ενός συμμετρικού κλειδιού με τη χρήση κλειδιών ασύμμετρης κρυπτογράφησης. Η ασφαλής ανταλλαγή του συμμετρικού κλειδιού επικυρώνεται από το χρήστη ή τους χρήστες της επικοινωνίας στην αρχή της, μέσω της επιβεβαίωσης -φωνητικής ή οπτικής- κάποιων ειδικών αντικειμένων ελέγχου που ανταλλάσσονται ειδικά για το σκοπό αυτό με τη χρήση των ίδιων κλειδιών ασύμμετρης κρυπτογράφησης. Η μέθοδος εκτελείται στην αρχή κάθε επικοινωνίας και το αποτέλεσμα είναι είτε η ασφαλής εγκαθίδρυση του συμμετρικού κλειδιού κρυπτογράφησης της συγκεκριμένης επικοινωνίας ή η ενημέρωση του χρήστη ότι το κανάλι έχει δεχθεί επίθεση από εισβολέα.

Συνεργαζόμενοι οργανισμοί διασυνδέουν τις υποδομές τους για να δημιουργήσουν μεγαλύτερες δικτυακές οντότητες, τα δίκτυα πλέγματος (Grid Networks) για να καλύψουν τις ερευνητικές ή αναπτυξιακές ανάγκες των χρηστών τους. Τα δίκτυα πλέγματος αποτελούν μία ειδική κατηγορία συνεργαζόμενων δικτύων για την παροχή ενός συνόλου υπηρεσιών από όλα τα συνεργαζόμενα δίκτυα που το αποτελούν, επιπρόσθετα με τις υπηρεσίες που το καθένα αυτόνομα και αποκλειστικά παρέχει στους χρήστες του. Στα δίκτυα αυτού του τύπου πέρα από τα τοπικά Κέντρα Λειτουργιών Δικτύου (ΚΛΔ) του κάθε δικτύου εγκαθίσταται και ένα κεντρικό ΚΛΔ το οποίο χρησιμοποιείται για την επίβλεψη και το συντονισμό του σχηματιζόμενου δικτύου υπηρεσιών πάνω από τις φυσικές υποδομές των δικτύων. Η παρακολούθηση των δικτύων και η αποτελεσματική διαχείρισή τους αποτελούν κύριο μέλημα των οντοτήτων που τα λειτουργούν. Η καταγραφή των προβλημάτων που εμφανίζονται στη λειτουργία τους και η ενημέρωση των αρμόδιων για την ανάληψη δράσης προσώπων και οντοτήτων ώστε να λυθεί το πρόβλημα άμεσα και διασφαλίζοντας την όσο το δυνατό απρόσκοπτη λειτουργία τους είναι βασικό κριτήριο επιτυχούς λειτουργίας του δικτύου.

Η διατριβή αυτή ασχολείται επίσης με τη διαχείριση δικτύων πλέγματος, και συγκεκριμένα με την ομογενοποίηση της ανταλλαγής τεχνικών δελτίων αναφοράς προβλημάτων μεταξύ των δικτύων αυτών. Στην παρούσα διατριβή μας απασχόλησε ένα μείζον ζήτημα στη διαχείριση των δικτύων πλέγματος, η κοινοποίηση των τεχνικών δελτίων αναφορών προβλημάτων των Κέντρων Λειτουργιών Δικτύου στο Κεντρικό Κέντρο Λειτουργιών Δικτύου και τα άλλα Κέντρα Λειτουργιών Δικτύου των συνεργαζόμενων φορέων, ώστε να ομογενοποιηθεί η μορφή τους και να απλοποιηθεί η διαδικασία καταγραφής, επεξεργασίας και διάχυσής τους. Με αυτό τον τρόπο αυξάνεται η αποτελεσματικότητα στη διαχείριση του δικτύου, κάτι που μεταφράζεται σε ελαχιστοποίηση των χρόνων απόκρισης στα προβλήματα που παρουσιάζονται, την αποτελεσματικότερη λειτουργία των δομών υποστήριξής τους καθώς και την μείωση του χρόνου απασχόλησης του προσωπικού διαχείρισης για την επίλυση των προβλημάτων.

Η δομή της παρούσας διατριβής είναι η εξής: στο πρώτο κεφάλαιο αναφέρονται γενικές παράμετροι της διαχείρισης και πως αυτή επηρεάζει τη σχέση του χρήστη με το δίκτυο που συνδέεται. Περιγράφονται οι συνηθισμένες μέθοδοι ασφάλειας των επικοινωνιών και εξερευνούνται οι αδυναμίες τους. Στο δεύτερο κεφάλαιο παρουσιάζεται μία βασική μέθοδος με δύο παραλλαγές για τη διαχείριση του απορρήτου των προσωπικών επικοινωνιών από τους χρήστες και περιγράφεται η χρήση της για την ασφαλή διασύνδεση συσκευών που είτε βρίσκονται κοντά ή μία στην άλλη επικοινωνώντας μέσω ασύρματου καναλιού είτε διασυνδέονται πάνω από οποιουδήποτε τύπου δίκτυα επικοινωνιών (διαδίκτυο, δίκτυο κινητής τηλεφωνίας κλπ). Αναλύεται η ασφάλειά της, ορίζονται οι παραλλαγές της μεθόδου, περιγράφονται τα χαρακτηριστικά λειτουργίας τους και δίνεται μια πρωτότυπη υλοποίηση εφαρμογής που δείχνει τον τρόπο χρησιμοποίησής της για την ασφαλή διασύνδεση συσκευών. Στο τρίτο κεφάλαιο περιγράφονται οι συνηθισμένες πρακτικές διαχείρισης δικτύων πλέγματος και παρουσιάζεται ένα μοντέλο διαχείρισης ομογενοποιημένων αναφορών προβλημάτων σε δίκτυα πλέγματο με την υλοποίηση του σχήματος για την διαχείριση του δικτύου πλέγματος EGEE (Enabling Grids for E-sciencE). Στο τέταρτο κεφάλαιο δίνονται γενικά συμπεράσματα και προτάσεις για μελλοντική έρευνα. Τέλος, στο παράρτημα Α περιγράφεται αναλυτικά το μοντέλο ομογενοποίησης των αναφορών προβλημάτων, δίνεται ο ορισμός του μοντέλου δεδομένων (τύποι δεδομένων, τιμές, ιδιότητες) καθώς και η αντίστοιχη υλοποίησή του σε xml.

Οι βασικότερες συνεισφορές της παρούσας διατριβής είναι οι ακόλουθες:

- Ορίστηκε η μέθοδος ασφαλούς διασύνδεσης συσκευών με τη χρήση φωνητικής ή οπτικής επιβεβαίωσης από το χρήστη, ανάλογα με τον τύπο των χρησιμοποιούμενων συσκευών και της τηλεπικοινωνιακής υπηρεσίας που απαιτείται από τους χρήστες. Περιγράφτηκε με λεπτομέρειες ο τρόπος λειτουργίας της, τα μηνύματα που χρησιμοποιούνται και η σειρά διαδοχής τους και έγινε εκτενής ανάλυση της μεθόδου για την πιστοποίηση της ορθής και ασφαλούς λειτουργίας της.

- Αναπτύχθηκε επίσης μία πρωτότυπη εφαρμογή ασφαλούς διασύνδεσης συσκευών, η οποία λειτουργεί με οπτική επιβεβαίωση από το χρήστη (ή τους χρήστες) των συσκευών, η οποία μπορεί να υιοθετηθεί σε πληθώρα εφαρμογών προσωπικών επικοινωνιών που απαιτούν ασφαλή ανταλλαγή δεδομένων, π.χ. μεταξύ δύο φορητών υπολογιστών, PDA ή κινητών τηλεφώνων, ενός PDA ή ενός φορητού υπολογιστή με ένα εκτυπωτή ή ένα δίσκο δικτύου κλπ
- Η μέθοδος ασφαλούς διασύνδεσης συσκευών με τη χρήση φωνητικής επιβεβαίωσης από τους χρήστες έχει κατοχυρωθεί με Δίπλωμα Ευρεσιτεχνίας στον Οργανισμό Βιομηχανικής Ιδιοκτησίας.
- Καθορίστηκε ένα συμπαγές αλλά συνάμα επαρκές και αποδοτικό σύνολο πεδίων που πρέπει να περιέχονται στα ομογενοποιημένα τεχνικά δελτία αναφορών προβλημάτων (ΤΔΑΠ) για τη διαχείριση δικτύων πλέγμαυος. Τα ομογενοποιημένα ΤΔΑΠ παρέχουν όλες τις απαραίτητες πληροφορίες για την επικοινωνία του προβλήματος, τον τρόπο χειρισμού του καθώς και τις συνέπειες για το δίκτυο πλέγματος και τις υπηρεσίες που παρέχονται στους χρήστες του.
- Ορίστηκε ένα μοντέλο ΤΔΑΠ, στο οποίο μετατρέπονται όλα τα ΤΔΑΠ των συνεργαζόμενων δικτύων πριν την αποθήκευσή τους και την επεξεργασία τους. Καθορίστηκαν τα ομογενοποιημένα πεδία, οι τιμές τους και ο τρόπος καθορισμού τους. Ο ορισμός του μοντέλου έγινε σε XML, λόγω της απλότητας και ευελιξίας που παρέχει στην υλοποίηση. Το μοντέλο μπορεί να χρησιμοποιήσει δεδομένα από βάσεις δεδομένων των Κέντρων Λειτουργιών Δικτύου των διασυνδεδεμένων δικτύων, όπως π.χ. τα ονόματα των μηχανικών του δικτύου ή των τηλεπικοινωνιακών συνδέσμων και των δρομολογητών του. Το μοντέλο είναι δυναμικό και μπορεί να επεκταθεί για να συμπεριλάβει νέα δεδομένα που ένα Δίκτυο Πλέγματος μπορεί να απαιτήσει (π.χ. νέα συνεργαζόμενα δίκτυα).
- Το μοντέλο υιοθετήθηκε από το Δίκτυο Πλέγματος του προγράμματος EGEE (Enabling Grids for E-sciencE), ενώ έχει υποβληθεί σαν Internet Draft στην IEEE.

Στο παράρτημα που ακολουθεί παρέχεται λεπτομερής ανάλυση όσων αναφέρθηκαν παραπάνω.

ΠΑΡΑΡΤΗΜΑ

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

Department of Computer and Communications Engineering

Security & Network Management
In Modern Telecommunications Networks

By

Dimitris Zisiadis

Dissertation

Presented to the Faculty of the Graduate School of the
Department of Computer & Communication Engineering
in the University of Thessaly in Greece
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

University of Thessaly
Volos 2010

Supervisor	Dr Leandros Tassiulas, Professor, University of Thessaly
Co-supervisors	Dr Elias Houstis, Professor, University of Thessaly Dr Apostolos Traganitis, Professor, University of Crete
Doctoral Committee	Dr Leandros Tassiulas, Professor, University of Thessaly Dr Elias Houstis, Professor, University of Thessaly Dr Apostolos Traganitis, Professor, University of Crete Dr Catherine Houstis, Professor, University of Thessaly Dr Leonidas Georgiadis, Professor, Aristotle University of Thessaloniki Dr George Pavlou, Professor, University College London Dr Iordanis Koutsopoulos, Assistant Professor, University of Thessaly

Dedicated to my family

Acknowledgements

I would like to express my gratitude to all those who made this dissertation possible.

Security & Network Management
In Modern Telecommunications Networks

by

Dimitris Zisiadis

This thesis concerns with security management in personal communications taking place over public networks and especially secure device pairing among devices which are either co-located or interconnected over a public telecommunications network. Telecommunications security is a major issue for all communications among very important persons (businessmen, government executives, financial executives etc.) as well as for personal communications of every citizen, who have the right for privacy in their communications and security when transferring information over public networks. Security becomes an even major issue in wireless environments, like mobile networks, WiFi hotspots and Bluetooth communications, where user communications take place over insecure broadcast channels. There is a long list of possible attacks in telecommunications, the most common being eavesdropping. The most serious attack though is when a Man in the Middle (MITM) where the attacker intercepts any signalling and/or user data exchanged and has the ability to eavesdrop, inject/modify/delete any signalling or user data exchanged between them. Encryption is the only way to overcome such problems.

This thesis addresses the issue of secure key exchange over public networks and uncontrolled wireless environments in order to ensure secure device pairing through encryption of any forthcoming exchange over the channel. We present a novel method for the secure establishment of a symmetric encryption key that is used for the encryption of the relevant session, making use of the data channel for signalling purposes also. The method can be used to ensure privacy of user communications, i.e. secure data exchange or secure

device pairing. A symmetrical encryption key is established through the use of asymmetric cryptography, the integrity of which is validated visually or vocally by the participating user(s) before any communication takes place. The method is appropriate for secure device pairing and for securing privacy of telecommunications among any two parties.

Moreover this thesis concerns with Grid networks management, addressing the issue of harmonization of trouble tickets exchanged among the participating network entities that form the Grid. Whenever a problem arises in any of the participating institutions of the Grid, its Network Operations Centre (NOC) opens up a specific form of incident reporting, namely a "trouble ticket" (TT). Ideally, a uniform infrastructure should be put in place for all NOCs in order to provide services to the users of the Grid and to manage the network. In practice though, this is not the case. Unfortunately, different TT systems are used by the participating networks. There is a wide variety of commercial and open source TT systems available, with differentiated functionality among them. In addition to that, in-house developed systems are used, making it even harder to achieve interoperability among the established TT systems of the partners of the Grid. A central NOC is responsible for collecting and handling the TTs received by the participating NOCs. TT load is growing proportionally with the network size and the serviced users of the Grid, and, unfortunately, TT systems producing incidents reports follow the same trend. This leaves the central NOC with a vast amount of different types of TTs from the various NOCs that need immediate attention in order for the Grid to provide services to its users. TT normalization, i.e. transformation to a common format that is reasonable for all parties and copes with service demands in a dynamic and effective way, is of crucial importance for successful management of the Grid.

In the present work we define a data model for TT normalization for the participating institutions in a Grid. The model is designed to meet the needs of the Grid, meeting requirements of the multiple TT systems available. It is both effective and comprehensive, as it compensates for the core activities of the NOCs. It is also dynamic as it allows other options to be included in the future, according to demand.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

While in the past, technological devices, networks and telecommunications were for limited use in the hands of a few qualified specialists, they have nowadays become important tools for everyone. Users acquire knowledge and skills in using new devices and services not only through formal education and training structures but also through the use of devices and services in every aspect of human activity: social, economical, educational, etc. The result of this technological explosion is the necessity for deployment of networks and services from various agencies: private companies, academic/research institutions and also local and central governments are deploying public broadband networks, offering a variety of services to their subscribers/users through the use of numerous types of end devices available.

In the research and academic area, there is a great need for cooperation among institutions in the network and services planes, in their effort to provide to their users the necessary processing power and network infrastructures to cope with the ever increasing need for experimenting, designing, implementing and testing new ideas and technologies. Network operations management is a critical factor that ultimately affects the success of the network itself. Telecommunications and networks management covers a broad list of network operations: monitoring, traffic management, authentication, authorization, accounting, security, troubleshooting, reporting and also device control and configuration.

Wireless is a widely deployed technology offering advanced features and usability. Wireless networks are of particular interest and they have rapid growth, as these networks have the unique characteristics of freedom and mobility inherent in the pace of our times. The constant evolution of wireless technologies along with their natively social characteristics, ease of use and intuitive connectivity to them, mark their high proliferation as a means of communication. Wireless networks have become the de facto standard for connecting devices directly over open communication channels in open air locations, eliminating the need to use wireline infrastructures which are expensive, hard to deploy and manage and also inconvenient for their users. Laptops, palmtops, mobile phones and any other type of device are equipped with mobile connectivity options, such as infrared, Bluetooth and WiFi. The users are free to use any type of device, anytime, anyplace.

Openness doesn't come for free though, as it is a counterpart to security. Managing telecommunications security in these environments is a constant challenge. Communications privacy is fundamental for most people and security in any kind of telecommunications activity (voice, data, video) raises as a major concern. Especially for the most common wireless technologies like WiFi or Bluetooth, the problem is even bigger, as a plethora of software solutions are available for breaching integrity of communications carried over these types of connections, even by novice users.

This thesis concerns with secure device pairing taking place over public networks. Telecommunications security is a major issue for all communications among very important persons (businessmen, government executives, financial executives etc.) as well as for personal communications of every citizen, who have the right for privacy in their communications and security when transferring information over public networks. Security becomes an even major issue in wireless environments, like mobile networks, WiFi hotspots and Bluetooth communications, where user communications take place over insecure broadcast channels. There is a long list of possible attacks in telecommunications, the most common being eavesdropping. One of the most known paradigms is that of eavesdropping governmental and business communications before and during the Olympic Games of Athens 2004, when only one central node of a mobile network operator was compromised and intercepting software was installed on it by the attackers. The most serious attack though is when a Man in the Middle (MITM) where the attacker intercepts any signalling and/or user data exchanged and has the ability to eavesdrop, inject/modify/delete any signalling or user data exchanged between them. Encryption is the only way to overcome such problems.

This thesis addresses the issue of secure key exchange over public networks and uncontrolled wireless environments in order to encrypt any forthcoming exchange over the channel. We present a novel method for the secure establishment of a symmetric encryption key that is used for the encryption of the relevant session, making use of the data channel for signalling purposes also. The method can be used to ensure privacy of user communications, i.e. secure data exchange or secure device pairing. A symmetrical encryption key is established through the use of asymmetric cryptography, the integrity of which is validated visually or vocally by the participating user(s) before any communication takes place. The method is appropriate for secure device pairing and for securing privacy of telecommunications among any two parties

Moreover this thesis concerns with Grid networks management, addressing the issue of harmonization of trouble tickets among the Network Operations Centres of the participating institutions and the Grid's. In particular we define a data model for TT normalization for the participating institutions in a Grid. The model is designed to meet the needs of the Grid, meeting requirements of the multiple TT systems available. It is both effective and comprehensive, as it compensates for the core activities of the NOCs. It is also dynamic as it allows other options to be included in the future, according to demand. The model automates the collection and normalization of the trouble tickets produced by the independent network domains that form the Grid. Each of the participants is using its home trouble ticket system within its domain for handling trouble incidents, whereas the central Network Operation Centre is gathering the tickets in the normalized format for repository and handling. Our approach is using XML as the common representation language. The model was adopted and used as part of the EGEE project. It has been also posted as an IEEE Internet draft.

The structure of the thesis is as follows: in the rest of this chapter we present the basic threats in telecommunications as well as cryptography solutions to the problem. In chapter 2 we present our method for user based secure device pairing. We define two variations, one for secure device pairing for devices closely located and the other when users are apart from each other and their devices are connected over public networks. We prove its security though extensive security analysis and we present a prototype implementation for secure device pairing. In chapter 3 we present a trouble ticket harmonization system and its respective implementation for managing the EGEE (Enabling Grids for E-sciencE) Grid network. Conclusions of our work are provided in chapter 4. Finally, in Appendix A we present the network trouble ticket data model for trouble ticket harmonisation in a Grid network environment. We define the data model, its data types and its components and we provide an xml sample implementation of the model.

1.1. Types of threats – MITM

There are many kind of threats that current networks face, especially those using wireless technology. The most important ones for voice communications are:

- **Eavesdropping:** the attacker eavesdrops on users' signalling and data connections compromising confidentiality. In open Internet infrastructures sniffing is a very common method to obtain unencrypted data. There is a variety of sniffing applications like tcpdump [1], ethereal [2], sniff-em [3].When unencrypted data travel over open multiple access links are subject to sniffing attacks. The widely spread WiFi hotspot environments is a typical example. In many wireless hotspots there is no security whereas in many others minimum security is enabled, like WEP, where an attack can easily breach security. In August 2001, Scott Fluhrer et al. published a cryptanalysis of WEP [4] that exploits the way the RC4 cipher and IV is used in WEP, resulting in a passive attack that can recover the RC4 key after eavesdropping on the network (depending on the network traffic (the number of packets you can inspect) the length could be from 10 minutes to indefinitely (if there is no data being sent at all)). There are also ways to force the traffic onto the network which is rejected, but packets are sent and thus can also be inspected to find the key. The attack was soon implemented, and automated tools have since been released. It is possible to perform the attack with a personal computer, off-the-shelf hardware and freely-available software such as aircrack-ng and crack *any* WEP key in two minutes or less. In 2005, a group from the U.S. Federal Bureau of Investigation gave a demonstration where they cracked a WEP-protected network in 3 minutes using publicly available tools. An analytical study for VoIP over WLAN has been performed in 2006 [5]. Apart from unauthorized third parties, privileged users are able to sniff any data passing through their networks. Malicious sniffing agent software can be installed either as a consequence of ISP network compromisation or even as a result of privileged employee behaviour or under administrative orders. In any case the result is the same. Personal credential information and sensitive data are eavesdropped and can be used by unauthorized third parties.

- **Impersonation.** An Impersonator is someone who imitates or copies the behaviour or actions of another. Impersonation is an active attack method where the attacker communicates with another party and uses credentials and data that belong to a third party in order to attack the communicating party. In that case the party under attack is misled by the attacker and falsie authenticates the attacker as the (already compromised) third party.
- **Man in the middle.** The attacker breaches confidentiality and integrity of the session. This is a most sophisticated type of attack since the attacker impersonates the network to the user and vice versa. The attacker intercepts any signalling and/or user data exchanged and has the ability to eavesdrop, inject/modify/delete any signalling or user data exchanged between the user and the network. The worst case in the MITM attacks is the case that the authentication and ciphering data have been compromised, resulting in loss of encryption and identity control over the established session. Current public networks have employed cryptographic mechanisms in order to overcome any security breach in the communication sessions. However, current voice communications, implemented over any kind of public network (GSM, UMTS, CDMA, Internet/VoIP) suffer

from MITM problem. [6][7][8] [9] A MITM attack on a mobile phone network has been extensively described in many articles while next generation mobile networks based on UMTS technology seem to suffer from problems of the same nature [10] [11]. Furthermore it is obvious that it is much simpler for a MITM attack to take place on an IP-based public network since despite the plethora of security protocols involved in IP communications there is no security mechanism that overcomes the problem. While there are attempts to fortify the network against MITM attacks, up to this day there isn't any public network or protocol that manages to overcome this problem. Pre-shared key systems are a solution in some cases [12] they are inefficient though even when the communicating parties belong to the same closed user group. Therefore such solutions are not suitable for use in public networks. Hybrid Cryptographic Systems [13], even the most advanced of them, are not in position to eliminate this threat since the attacker is still able to eavesdrop and/or modify exchanged data traffic.

Fig. 1.1 below shows the typical MITM scenario, where the attacker puts itself "in the middle" of the user and the network, acting as a proxy for each other. MITM behaviour is the same in both wireline and wireless IP networks.

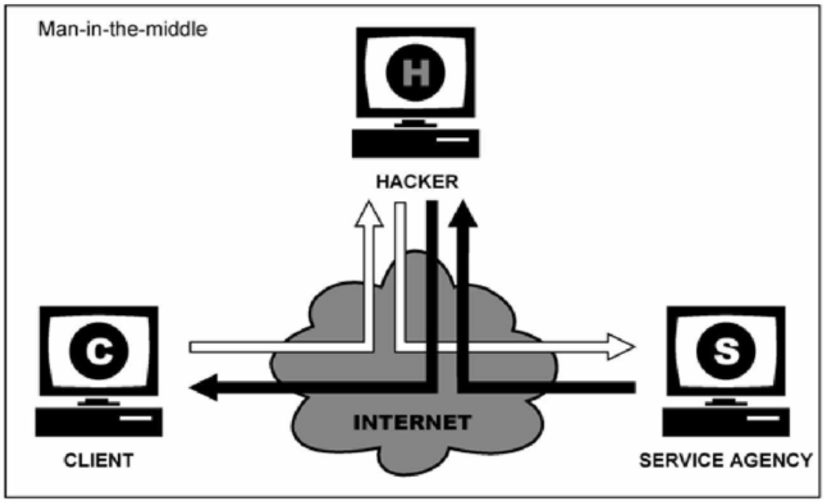


Figure 1.1: Man in the Middle attack

In practice, a MITM attack is implemented through impersonation; the MITM intercepts authentication and ciphering data exchanged over the compromised communications channel, managing to breach authentication and integrity of the session. While there are attempts to fortify the network against MITM attacks, up to this day there isn't any public network or protocol that manages to overcome this problem. From the user perspective there exists however a more horrifying scenario: the case when the network itself acts as a MITM, either intentionally or when under compromisation from an attacker.

1.2. Cryptography

The most common method for securing communications is through encryption. The strength of any cryptographic system is based upon the security of its key distribution mechanism. [14] In symmetric encryption, a key is either shared by the two communicating parties or it is obtained using an out-of-band private channel (offline transfer) [15]. Symmetric key algorithms require a large number of distinct keys in order to achieve secure communications. On the other hand, asymmetric or public key cryptography algorithms rely on public keys for the encryption of data and private keys for the decryption, as well as the creation of digital signatures for user identification [16]. A basic rule in asymmetric cryptography is that it is computationally infeasible to determine the private key given knowledge of the algorithm and the public key or the digital signature. Public/Private key algorithms are used in many crypto systems including ciphers and authentication protocols.

Typical key sizes for symmetric algorithms are 128 to 256 bits while asymmetric algorithms use 2048 to 4096 bit keys. The advantage of faster encryption/decryption in symmetric key algorithms is counterbalanced by the difficulty of supporting a large number of keys. Asymmetric encryption has more computational requirements than symmetric key encryption. That is the main reason for most users to make exclusive use of symmetric key encryption for communications.

1.2.1. Symmetric-key cryptography

A symmetric-key system is an encryption system in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message, as shown in Fig. 1.2. Symmetric-key systems are simpler and faster, but their main drawback is that the two parties must somehow exchange the key in a secure way. Therefore there is the need for an out of band secure channel for the key exchange.

One disadvantage of symmetric-key algorithms is the requirement of a *shared secret key*, with one copy at each end. In order to ensure secure communications between everyone in a population of n people a total of $n(n-1)/2$ keys are needed, which is the total number of possible communication channels. To limit the impact of a potential discovery by a cryptographic adversary, they should be changed regularly and kept secure during distribution and in service. The process of selecting, distributing and storing keys is known as key management, and is difficult to achieve reliably and securely.

Symmetric-key cryptography is sometimes called *secret-key cryptography.* The most popular symmetric-key algorithms are Advanced Encryption Standard (AES), Blowfish, Data Encryption Standard (DES) and Triple DES.

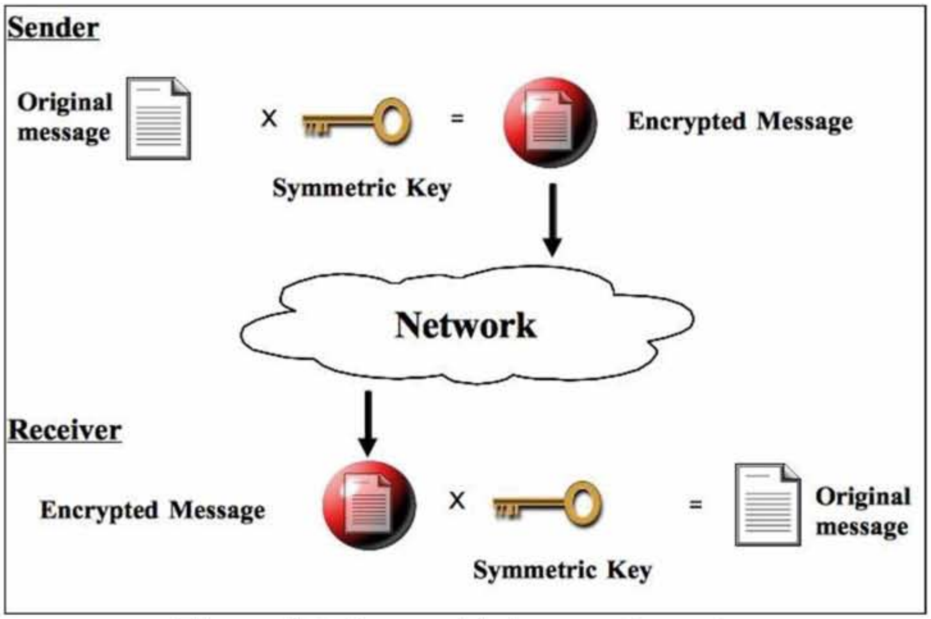


Figure 1.2: Symmetric-key cryptography

1.2.2. Public key cryptography

Public key cryptography is a fundamental and widely used technology around the world. It comprises of a cryptographic system that uses two keys -- a *public key* known to everyone and a *private* or *secret key* known only to the recipient of the message, . When Bob wants to send a secure message to Alice, he uses Alice's public key to encrypt the message. Alice then uses her private key to decrypt it as shown in Fig. 1.3.

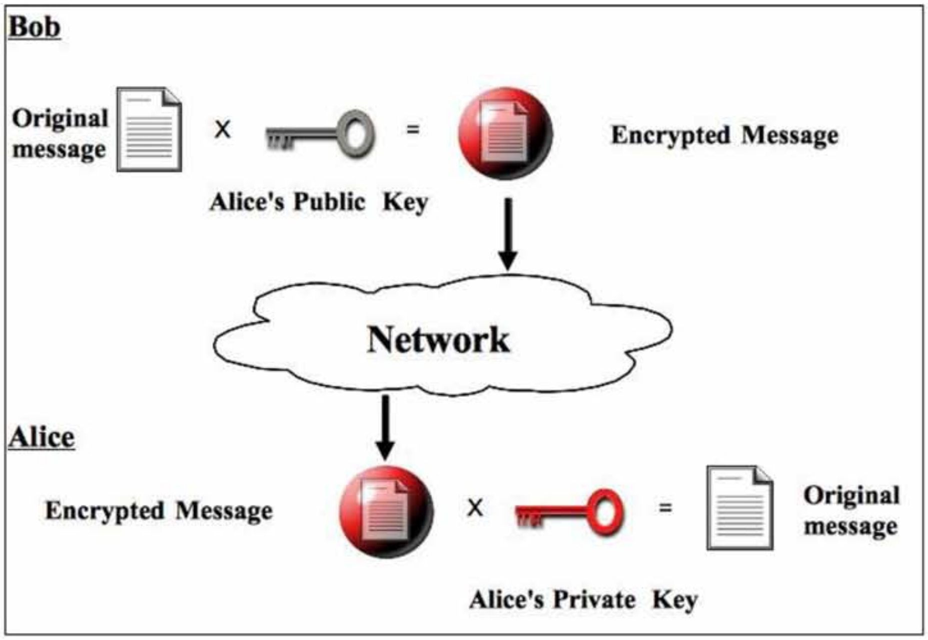


Figure 1.3: Public key cryptography

An important element to the public key system is that the public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt them. Moreover, it is virtually impossible to deduce the private key if you know the public key.

Public-key systems, such as Pretty Good Privacy (PGP), are becoming popular for transmitting information via the Internet. They are extremely secure and relatively simple to use. The only difficulty with public-key systems is that you need to know the recipient's public key to encrypt a message for him or her. What's needed, therefore, is a registry of public keys.

Public key cryptography was invented in 1976 by Whitfield Diffie and Martin Hellman. For this reason, it is sometime called *Diffie-Hellman encryption.* It is also called *asymmetric encryption* because it uses two keys instead of one key (*symmetric encryption*).

In comparison to symmetric encryption it features enhanced security and flexibility but it requires longer keys which greatly increases processing time.

1.2.3. Hybrid cryptography

A hybrid cryptosystem is one which combines the convenience of a public-key cryptosystem with the efficiency of a symmetric-key cryptosystem. A hybrid cryptosystem can be constructed using any two separate cryptosystems:

- A **key encapsulation** scheme, which is a public-key cryptosystem, and
- A **data encapsulation** scheme, which is a symmetric-key cryptosystem.

The hybrid cryptosystem is itself a public-key system, who's public and private keys are the same as in the key encapsulation scheme. Note that for very long messages the bulk of the work in encryption/decryption is done by the more efficient symmetric-key scheme, while the inefficient public-key scheme is used only to encrypt/decrypt a short key value.

To encrypt a message addressed to Alice in a hybrid cryptosystem, Bob performs the following actions:

1. Obtains Alice's public key.
2. Generates a fresh symmetric key for the data encapsulation scheme.
3. Encrypts the message under the data encapsulation scheme, using the symmetric key just generated.
4. Encrypt the symmetric key under the key encapsulation scheme, using Alice's public key.
5. Send both of these encryptions to Alice.

Fig. 1.4 below illustrates the respective cryptographic actions.

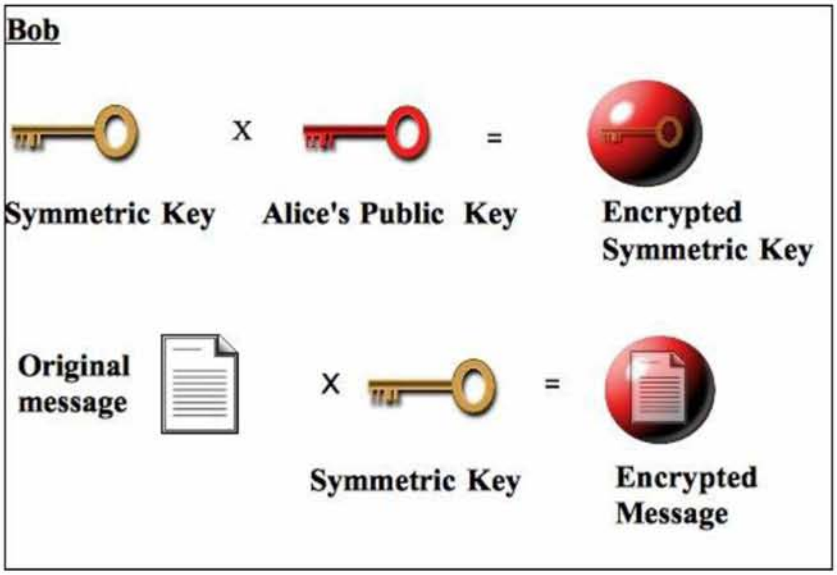


Figure 1.4: Hybrid cryptography, sender's side

To decrypt this hybrid cipher-text, Alice does the following:

1. Uses her private key to decrypt the symmetric key contained in the key encapsulation segment.
2. Uses this symmetric key to decrypt the message contained in the data encapsulation segment.

Fig. 1.5 below illustrates the respective cryptographic actions.

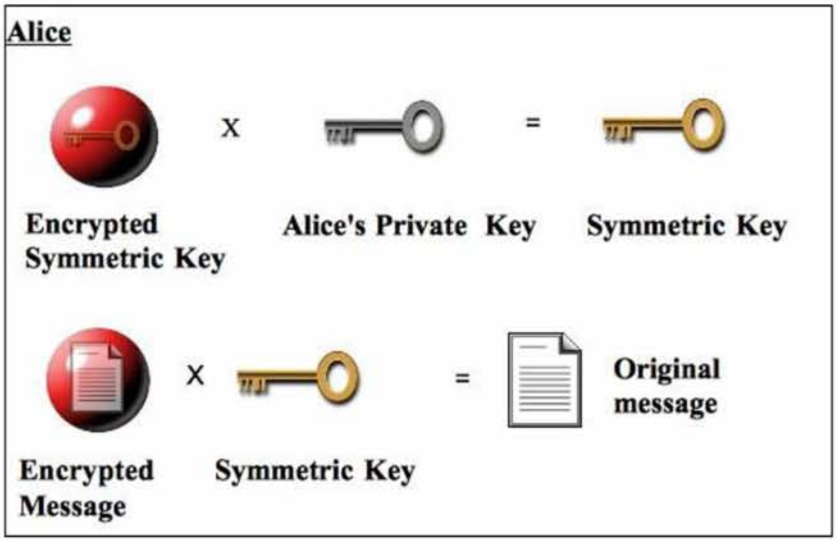


Figure 1.5: Hybrid cryptography, receiver's side

1.3. Public Key Infrastructure

A PKI (public key infrastructure) enables users of a basically unsecure public network such as the Internet to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority. The public key infrastructure provides for a digital certificate that can identify an individual or an organization and directory services that can store and, when necessary, revoke the certificates. Although the components of a PKI are generally understood, a number of different vendor approaches and services are emerging. Meanwhile, an Internet standard for PKI is being worked on.

The public key infrastructure assumes the use of *public key cryptography*, which is the most common method on the Internet for authenticating a message sender or encrypting a message. Traditional cryptography has usually involved the creation and sharing of a secret key for the encryption and decryption of messages. This secret or private key system has the significant flaw that if the key is discovered or intercepted by someone else, messages can easily be decrypted. For this reason, public key cryptography and the public key infrastructure is the preferred approach on the Internet.

A public key infrastructure consists of:

1. A certificate authority (CA) that issues and verifies digital certificate. A certificate includes the public key or information about the public key
2. A registration authority (RA) that acts as the verifier for the certificate authority before a digital certificate is issued to a requestor
3. One or more directories where the certificates (with their public keys) are held
4. A certificate management system

1.3.1. Use of Public and Private Keys

In public key cryptography, a public and private key are created simultaneously using the same algorithm (a popular one is known as RSA) by a certificate authority (CA). The private key is given only to the requesting party and the public key is made publicly available (as part of a digital certificate) in a directory that all parties can access. The private key is never shared with anyone or sent across the Internet. The private key is used to decrypt text that has been encrypted with a respective public key by someone else (who can find out what the public key is from a public directory). Thus, if Bob send Alice a message, he can find out her public key (but not her private key) from a central administrator and encrypt a message to her using her public key. When Alice receives it, she decrypts it with her private key. In addition to encrypting messages (which ensures privacy), Bob can authenticate himself to Alice (so Alice knows that it is really Bob who sent the message) by using his private key to encrypt a digital certificate. When Alice receives it, she can use Bob's public key to decrypt it. Here's a table that restates it:

Table 1.1: Use of Public and Private keys

Action	**Key used**	**Kind of key**
Send an encrypted message	receiver's	Public key
Send an encrypted signature	sender's	Private key
Decrypt an encrypted message	receiver's	Private key
Decrypt an encrypted signature (and authenticate the sender)	sender's	Public key

1.3.2. Infrastructure Providers

A number of products are offered that enable a company or group of companies to implement a PKI. The acceleration of e-commerce and business-to-business commerce over the Internet has increased the demand for PKI solutions. Related ideas are the virtual private network (VPN) and the IP Security (IPSec) standard. Among PKI leaders are:

- RSA, which has developed the main algorithms used by PKI vendors
- VeriSign, which acts as a certificate authority and sells software that allows a company to create its own certificate authorities
- GTE CyberTrust, which provides a PKI implementation methodology and consultation service that it plans to vend to other companies for a fixed price
- Xcert, whose Web Sentry product that checks the revocation status of certificates on a server, using the Online Certificate Status Protocol (OCSP)
- Netscape, whose Directory Server product is said to support 50 million objects and process 5,000 queries a second; Secure E-Commerce, which allows a company or extranet manager to manage digital certificates; and Meta-Directory, which can connect all corporate directories into a single directory for security management

1.4. Problem reporting and management

Network management refers to the activities, methods, procedures, and tools that pertain to the operation, administration, maintenance, and provisioning of networked systems.

1. Operation deals with keeping the network (and the services that the network provides) up and running smoothly. It includes monitoring the network to spot problems as soon as possible, ideally before users are affected.
2. Administration deals with keeping track of resources in the network and how they are assigned. It includes all the "housekeeping" that is necessary to keep the network under control.
3. Maintenance is concerned with performing repairs and upgrades - for example, when equipment must be replaced, when a router needs a patch for an operating system image, when a new switch is added to a network. Maintenance also involves corrective and preventive measures to make the managed network run "better", such as adjusting

device configuration parameters.

4. Provisioning is concerned with configuring resources in the network to support a given service. For example, this might include setting up the network so that a new customer can receive voice service.

A common way of characterizing network management functions is FCAPS - Fault, Configuration, Accounting, Performance and Security.

It is clear from the first objective that identifying network problems and managing them as soon as possible is a major issue for the network. A plethora of network management products have been developed, the most popular of which are HP OpenView Network Node Manager (NNM), Aprisma Spectrum, Computer Associates NetworkIT, Tivoli NetView, What's Up Gold, Concord Nethealth, MRTG and Ciscoworks.

Although they can be used effectively in one administrative domain, their use is prohibited in collaborative network environments where multiple networks are interconnected, forming a Grid that provides services to user from multiple networks.

A trouble ticket (sometimes called a *trouble report*) is a mechanism used in an organization to track the detection, reporting, and resolution of some type of problem. Trouble ticketing systems originated in manufacturing as a paper-based reporting system; now most are Web-based and associated with customer relationship management (CRM) environments, such as call centres or e-business Web sites, or with high-level technology environments such as network operations centres (NOCs). A number of companies make software for trouble ticketing, such as NesterSoft's *Request Commander*. Several other types of software, such as Bluebird include a trouble ticket component.

The Internet Engineering Task Force's Network Working Group specified requirements for a trouble ticketing system in RFC 1297 (NOC Internal Integrated Trouble Ticket System Functional Specification Wishlist). In the RFC document, the author compares the trouble ticket to a patient's hospital chart, because both define a problem and help to coordinate the work of several different people who will work on the problem at different times.

As a ticket moves though the system, it is usually classified as a certain type of issue, which in turn determines the skill set and expertise level of the agent(s) the ticket is assigned to. Until the issue is resolved, the "open ticket" for the problem remains in the work queue, with issues of highest priority taking precedence in terms of work flow.

The format of a ticket is defined per administration domain according to its services management plane and its administrative policies. For Grid networks though, where multiple operators and organisations are involved in network operations, it is lacking a model for the harmonization of the exchanged trouble reports among the Grid members.

Chapter 2

Introduction

While communication technology advances, even more secure communication channels need to be set to guarantee secure information exchange. In nowadays communication, where a secure connection between two devices needs to be established, the pairing problem outcrops. Through a wireless communication that takes place in a specific geographic location, e.g. a meeting room, two devices that share no prior context with each other need to be properly paired in order to protect their subsequent information exchange. During this operation, these devices must agree upon a security association in order to provide the outmost protection to the following communication. On the other hand, even more sophisticated attack methods are planned as wireless communication is easily eavesdropped and manipulated by intruders. These intruders are assumed to be capable of listening to or modifying messages exchanged between the devices. They achieve this manipulation by impersonating one or both of these devices during the process. As a result, secure information exchange between wireless devices needs to be initialized with intuitive techniques for ordinary users, so the devices must be securely paired. The situation is very much the same and even worse when communication channels are carried over multiple operators' public networks, increasing the uncertainty for the users and compromising the channel security.

The traditional approach for securing the communication between two peers is through the use of secret key encryption combined with a Public key approach for exchanging the common secret key to be used by the end peers. The Public key part of the communication is based on a trusted authority for providing the Public keys, a service provided through a Public

Key Infrastructure (PKI). PKIs are vulnerable to Man-In-The-Middle (MITM) attacks, among other approaches that compromise their integrity. There has been a lot of work for providing robust PKI infrastructures, the proposed solutions are fairly demanding on network resources, hence PKI based implementations are not the security approach of choice in several applications that require light-weight solutions.

2.1. Our proposal

In order to overcome malicious MITM attacks without depending on any third party, we propose an innovator, light-weight and sophisticated method for establishing secure data exchange among two (or more) end points. Prior to a trustworthy information exchange, each end point produces its signature data for the session and afterwards, through the use of asymmetric cryptosystem keys and the proposed exchange mechanism, they must exchange this data with each other. Eventually, the user must verify the exchanged signatures on both ends through a straightforward biometric test.

Each device sends a challenge object (we call it session signature) encrypted with its own session private key. The object itself can be anything the user chooses to be, from a random number (which is the default option) to an email address or a video of some kind, depending on user choice and network infrastructure capabilities. Next they exchange their session public keys and the initiating end also sends the session symmetric key.

The user(s) at the end of this handshake procedure perform a user-based biometric test on the session signature data, through visual confirmation of the user(s) of devices' displays when in the same location, or voice/video confirmation when two users are located apart from each other.

If and only if this procedure produces a positive acknowledgement then the communication channel is clear and all communications after that point between the two devices are secured. If not, then the user(s) know that the communication channel and any data exchanged during the procedure are compromised.

We propose two new protocols featuring this method, to handle this “security handshake”, namely *Visual Device Pairing Security* (*ViDPSec*) [79] for co-located devices and *Voice Interactive Personalized Security protocol* (*VIPSec*) [80] for users' devices located apart from each other. This method is in fact a 3-phase security handshake procedure: at first, session’s public cryptography based keys, the session’s symmetric key and session signature are produced; next, the session signatures, public keys and the symmetric key are exchanged over the channel, finally the ***biometric verification*** is performed on the exchanged session signatures. If the verification is successful, the communication channel is clear else the user(s) automatically become aware that security is breached and act appropriately.

The three-phase security handshake procedure of the proposed method is the following:

- Phase1: Private/Public pair of keys generation, Symmetric Session Key and session signature generation.
- Phase2: Session signature, Public keys and Symmetric Session Key exchange.
- Phase3: User Verification.

As far as robustness is concerned, the proposed method guarantees that the connection cannot be breached and the communication is secure due to its high resistance to hypothetical sophisticated attacks. We have developed two protocols, based on device/user proximity. The first protocol (ViDPSec) is for connecting display-equipped devices which are placed close within reach of the user(s). In this case the user verifies visually the contents of the signatures as they are displayed on both screens. The second protocol (VIPSec) is used when two users, located in different places that are connected through an IP communications network capable of transporting voice and/or video streams, wish to establish a secure channel between their devices. In this case the two users verify vocally (audio channel) or visually (video channel) the contents of the signatures among them.

2.2. Cryptographic features of the proposed method

- It does not rely on a public key infrastructure (PKI).

- It does not use permanent keys, instead one time keys are required.
- It uses one time signatures commitment. Signatures are derived from random machine generated data (e.g. random numbers) or user selectable data according to the user preferences.
- It detects Man in the Middle (MITM) attacks by having the user(s) to compare the received objects.
- It has perfect forward secrecy, since the keys are destroyed at the end of the connection.
- It reuses securely exchanged symmetric keys for a short period, in order to fasten connection set-up and minimize user burden.
- It does not rely on any form of signalling for the key management, and in fact does not rely on any servers at all. It performs its key agreements and key management in a purely peer-to-peer manner over the same channel.
- It can be used without being declared or indicated in the signalling path.
- It has two options for data verification, visual and vocal.

2.3. Secure device pairing (device proximity)

Pairing devices close to each other in order to establish secure data transfers among them is fundamental for personal communications, especially in wireless environments. Wireless communications are prone to Man in the Middle attacks; the task to secure the wireless channels is even more difficult when there is no trusted third party or even worse when either the delivering network or the third party itself can't be trusted. In this work we present a light-weighted yet sophisticated protocol for establishing secure communications channels between user devices. The protocol uses any asymmetric cryptosystem's keys and a novel exchange mechanism to establish a Symmetric key that encrypts any data exchanged over the channel. The protocol requires user intervention for its operation; challenge signature/tokens are sent at first that are verified at the end by the user; if the exchanged tokens are verified then the channel is secured by the Symmetric key exchanged inline. The only requirement posed on the devices is that they must be equipped with a display. The protocol is ideal for secure pairing among any display-equipped user device with an adequate processor to play mp3 music.

In order to overcome malicious MITM attacks without depending on any third party, we propose an innovator, light-weight and sophisticated device pairing protocol for establishing secure data exchange among devices with limited capabilities. Even the most inexperienced users will be able to use this technique to solve the pairing problems and thereupon secure their communication. Prior to a trustworthy information exchange, each device separately must produce its signature data for the session and afterwards, through the use of asymmetric cryptosystem keys and the hereafter defined exchange mechanism they must exchange this data with each other. Eventually, the user must verify the signatures on both devices' data through a straightforward biometric visual test. As far as robustness is concerned, the proposed protocol guarantees that the connection cannot be breached and the communication is secure due to its resistance to hypothetical sophisticated attacks.

2.3.1. Assumptions

For this work we are making the following assumptions:

- Scope: device pairing.
- Security: data exchange security objective needs to be satisfied.
- User freedom: users are device independent.
- Device proximity: the two devices need to be close to each other, e.g. two PCs/PDAs on a desk/conference hall, a PDA and a printer in the same location etc.
- Device identification: Our protocol does not identify the end points. The user physically identifies the devices.
- Trust: There is no Certification Authority (CA) or the CA is not trusted; the network is not

trusted either. The devices with the accompanying software are trusted.

- Session lifetime: the objects, Public keys and secrets span only a session's lifetime. Different sessions use distinct data and keys.
- Devices: the devices must be equipped with a small display.

2.3.2. Related work

There has been a significant amount of prior work on building secure pairing methods. They consider different types of device pairing, according to the means they use in order to establish a secure connection between these devices. The Diffie-Hellman key establishment protocol [17] is the base for many of them.

Maher's proposal [18] was the first one that took advantage of the Diffie-Hellman protocol. It uses the Diffie-Hellman key establishment protocol followed by hashing the produced key, e.g. using SHA-1 [19] and truncating it. The user is typically only required to type in (or compare) around 32 binary digits (e.g. in the form of eight hexadecimal digits) between the two devices in order to complete the mutual authentication.

Manual Authentication of wireless devices (MANA) as proposed by Gehrmann et al. [20] [21] is a family of authentication protocols with low bandwidth requirements that uses the human user as the out-of-band (OOB) channel. Subsequent proposals with the same principal have followed [22][23][24]. The Martini Synch protocol [25] achieves device pairing via joint quantization based on a 3-axis accelerometer added to each device. A deterministic key is derived at both sides with maximized entropy that is used as a Private Key for symmetric encryption. The accelerometers are used for relative motion sensing at both ends through random motion of the two devices.

Seeing-is-Believing [26] uses the visual channel to authenticate the devices to be paired. A two-dimensional barcode is attached on one device, representing security-relevant information unique to the device. Another camera-equipped device points its camera to the barcode so that it can read the barcode visually, and uses this information to set up an authenticated channel between the two devices. If both devices are camera-equipped, they can mutually authenticate each other. “Authentication” in this case is based on demonstrative identification rather than with respect to a claimed name. Loud-and-Clear [27], which is similar to Seeing-is-Believing, is based on hash comparison: a text-to-speech engine is used to read an auditorially-robust, grammatically-correct sentence derived from the hash of a device's Public key or a newly computed shared session key. Secure device pairing is achieved by coupling vocalization on one device with the display of the same information on another device.

Our proposal is based on the same principles with MANA family in the sense that we also propose to have the human user as the out-of-band channel. The exchange mechanism however differs from previous proposals; apart from the fact that every single element is generated in real time and spans only a session’s lifetime, we propose the notion of Device Session Signatures (DSS), which acts as the security fingerprint of the device for the specific session. Our approach can take advantage of any available algorithm for generating pairs of asymmetric cryptography keys along with a Private key; next the signatures and the Symmetric key are exchanged with the proposed mechanism and ultimately the Symmetric key is used for data encryption over the established channel.

2.3.3. Mode of operation

The objective of this work is to use the (human user’s) visual channel as the out-of-band verification channel in order to enforce a “security handshake” procedure for the pairing of two devices in proximity. The outcome of the security handshake is the establishment of a Symmetric session key that secures the communication channel by accomplishing user enforced device-to-device encryption. The Symmetric Session Key or SSK, produced with any algorithm available, is exchanged inbound following the procedure defined by our protocol as described hereafter. Asymmetric cryptography keys are used in conjunction with a novel exchange mechanism in order to securely exchange the Symmetric key. Again, any available cryptosystem’s algorithm can be used for the asymmetric key pair generation. The relevant

Public keys are also exchanged inline. There is no need for a Certification Authority (CA) or for permanent keys.

More specifically, when a user wishes to pair two devices the following steps are performed prior the data exchange. Upon device pairing initiation, a set of asymmetric cryptography keys (Private/Public) is automatically generated per device per session. A random number is also generated and encrypted with the respective device's Private key (P_i), resulting in the Device Session Signature (DSS_i). Next, DSS_i's are exchanged, followed by Public keys (C_i) exchange, followed by the Session Symmetric Key (SSK) exchange. At the end, the user verifies the successful exchange of the DSS_i's through visual comparison of the enclosed random numbers.

The sequence of actions is outlined below:

- *Pairing is initiated on both devices by the user.*
- *2nd device sends DSS_2.*
- *1st device upon receipt of DSS_2 responds with DSS_1.*
- *2nd device upon receipt of DSS_1 sends Public key (C_2).*
- *1st device upon receipt of C_2 responds with Public key (C_1). Immediately it encrypts SSK with C_2 and sends it along.*
- *Devices extract the random numbers out of the signatures received and calculate their perceived sum.*
- *User is prompted for verification: if the user positively acknowledges the perceived sums, then the devices are securely paired and everything sent over the open channel are encrypted with SSK. If not, the user is aware of a security breach over the open channel and aborts pairing.*

Fig. 2.1 illustrates the protocol operation, whereas in Table 2.1 we define the protocol messages that are used to implement the proposed exchange mechanism. As stated above, the protocol is simple hence its set of messages is fairly small. Each message is comprised of three fields: "type", "length" and "control data". The format of the ViDPSec protocol messages is presented in Fig. 2.2. The first part of the message, which is the "type" field, holds the type of the message exchanged. This field contains a number between 1 and 3 and the corresponding type values are listed in Table 2.2. The second part of the message, the “length” field, is used to hold the length of the “control data” field. The possible field’s values for the various message types are depended upon the selected key lengths, which is an attribute selected by the user. The third part of the message is the “control data” field, which is used to carry the message data payload. The possible field’s values for the various message types are arbitrary, as they are session specific and they are depended on the selected algorithms and the respective key lengths used.

The user is let free to select any algorithm from any cryptosystem available for the creation of the asymmetric key pairs, e.g. Diffie-Hellman, DSA [28], elliptic curve [29][30] etc. The algorithm for the creation of the Symmetric key is also user selectable, e.g. blowfish [31], 3DES [32], IDEA [33], etc.

Table 2.1: ViDPSec messages

Message	Usage
HELLO	Send DSS to the other device
SENDKEY	Send Public key to the other device
SENDSYMMETRIC	Send Symmetric key to the other device

Table 2.2: ViDPSec TYPE field

Value	Message Type
1	HELLO
2	SENDKEY
3	SENDSYMMETRIC

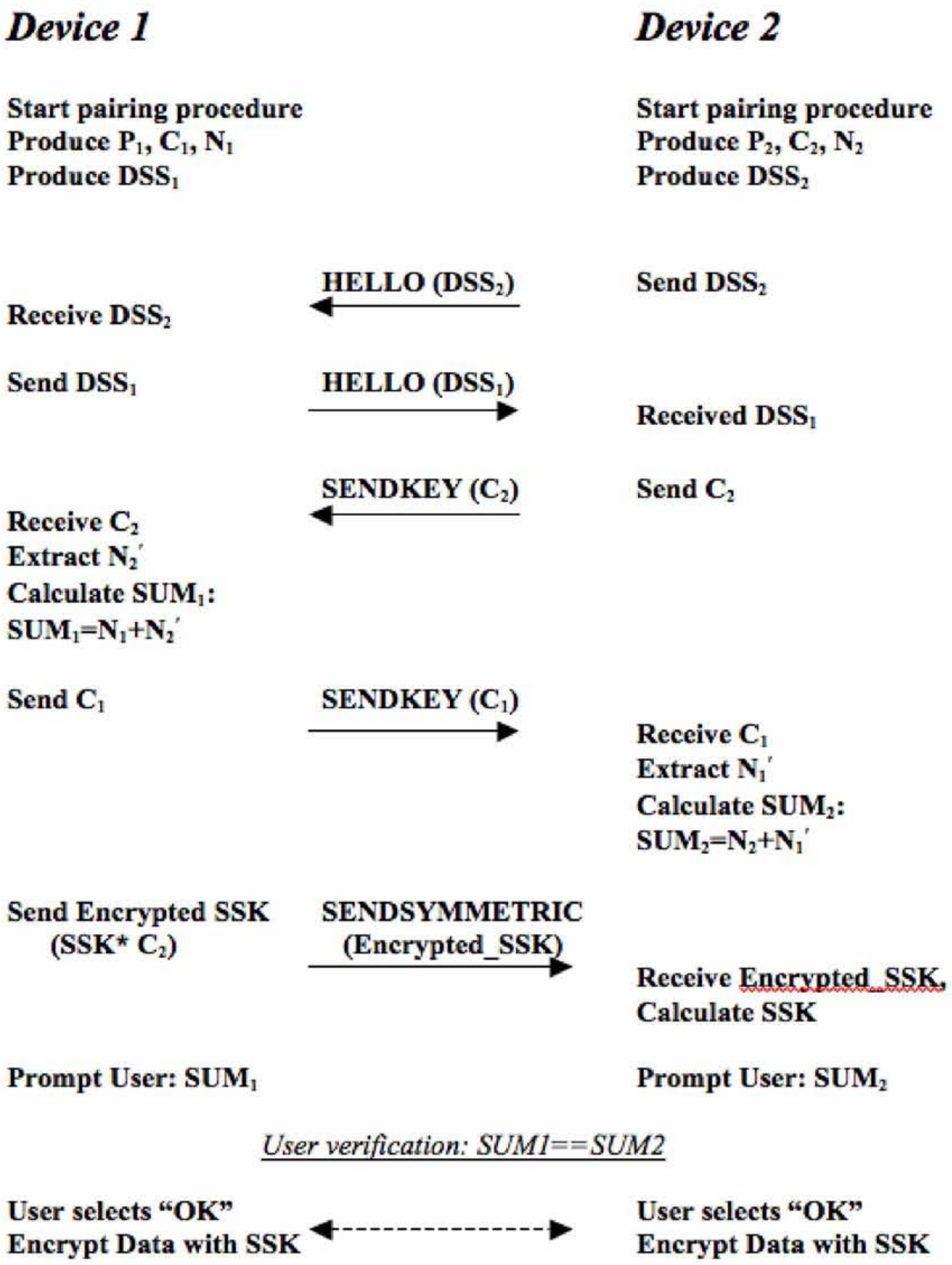


Figure 2.1: ViDPSec operation

TYPE	LENGTH	CONTROL DATA

Figure 2.2: ViDPSec message format

In the case of asymmetric cryptography key pairs, the cryptosystem's algorithms are used only for producing the required pairs of keys (Private/Public) and not for the exchange itself, which is performed following the proposed mechanism. Any future algorithm is inherently supported by the protocol; the protocol objective is the secure Symmetric key exchange and forthcoming secure data exchange and can take advantage of any current or future algorithm for key generation. The protocol is appropriate for ensuring the secure pairing of any two display-equipped devices in proximity, guaranteeing the integrity of the session key exchanged in the beginning of the communication.

The security of the protocol relies on the human assisted verification, through visual comparison of the exchanged challenge numbers, which is trivial, as well as the sequence in which the protocol messages are exchanged. That is, the information is exchanged piece-by-piece and in an orderly manner, forcing each device to deliver what promised, which does the user also verify. Any possible MITM attack is subject to the protocol's operation and therefore is enforced to follow the specified message sequence, which at the end unveils the attempt to the user. The protocol features the same exchange mechanism of the harder VIPSec protocol, where an analogous exchange mechanism is used for securing voice communications between any two familiar users, location free, and they verify among them vocally the session signatures (called User Session Signatures or USSs in this case) at the end of the exchange. ViDPSec inherits security from its predecessor; in fact the security is even stronger as only one user operates the devices and verifies results, without having to coordinate with any other party. Also, in ViDPSec the network architecture is much simpler, as the two devices are directly connected over the wireless channel and there is no third party control over the channel. In addition to the VIPSec proved security, in the ViDPSec there is another element of control for the user, which is the timing of the events, which is totally controlled by the user that performs the device pairing. Monitoring user actions and responding in real time in such a controlled environment adds another level of confidence to the user operations. The security of VIPSec and ViDPSec is analysed in section 2.6.

2.4. Secure device pairing over public networks

The second variation of the method, namely VIPSec, can be used to securely pair users' devices interconnected over multiple public networks, being either the Internet, mobile operators' networks etc. VIPSec is a symmetric key exchange protocol. It performs symmetric key exchange in the media path during call set-up, before any user communication take place. It is transported over the same media stream, resulting in the establishment of a symmetric secret, which is then used to encrypt/decrypt any media path exchange in the application layer for that session. The proposed scheme is both lightweight and effective. An animated presentation of the protocol operation can be found at [34].

2.4.1. Assumptions

VIPSec is based upon the following assumptions:

1. *Scope*: Secure device pairing.
2. *Security*: end-to-end security objective needs to be satisfied.
3. *Public use*: communications in public networks ("open" networks with significant number of users).
4. *User freedom*: users are independent of location and device.
5. *User identification*: Our protocol does not identify the end points. The communicating parties can identify physically each other, i.e. there exists a relationship between them.
6. *Trust*: There is no CA or the CA is not trusted; the network is not trusted either.
7. *Session lifetime*: the objects, Public keys and secrets span only a session's lifetime. Different sessions use distinctive data and keys. As an option, the temporary storage of a symmetric key following a successful key exchange procedure for a short period is supported.
8. *Devices*: support for voice and/or video communications.

2.4.2. Related work

In the security plane, the majority of telecommunications systems have identified weaknesses, depending either on their network architecture design or on their lack of encryption methods and algorithms in the source code. Secure key exchange and data integrity checks are performed in current implementations of voice applications at a low security level.

Recently, Phil Zimmerman introduced a new security protocol for voice communications called ZRTP [35] and a corresponding beta application called ZFone [36]. ZRTP's security is considered to be high, based on the verification through the human voice of a small hash that is derived out of the exchanged public key. Other approaches, similar to ZFone although less effective, are the one proposed by Goodrich et al [27] and the one from Cagalj et al [37]. In the above models based on a small string comparison where the string is derived out of the public key, it is feasible for a MITM to generate a different key with the same hash and act as a relay node with encryption/decryption capabilities [38][39][40][41]. This procedure requires significantly less resources if the attacker has the original hash source in his possession (in our case, the exchanged key(s)), making that kind of attacks possible even in real world conditions. Additionally, the short length of the produced hash is another important element that considerably helps a potential attacker to successfully complete an active attack as well.

In addition, ZRTP provides a second layer of authentication against a MITM attack, based on a form of key continuity. It does this by caching some hashed key material to use in the next call, to be mixed in with the next call's Diffie-Hellman shared secret, giving it key continuity properties. If the MITM is not present in the first call, he is locked out of subsequent calls. Although a valuable security add-on, the basic weakness of this method is that it is device dependent, since the users have to use the same equipment each time in order to call each other.

VIPSec just like ZRTP features confirmation of an object rather than a small hash value, through either voice or video, and a completely different key exchange mechanism. The main idea is that the calling parties firstly exchange random strings (or other objects) encrypted with their private keys and just after that their public keys. In the next step, the first user produces a symmetric key, encrypts it with the second user's public key and sends it to him. Finally, a symmetrically encrypted communication channel is established and the two users vocally confirm the exchanged strings (or objects). In VIPSec the security relies only on the effectiveness of the algorithms and the lengths of the keys used, avoiding the hash production phase which may constitute a security weakness.

2.4.3. Mode of operation

When Alice and Bob need to communicate the procedure explained hereafter is taking place before any data exchange is performed. A set of asymmetric cryptography keys (Private/Common) is automatically generated per user per session. Users exchange challenge objects encrypted respectively with their individual session Private Key (P_i), resulting in their User Session Signature (USS_i). The object itself can be anything the user selects: a random number, a string, an audio or video file etc. Next they exchange their session Public Keys (C_i) and the calling party also sends the Session Symmetric Key (SSK). The communicating parties at the end of this handshake procedure perform a user-based biometric test on the USSs, by using either voice (1st biometric level) or video confirmation (2nd biometric level) of each other's identity and the nature of the exchanged data. If this procedure produces a positive acknowledgement by both users then their communication is secured and everything sent over the network are encrypted with the Symmetric Key (SSK) that was securely established during the first phase above. If not, the parties know that the communication channel and any data exchanged during the procedure are compromised. Fig. 2.3 below illustrates the phases and actions of the VIPSec protocol as described above.

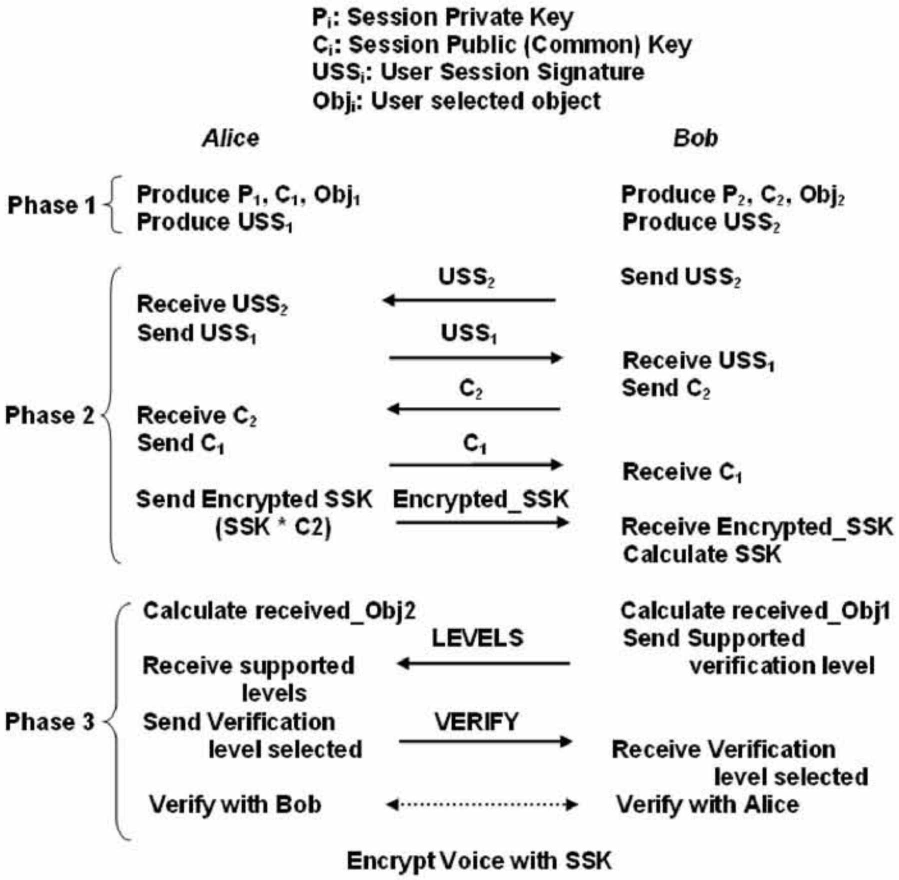


Figure 2.3: VIPSec phases and actions

Message Format

The message format is shown in Fig. 2.4. The first part of the message is the TYPE field that holds the type of the message exchanged. This field contains a number between 1 and 5.

The second part of the message is the LENGTH field, which is used to hold the length of the CONTROL DATA field. The third part of the message is the CONTROL DATA field, which is used to hold the message data payload.

TYPE	LENGTH	CONTROL DATA

Figure 2.4: VIPSec message format

Messages

The first part of the message is the TYPE field that holds the type of the message exchanged. This field contains a number between 1 and 5 and the corresponding message types are shown in Table 2.3. The second part of the message is the LENGTH field, which is used to hold the length of the CONTROL DATA field. The possible LENGTH values for the various message types are shown in Table 2.4. The third part of the message is the CONTROL DATA field, which is used to hold the message control data payload. The possible CONTROL DATA values for the various message types are shown in Table 2.5. The sequence of messages exchanged during the protocol operation is shown in Table 2.6.

After the message exchange Alice and Bob use the selected verification level to verify the USSs and then they proceed with normal voice communication encrypted with the symmetric key SSK.

Table 2.3: VIPSec TYPE field

Value	Message Type
1	HELLO
2	SENDKEY
3	SENDSYMMETRIC
4	LEVELS
5	VERIFY

Table 2.4: VIPSec LENGTH field

Type	Length
1	depended on selected key lengths
2	depended on selected key lengths
3	depended on selected key lengths
4	2
5	2

Table 2.5: VIPSec CONTROL DATA field

Type	Data
1	Arbitrary
2	Arbitrary
3	Arbitrary
4	00: voice not supported, video not supported 10: voice supported, video not supported 11: voice supported, video supported
5	10: voice confirmation selected 11: video confirmation selected

Table 2.6: VIPSec messages

Seq. No.	Message	Control Data Payload
1	HELLO (USS_2)	User2 Session Signature
2	HELLO (USS_1)	User1 Session Signature
3	SENDKEY (C_2)	User2 Common Key
4	SENDKEY (C_1)	User1 Common Key
5	SENDSYMMETRIC (Encrypted_SSK)	Session Symmetric Key as selected by User1, encrypted with User2 Public key
6	LEVELS (Voice, Video)	Flag Voice and/or Video capabilities for User 2
7	VERIFY (Voice/Video)	Selected verification level (Voice/Video)

Caller Operation

The initiator of a VIPSec secure device pairing performs the following operations:

1. Trigger device pairing (this is achieved either using SIP, or any other way i.e. directly using the remote IP address).
2. Private/public session key generation; object selection; signature creation.
3. Upon receipt of the “Hello” message responds with his own “Hello” message.
4. Upon receipt of the “SendKey” message responds with his own “SendKey” message and the “SendSymmetric” message.
5. It performs USS_2*C_2 to derive $Object_2$.

6. Upon receipt of the “Levels” message, performs a bitwise ‘&’ operation on the supported levels of both parties and sends the result in his own “Verify” message. It also triggers the appropriate confirmation procedure (i.e. video verification application call).
7. Start encrypting all packets with SSK symmetric key.
8. Verify Object_2 and Object_1 with other party.
9. If verification succeeds then channel is secure, otherwise security is compromised (pairing can be aborted).

Callee Operation

The receiver of a VIPSec secure device pairing performs the following operations:

1. Accept trigger for secure device pairing.
2. Private/public session key generation; object selection; signature creation.
3. “Hello” message sent.
4. Upon receipt of the “Hello” message responds with “SendKey” message.
5. Upon receipt of the “SendSymmetric” message it performs $\text{Encrypted\_SSK}*P_2$ to derive SSK.
6. It performs USS_1*C_1 to derive Object_1.
7. “Levels” message sent.
8. Upon receipt of the “Verify” message triggers the appropriate confirmation procedure (i.e. video verification application call).
9. Start encrypting all packets with SSK symmetric key.
10. Verify Object_2 and Object_1 with other party.
11. If verification succeeds then channel is secures, otherwise security is compromised (pairing can be aborted).

2.5. Security analysis

In this section we analyse the security of our method. This is achieved (a) through intuitive validation, (b) through the provision of security fundamentals implemented by the protocol followed by (c) an exhaustive step by step analysis.

We perform the analysis for ViPSec that operates with stronger network and user requirements, as ViDPSec resistance to attacks is as strong as VIPSec; in fact the security is even stronger as only one user operates the devices and verifies results, without having to coordinate with any other party. Also, in ViDPSec the network architecture is much simpler, as the two devices are directly connected over the wireless channel and there is no third party control over the channel. In addition, there is another element of control for the user, which is the timing of the events, which is totally controlled by the user that performs the device pairing. Monitoring user actions and responding in real time in such a controlled environment adds another level of confidence to the user operations.

2.5.1. Intuitive protocol validation

Intuitively the protocol can be validated if we focus respectively on:

- What is exchanged.
- How it is exchanged.
- How it is verified.

For the first phase, the key is the *User Session Signatures.* Each party creates a challenge signature token by *ciphering the data to be verified with the user session's Private Key.* By doing so, not only it is impossible to get the data without the Common Key which is exchanged in forthcoming steps of the protocol but also the signature itself is both *dynamic* and *personalized*. This is of key importance as the signature data are verified by the users at the end of the 3-phase security handshake process.

For the second phase, the answer comes from the steps that we follow: each user is giving out piece by piece its authenticity and identity data. At first, signatures are exchanged among the two parties; next the Common keys are exchanged in order to extract the useful data out of them. Only after that point the Symmetric Session Key is exchanged, which will be used to secure the communication channel *in case the USS's are verified successfully.*

For the third phase, personalization of the verification procedure through a simple yet powerful biometric test is the answer. Personalized data used as the signature tokens which are verified through the use of voice or video by the users that created them produces a most powerful verification procedure with unmet security at this level.

2.5.2. Asymmetric cryptography key features

Asymmetric or public key cryptography algorithms rely on public keys for the encryption of data and private keys for the decryption, as well as the creation of digital signatures for user identification [18]. A basic rule in asymmetric cryptography is that it is computationally infeasible to determine the private key given knowledge of the algorithm and the public key or the digital signature. Public/Private key algorithms are used in many crypto systems including ciphers and authentication protocols. Asymmetric cryptography keys have the following properties:

1. Independency: keys while paired, they are independent, and it is not feasible to generate one out of the other.
2. Non inversion: encrypted data don't reveal the key used for the encryption.
3. Encryption: when data destined for a user are encrypted with the user's common key then only this user can decrypt them with his private key.
4. Identification: when data are encrypted with a user's private key they form a signature data object that verifies the identity of the sender. Anyone can decrypt these data with the user's common key.

2.5.3. Bonding Pair

The User Session Signature (USS) of a user is encrypted with the Private Key of that user and the user's Common key is used to decrypt the signature and reveal the user selected data to be verified. We say that the USS and the Common key form a ***bonding pair***.

Bonding Pair Properties

- USS reveals sender's identity.
- USS carries one time user selected data.
- USS and Common key are paired: USS_i is decrypted only with the relevant common key C_i.
- The data carried in the USS are selected by the user and they are verified by the same user when the key exchange is completed. A bonding pair is considered valid (*authentic*) at the receiving party only after a successful voice or video verification procedure.

2.5.4. Biometric Signatures

Signatures are one time session data that are encrypted with the respective session Private key of the user. The nature of the data is left to the user. One can use a picture that reflects a situation the other user is aware of, a video clip etc. i.e. objects that embed emotional human data that only the two users can recognize (and therefore produce and verify). A more lazy user or when such data are not present, for example when someone is using another device, the signature is obtained by encrypting a session generated random number.

2.5.5. Signature Commitment

Signature commitment is a key feature of the protocol. The semantics of the signatures commitment are the following: a) a user who commits to a signature cannot change it afterwards during the session lifetime (*binding*) and 2) the commitment is not revealed to the receiver until the sender "opens" it (*hiding*).

2.5.6. Verification of signatures by the users

At the end the signature data are verified by the communicating parties as users. In our approach we use two levels for verification, the second being much stronger: 1) voice verification and 2) video verification.

2.5.7. Analysis

Below we provide the facts that validate the correctness of VIPSec and its resistance to attacks:

1. USS/Common key form a *bonding pair.*

2. USS exchange is performed *prior* to any key exchange; both parties have in their disposal a USS to verify at the end *before* they send the accompanying part of the bonding pair.

3. USS data are *one time user selected data* of arbitrary type (challenge) which are encrypted with the private key of the user; it is not feasible for any third party to guess the challenge data.

4. Elements are mutually exchanged *step by step*. The relevant USS and session keys are exchanged in turns by both parties piece by piece and not altogether.

5. An *authentic bonding pair* that reaches its destination (it was not altered or faked) guarantees a successful verification procedure at the end, which in turn results in a *successful* symmetric key exchange.

6. If a user receives a bonding pair that is not authentic, then the user can safely assume that an attack is in place and abort the procedure.

7. As a result of 5 and 6 above, the only way for a malicious middle man to compromise the

channel is through *impersonation*, i.e. a fake bonding pair has to be created. This means than MITM need to guess the object to be exchanged. Due to the user based verification procedure at the end of the process the forgery of the bonding pair is proven and the attack is noticed by the users.

2.5.8. Step by step exhaustive analysis

In order to intercept and interfere (I&I) in protocol handshaking steps knowledge of user selected data is required. This is proven in the following step by step possible I&I by any attacker to the protocol operation steps.

Possible I&I scenarios:

There are three possible scenarios that can take place where a malicious middle man X can attack by trying to intercept & interfere (I&I) communications between A and B:

- X is in the middle trying to alter protocol messages during its execution with both parties
- X impersonates first B to A and then A to B. In this case when A calls B, X responds as if it were B, executes the protocol with A and then X calls B as if it were A and executes a second protocol instance with B. Finally X bridges the two "half circuits" together.
- X impersonates first A to B and then B to A. In this case when A calls B, X "holds" the call, calls B as if it were A, executes the protocol with B and then X continues and completes execution of a second protocol instance with A as if it were B. and. Finally X bridges the two "half circuits" together. This case is symmetrical to case 2 above.

First we consider I&I scenario 1. We will examine step by step the protocol execution between the two parties and we will identify any possible I&I by X. In every case a dead-end is reached (no I&I feasible).

Step1: User B sends USS_2 to User A

Possible I&I:

In order to create and forward a USS, an object has to be encrypted with a corresponding private key. This means that the only feasible I&I in this step is to produce and forward a fake USS (USS_X) using an arbitrary object. This object has to be encrypted with a (fake) private key P_X, which as anticipated results in a *fake bonding pair* (which in turn results to an attack event noticed by the communicating parties at the end, item 7 in the Analysis part F above).

Step 2: User A sends USS_1 to User B

Possible I&I: Same as in Step 1.

Step 3: User B sends C_2 to User A

Possible I&I:

- If USS_2 was not faked in Step 1 then bonding pair properties are followed or else no pair is formed; as a consequence C_2 *has* to be forwarded and therefore there is no feasible I&I.

- If USS was faked (USS_X was forwarded to User 1 in Step 1 instead of USS_2) then bonding pair properties must be followed. An also fake common key C_X has to be forwarded. The fake bonding pair will be accepted by A and the extracted object that reaches A differs from the original send out by B. This will be revealed in the verification phase where *verification will fail.*

Step 4: User A sends C_1 to User B

Possible I&I: Same as in Step 3.

Step 5: User A sends Encrypted_SSK to User B (symmetric key SSK is encrypted with C_2)

Possible I&I:

Original encrypted symmetric key (Encrypted_SSK) is forwarded to B which in turn uses his private key P_2 to decrypt it. This leaves both A and B with the same symmetric key SSK which means that no I&I is possible. In this case *X gets the encrypted form of the symmetric key but not the symmetric key* itself.

OR

A different symmetric key (SSK_X) is encrypted with (eavesdropped in Step 3) C_2 resulting in Encrypted_SSK_X. B uses his private key P_2 to decrypt it to derive SSK_X. In this case A and B end up with *keys that don't match (A has SSK <> SSK_X that B has)*, symmetric cryptography between A and B *fails*. A has SSK, B has SSK_X, MITM knows SSK_X but not SSK, so X is not able to use different key for each party.

OR

A different symmetric key (SSK_X) is encrypted with a fake common key (C_X), resulting in Encrypted_SSK_X. In this case B can't decrypt it with his private key P_2 which results once more in procedure *fail.*

This proves the resistance of our protocol for the first scenario. The rest two scenarios are schematically shown in Fig. 2.5 below. In fact they are different views of the same principles. For the second scenario, X is impersonating B to A and has to send a bonding pair to A. For start, a USS has to be sent in Step1. Thus X has to select an object that will be verified at the end as originating by user B. Since B selects arbitrary objects it is not feasible for X to guess the object that B will choose later in time, therefore verification will fail at the end (same as in Step 1 for scenario 1). For the third scenario the same argument holds with users A and B interchanged.

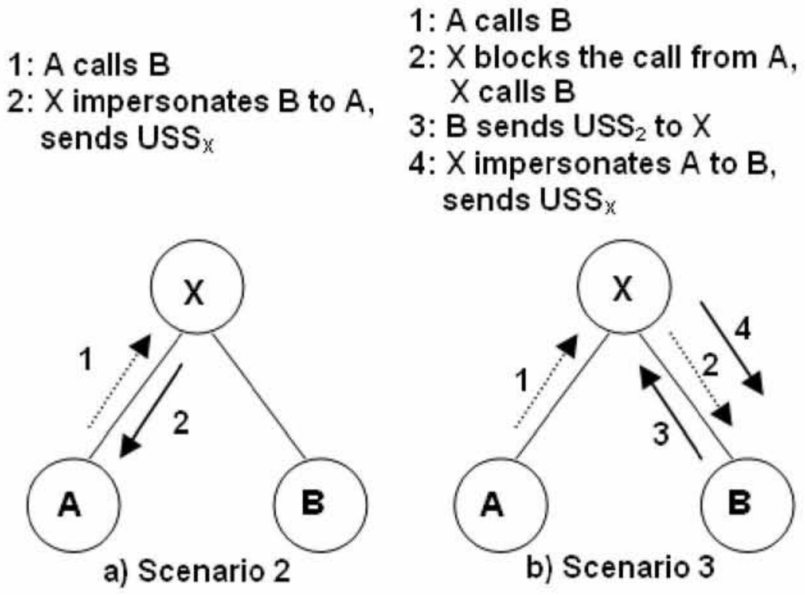

1: A calls B
2: X impersonates B to A, sends USS$_X$
1: A calls B
2: X blocks the call from A, X calls B
3: B sends USS$_2$ to X
4: X impersonates A to B, sends USS$_X$
X
A
B
1
2
3
4
a) Scenario 2
b) Scenario 3

Figure 2.5: VIPSec possible attack scenarios.

2.5.9. Voice/video mimic attacks

We examine below the protocol's resistance to various kinds of voice attacks.

- Pure Voice/Video Forgery Attack: In case 1st level verification is selected (vocal confirmation) and the MITM can mimic both human voices, he is able to fake the two bonding pairs to the caller and the callee respectively, thus implementing two separate channels; in the verification phases with both parties he could verify the faked objects. Bridging the two channels (decrypting original packets and re-encrypting them before forwarding to the other party) completes the attack. Even in this case, if biometric data are used in the original USSs (like personal photo, etc.) and the verification is repeated during the normal call, the attack will fail. If 2nd level verification is selected (video verification) the attack is practically impossible.
- Bill Clinton Attack: The Bill Clinton Attack is based on the assumption that the celebrity will not remember an ordinary person's voice, while that person will remember the celebrity's voice. In case 1st level verification is selected (vocal confirmation), the MITM

can initiate a call some time in advance and store the verification of a selected object with the celebrity. When the celebrity is contacted in the future by that person, the MITM forges the celebrity's voice for the false object and then establishes a session with the celebrity. Bridging the two channels (decrypting original packets and re-encrypting them before forwarding to the other party) completes the attack. Again, if biometric data are used in the original USSs (like personal photo, etc.) and the verification is repeated during the normal call, the attack will fail. Also, if 2nd level verification is selected (video verification) the attack is practically impossible.

- Six Month Attack - Court Reporter Attack: Both the 6 Month Attack and the Court Reporter Attack are based on the assumption that the two parties are not familiar with each other, which is controversial to the assumptions of the current work.

2.6. Case study

We hereby provide a comprehensive presentation of all the latest contesters in this research area, which we compare to the one proposed by this thesis. We have to note that all the other proposals offer a solution to "half" of the problem, as they can pair devices that are either close together (SiB, L&C) or communicate over a public network (ZRTP). In addition to that ZRTP is limited to VoIP communications and can' t be generally applied to every communications network (GSM, UMTS, etc).

Our proposal comprises a unified scheme for both use cases using verification exploiting the nature of the application, which is not limited to IP communications and can be generally applied to any communications network that can support its basic requirements, regardless the adopted network technology.

2.6.1. Seeing-is-Believing (SiB)

SiB uses the visual channel to authenticate the devices to be paired. A two-dimensional barcode is attached on one device, representing security-relevant information unique to the device (hash of the Public Key). Another camera-equipped device points its camera to the barcode so that it can read the barcode visually and also gets the same information over the wireless channel. It compares the two forms and upon verification it uses this information to set up an authenticated channel between the two devices. If both devices are camera-equipped, they can mutually authenticate each other.

Strengths & Weaknesses

SiB provides a reasonable level of security. It is also adequately practical particularly because it places not much burden upon the human user: visual identification of the target device and taking a picture of the barcode.

It suffers though from various weaknesses in security and usability terms:

1. It assumes device proximity and is inadequate for devices further apart.
2. The devices must be camera-equipped, therefore

   a) it is good for mobile phones but impractical for laptops. Use of PDAs, access points, printers etc is generally excluded by nature.

   b) it is depended on lighting conditions and other camera-related issues
3. It provides one way authentication in most cases (e.g. access-point to laptop, printer to PDA etc), two-way authentication can be achieved only when both devices are camera-equipped and user controlled.
4. A faked barcode attached on a device can lead to false authentication.

Security evaluation

SiB provides weak security. It proves insufficient if the adversary: (1) either hack into the target device and cause it to display wrong barcodes or physically plaster fake barcodes on the device, and, (2) mount a man-in-the-middle attack on the wireless channel.

2.6.2. Loud&Clear (L&C)

Similar to Seeing-is-Believing, it is based on Short Authentication String (hash) comparison: a text-to-speech engine is used to read an auditorially-robust, grammatically-correct sentence derived from the hash of a device's Public key. Secure device pairing is achieved by coupling vocalization on both devices or on one device and the display of the same information on another device. Currently the Text-to-Speech functionality is supported for Pocket PC and Windows PC. They are working on porting Sun's FreeTTS and JSAPI to Ewe so that TTS is supported in most devices.

Strengths & Weaknesses

L&C provides a reasonable level of security, quite better that SiB. It is moderately practical since it requires the user(s) to verify the audio sources of the devices and the vocalized hashes to verify the proper sequence of actions.

It suffers though from various weaknesses in security and usability terms:

1. It assumes device proximity and is inadequate for devices further apart.
2. It requires audio outputs on device, which is not always available.
3. It is based on small hash comparison, via a text-to-speech engine and the audio channel.
4. It is based on the audio channel; sometimes it proves difficult to verify the audio source (i.e. when in a crowded meeting place)
5. It suffers from portability issues.

Security evaluation

L&C level of security is adequate and in any case stronger than SiB. It may prove inefficient though when a device is further than anticipated and the adversary's device acts in its place, or when the audio source is difficult to be identified. It compares hashes of public keys, which is another weak point.

2.6.3. ZRTP

ZRTP is described in the Internet Draft as a *"key agreement protocol which performs Diffie-Hellman key exchange during call setup in-band in the Real-time Transport Protocol (RTP) [75] media stream which has been established using some other signalling protocol such as Session Initiation Protocol (SIP) [77]. This generates a shared secret which is then used to generate keys and salt for a Secure RTP (SRTP) [76] session."* One of ZRTP's features is that it does not rely on SIP signalling for the key management, or on any servers at all. The protocol does not require prior shared secrets or rely on a Public key infrastructure (PKI) or on certification authorities. According to the protocol description there is no assumption on user familiarity, i.e. that the two parties know each others' voices, but only in their ability to distinguish different voices.

ZRTP uses the Diffie-Hellman key exchange to establish a shared secret among the peers. In order to overcome DH's vulnerability to MITM attacks, ZRTP uses the Short Authentication String (SAS) feature. SAS is in fact the hash value of the shared secret, which is displayed to the end users and is intended to be verified vocally by them during their conversation. If the two values are verified and found in order, there is a high probability that no MITM attack is taking place during the specific conversation. It assumes use of the same device and thus can take advantage of key continuity. It does so by using Zid, a unique identifier calculated at software installation time and which is used as an index for all cryptographic data over time with the specific install.

Strengths & Weaknesses

ZRTP provides a good level of security. It is quite practical since it requires the user(s) to verify the hashes of the keys.

It also suffers from the following inefficiencies:

1. It is device depended in order for key continuity to work.
2. It is based on small hash comparison.
3. It has complex implementation.

4. It is limited for VoIP communications.
5. It is impractical when devices are close together.
6. It cannot be used for pairing closely located low feature devices, e.g. PDAs & printers.

Security evaluation

ZRTP security level is strong. It compares small hashes though, which is a weak point, as shown in 2.6.5 below. It also lucks a video verification mode, which is much stronger than the vocal one.

ZRTP can fail in the following cases:

1. Key continuity can be broken at any time by the attacker: a new Zid is calculated every time the software is installed. This can normally happen as a result of re-installation of the software on the same platform or due to the use of a new device from the user side. Therefore the attacker when acting as MITM can always use fresh Zid's, which is accepted by the implementation.
2. Diffie-Hellman key exchange compromisation: a powerful MITM can practise a successful attack as described hereafter. Under normal conditions Alice sends K_A to Bob and Bob sends K_B to Alice. The Diffie-Hellman session key S_{AB} is established from K_A and K_B. Alice and Bob have to verify two small strings, $SAS_A=h(K_A)$ and $SAS_B=h(K_B)$; If the attacker can find K_A' and K_B' such that $h(K_A)=h(K_A')$ and $h(K_B)=h(K_B')$, then he intercepts the media path and implements two half-connections between the two parties, using the Diffie-Hellman keys created with his own cryptographic material for the two halves, S_{AM} and S_{MB} respectively. He continues by de-encrypting and re-encrypting every message exchanged among the two compromised sessions, using S_{AM} and S_{MB} respectively. Alice and Bob verify the correct SAS values but MITM can intercept all encrypted communication under the compromised sessions. Fig. 2.6 below illustrates this scenario.
3. Pure voice forgery attack: When Alice starts a ZRTP session with Bob, MITM intercepts it and, instead of forwarding it to Bob, impersonates Bob to Alice,

establishing a session with Alice. At the same time, MITM initiates a second ZRTP session with Bob, impersonating Alice to Bob. This results in two half-sessions (Alice-to-MITM and MITM-to-Bob), with MITM sitting in the middle having compromised the audio channel. The relevant keys S_A and S_B are different since they are the result of two independent Diffie-Hellman key exchanges, one between Alice and MITM and another one between MITM and Bob. MITM can mimic both human voices, therefore he is able to fake the two SAS values to both parties respectively; so, when the parties wish to verify these values he could verify the faked SAS and complete the attack by bridging the two channels (decrypting original packets and re-encrypting them before forwarding to the other party). Fig. 2.7 below illustrates this scenario.

4. Bill Clinton Attack: A celebrity will not remember a random person's voice; however the random person will remember the celebrity's voice. The intruder executes SAS with the celebrity some time in advance, to create a false shared secret. When the initiator then later on wants to contact the celebrity, the MITM forges the celebrity's voice for the SAS string, and then establishes a session with the celebrity. Finally, the intruder relays and records all traffic. Note that for this attack, one does not need to know who the random person that wants to contact the celebrity is -- one only needs the celebrity to forget voices between sessions.

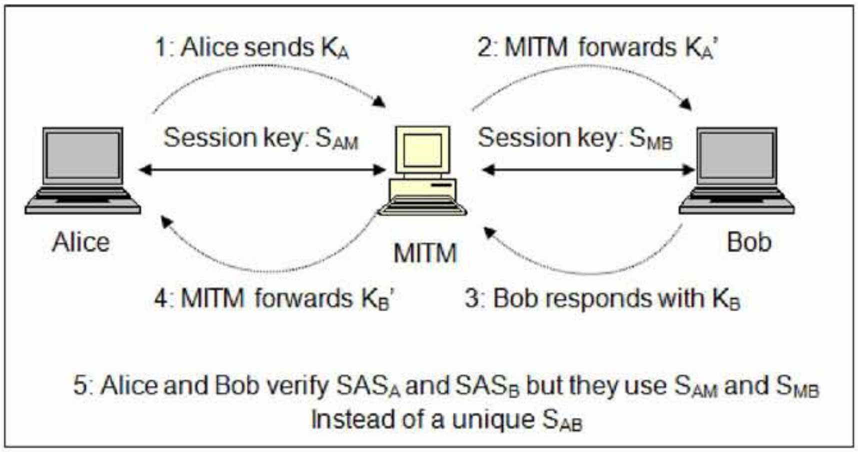

1: Alice sends K_A
2: MITM forwards K_A'
Session key: S_{AM}
Session key: S_{MB}
Alice
MITM
Bob
4: MITM forwards K_B'
3: Bob responds with K_B
5: Alice and Bob verify SAS_A and SAS_B but they use S_{AM} and S_{MB}
Instead of a unique S_{AB}

Figure 2.6: Diffie-Hellman compromisation attack on ZRTP

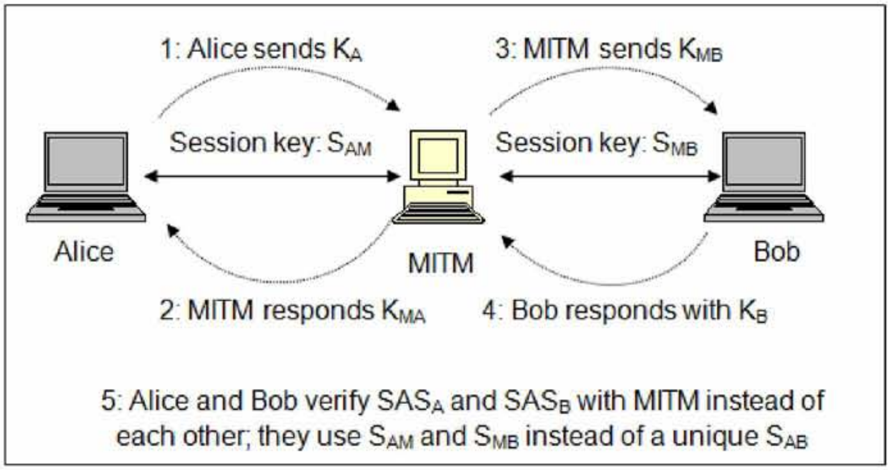

1: Alice sends K_A
3: MITM sends K_{MB}
Session key: S_{AM}
Session key: S_{MB}
Alice
MITM
Bob
2: MITM responds K_{MA}
4: Bob responds with K_B
5: Alice and Bob verify SAS_A and SAS_B with MITM instead of each other; they use S_{AM} and S_{MB} instead of a unique S_{AB}

Figure 2.7: Pure voice forgery attack on ZRTP

2.6.4. Our proposal (ViDPSec/VIPSec)

Signatures are created from one time random data by encrypting them with the relevant Private Key; information is exchanged piece-by-piece, first the Signatures - then the Public Keys and finally the signatures are opened in order for the random data to be retrieved and verified by the user(s). According to the use case, verification is done either visually (device proximity – ViDPSec) or through voice or video (devices in distance, VIPSec).

Advantages over other proposals

1. It works in distance mode (VIPSec) in addition to proximity mode (ViDPSec).
2. The video verification mode adds an unmatched level of security (VIPSec).
3. It uses user verifiable data, which can be either random or personal data.
4. It does not make use of hashes or permanent cryptographic elements.
5. It provides maximum practicality with minimal user burden.
6. It has simple implementation.
7. It does not interact with the protocol stack; therefore it can be used on any communications network (IP, UMTS, etc).
8. Resistant to voice forgery and Bill Clinton attacks.

Security evaluation

The security of our method has been proven and accepted by the research community [78].

2.6.5. Hash functions & small hash comparison

Use of hashes is a common practice in cryptographic solutions. There exist however an ever increasing number of publications as to why hashes are not the ideal solution, as they can prove inefficient.

Hash algorithms are used in many cases, especially in the Internet, because they have two security properties: to be one way and collision free. The recent attacks have demonstrated though that one of those security properties is not true, more commonly the second one [70]. A lot of researchers have explored those weaknesses [40] [68] [69] [71].

A lot of work has been done on attacking specific hash functions. The MD5 hash algorithm was proven susceptible to collision attacks [72]. A faster method for finding MD5 collisions was also published [39]. A variant of the SHA-1 was also attacked [73]; there was also predicted that the normally used SHA-1 would also be susceptible with a large amount of work [74].

The outcome is that, while hash codes are supposed to generate a unique value for a given input, the fact is that, while difficult to accomplish, it is technically feasible to find two different data inputs that hash to the same value. The length of the generated hash code and the complexity of the data being hashed are determining factors, as it is stated in [41].

2.7. Experimental results

We focused on VIPSec protocol which is more demanding and less tolerant to network delays than ViDPSec. For the calculation of the overheads imposed by the protocol primitives during execution we used a typical user PC equipped with a 3GHz Pentium 4 processor and 1GB of RAM. We measured overheads for the following primitive protocol's operations:

1. Primitive I: time to produce asymmetric cryptography keys. This can be performed during the connection establishment phase or in advance.
2. Primitive II: time to produce symmetric cryptography key. This can also be performed during the connection establishment phase or in advance.
3. Primitive III: time and CPU utilization to simultaneously encrypt and decrypt voice data packets. This is the normal communication phase where signatures have been verified and secure voice traffic is exchanged between the two parties.

Primitive operations I and II above are always performed, i.e. these steps are performed regardless of the specific application that will be used. Primitive operation III refers only to VoIP applications that are delay and jitter sensitive. Jitter is not affected at all by the application, being a network operation metric; the host application adds only a small fraction of time (tiny lag) at the transmission of the 1st packet in a row, which is negligible as expected, something that was proven in the experiments.

We used Speex to encode a wav stream with 8 KHz sampling rate and 16bits sample size (128Kbps rate) into an 8 KHz and 16Kbps encoded voice stream which is acceptable for VoIP communications. The voice payload in each packet is 40 bytes. We performed multiple runs and we calculated the mean times and CPU utilization percentages of the runs. Deviation was negligible to show. Tables 2.7 and 2.8 show the respective measurements for primitive operations I and II (asymmetric and symmetric key generation respectively).

For primitive operation III, we are using four metrics in order to evaluate the performance of the system:

1. Metric M1: time to encode the voice payload of a packet (40 bytes) with Speex.
2. Metric M2: time to decode the voice payload of a packet with Speex.
3. Metric M3: time to encrypt the voice payload of an outgoing packet and simultaneously decrypt the voice payload of an incoming encrypted packet.
4. Metric M4: CPU utilization when performing all of the above tasks simultaneously.

M1 and M2 are fairly simple to calculate. They are based only on codec's characteristics and are independent of the encryption algorithm. Table 2.9 shows the relative values for a 40 byte voice payload. Voice encoding and decoding are functions that are performed in any VoIP session. M3 is of great interest in evaluating our proposal as this is the additive delay due to the VIPSec encryption and decryption operation. Table 2.10 provides the relevant values for M3 and M4.

The voice channel delay budget is not affected significantly by the protocol operation. As it is shown by the experiments (metrics m1 to m4) the one way delay sums up to less than 1msec. The time to produce asymmetric cryptography keys is fairly small, well below 1 sec (0,53 sec is the maximum key generation time for a 4096 bits DH key) whereas the time to produce symmetric keys is less than 1 msec. As stated in Section V, even if asymmetric keys generation times can be significantly larger for highly loaded CPUs, the pre-generated asymmetric keys database addresses effectively with this issue. The experimental results show that runtime overheads of the proposed architecture are minimal and the implementation analysis is both functional and effective.

Table 2.7: Time to generate asymmetric cryptography keys

Algorithm	**Asymmetric key generation**	
	Key size (bits)	*Time (msec)*
DH	1024	24.448
	2048	98.868
	4096	525.254

Table 2.8: Time to generate symmetric cryptography keys

Algorithm	**Symmetric key generation**	
	Key size (bits)	*Time (msec)*
Blowfish	128	0.645
	160	0.683
	256	0.720

Table 2.9: Time to encode/decode a voice packet

Packet Voice Payload	**Metrics**	
	M1 *Packet encode time*	*M2* *Packet decode time*
40 bytes	0.707 msec	0.082 msec

Table 2.10: Time and processing overheads during encrypted VoIP Sessions

Algorithm	**Key size (bits)**	**Metrics**	
		M3 *Time to encrypt & decrypt a 40-byte packet (msec)*	*M4* *CPU utilizatio n (%)*
Blowfish	128	0.072	12
	160	0.075	17
	256	0.076	20

2.8. Implementation

We have implemented a ViDPSec-enabled device pairing application for secure message exchange between any two display equipped devices [81]. Java [42] was selected as the implementation language since it is powerful, light-weight and it is widely supported by any hardware device of concern. The application can support two modes:

The server mode — When run in server mode, the application is ready to accept incoming connections from other devices running the application in client mode.

The client mode — When run in client mode, the application sends connection requests to the server.

At first, applications run on both ends and the server is initialized, as shown in Fig. 2.8 and Fig. 2.9 below. Server's IP address for our example is 195.251.18.165. Following, a connection request is sent to the server and both ends are running the ViDPSec protocol.

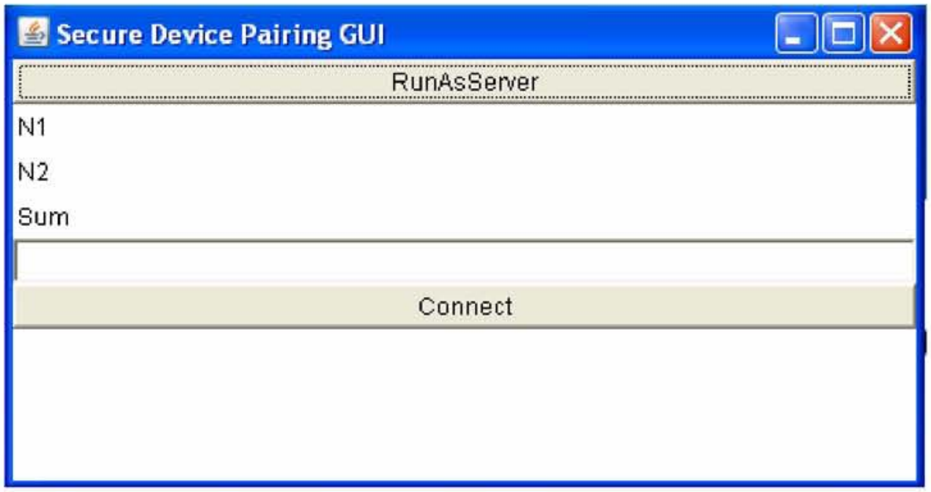


Figure 2.8: Client window

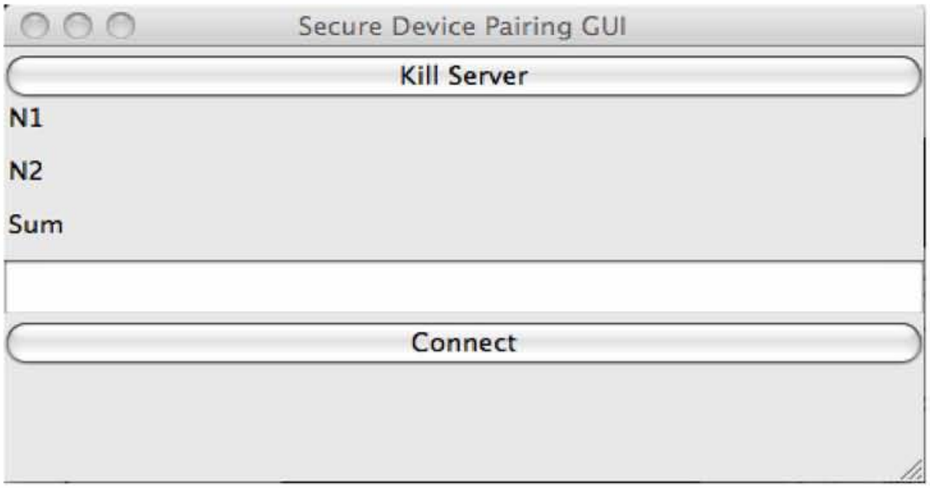
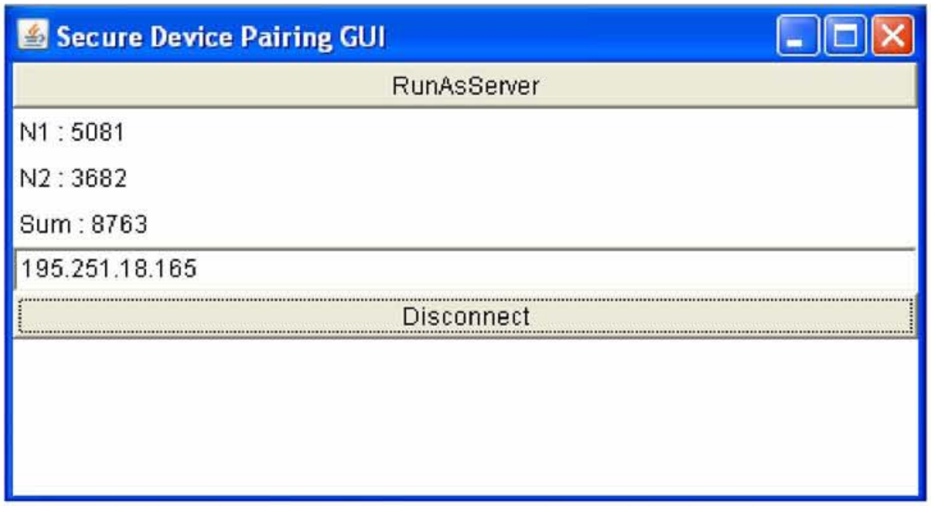


Figure 2.9: Server window (server IP address: 195.251.18.165)

Each device produces a random number along with a pair of asymmetric cryptography keys and the server also produces a Symmetric key. Each side encrypts the random number generated with its own Private Key producing the Device's Session Signature; next they exchange their signatures and their Public keys, they extract the random numbers out of the DSSs and each side computes its perceived sum, as depicted in Fig. 2.10 and Fig. 2.11.



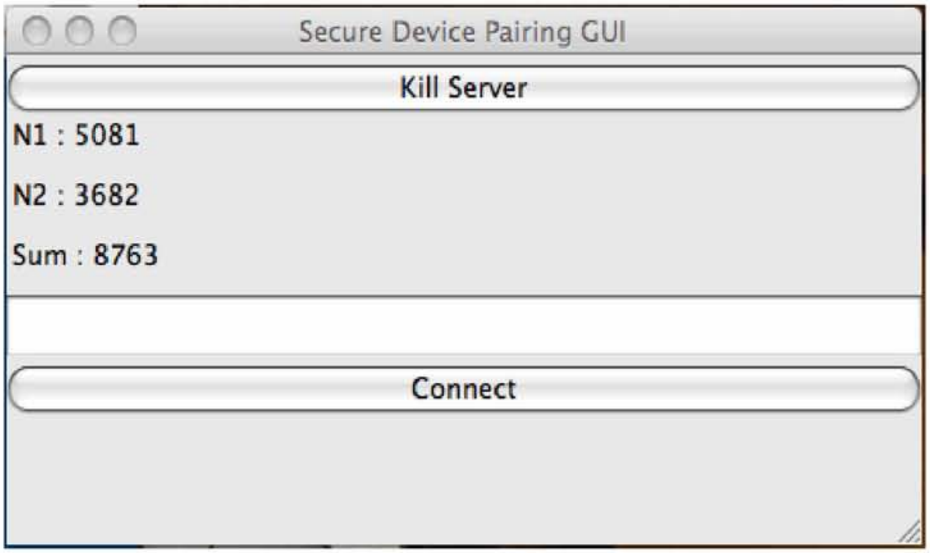
N1 : 5081

N2 : 3682

Sum : 8763

195.251.18.165

Figure 2.10: Client connect request



N1 : 5081

N2 : 3682

Sum : 8763

Figure 2.11: Server connect request

The user is then prompted for verification on both devices as to whether the two perceived sums are equal, as illustrated in Fig. 2.12 and Fig. 2.13. If the two sums are equal, then the user selects “Verify” on both devices and the forthcoming data exchange is encrypted using the Symmetric key as exchanged by the protocol in accordance with Fig.1. In case the two perceived sums are different the user has the option to cancel the pairing as a MITM attack has taken place.

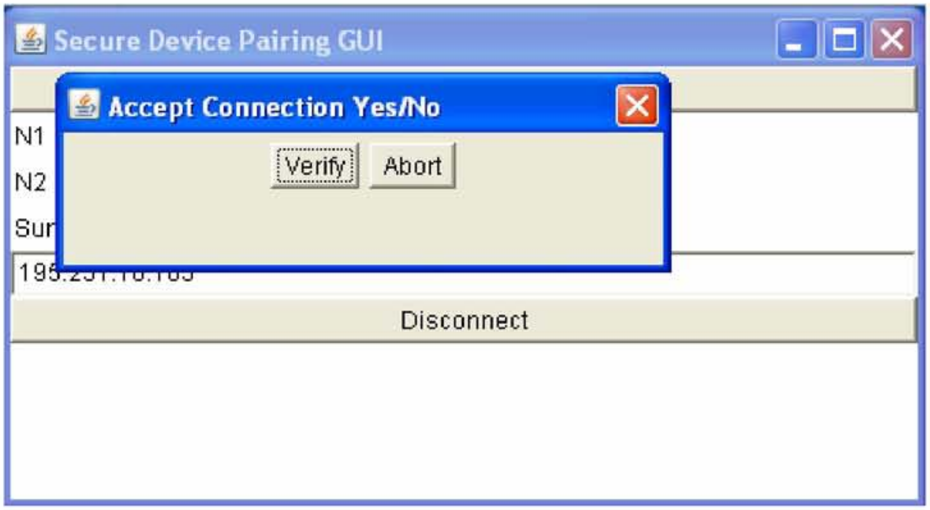


Figure 2.12: Client verification window

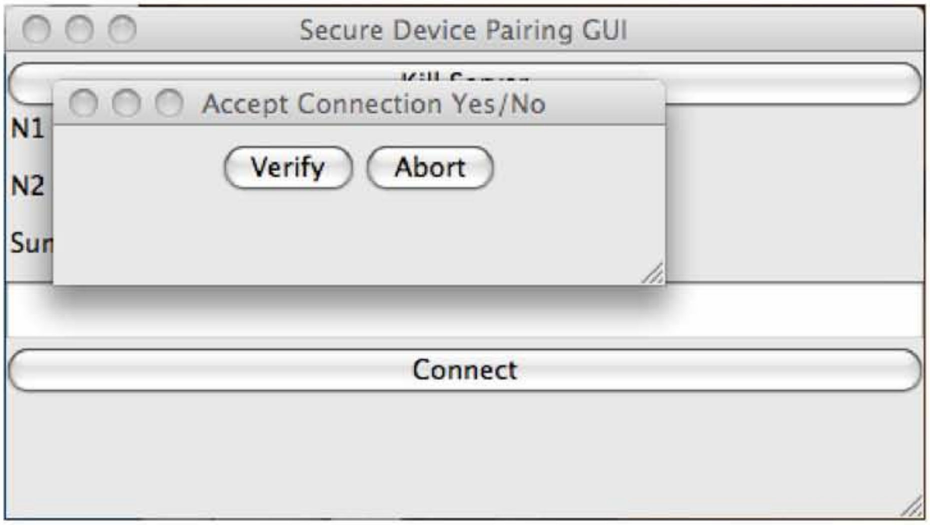
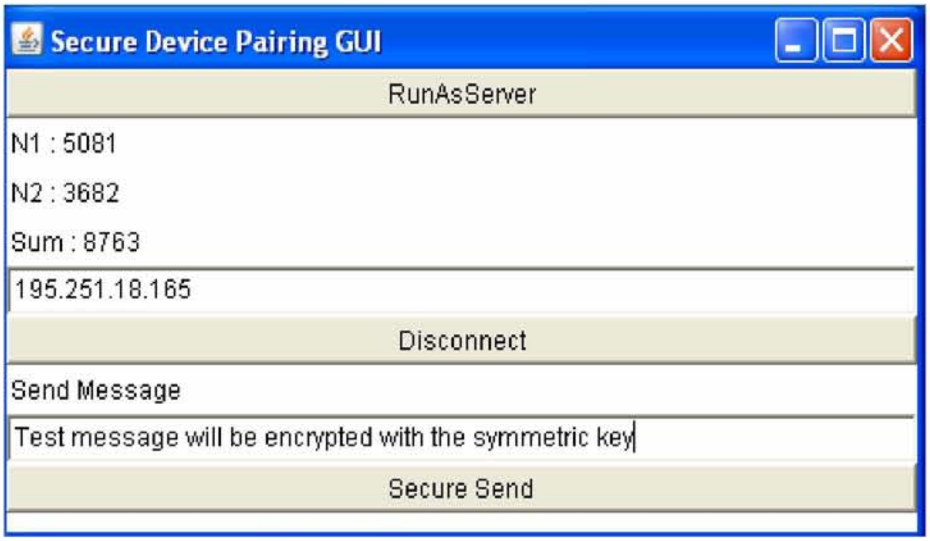


Figure 2.13: Server verification window

After selecting "Verify" at both ends, a "secure send" box is enabled at the client for testing purposes; an accompanying communication text window is opened at the server, that has two parts: at the top part the received encrypted text from the client is displayed, whereas at the bottom part the clear text message as decrypted using the Session Symmetric Key is displayed. The application windows are shown in Fig. 2.14 and Fig. 2.15 below.



N1 : 5081

N2 : 3682

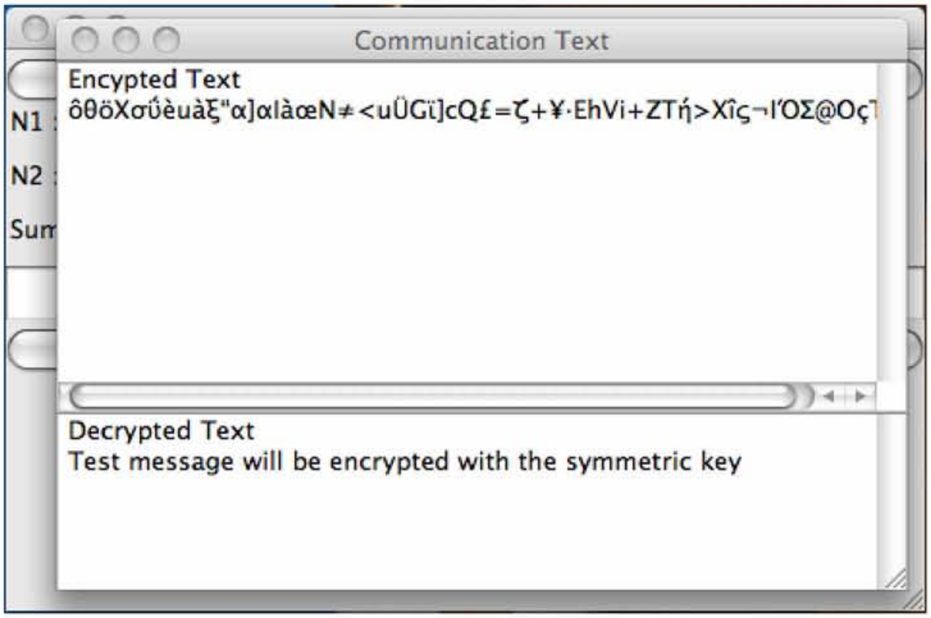
Sum : 8763

195.251.18.165

Send Message

Test message will be encrypted with the symmetric key

Figure 2.14: Client secure data exchange (send)



Encypted Text

ôθöXσΰèuàΞ"α]αlàœN≠<uÜGï]cQ£=ζ+¥·EhVi+ZTή>Xîç¬I'OΣ@Oç

Decrypted Text

Test message will be encrypted with the symmetric key

Figure 2.15: Server secure data exchange (receive)

The application has been tested on and among various hardware platforms:

- PC running windows XP.
- Mac running Leopard.
- PDA running Windows CE 5.0.

As the application is built in Java, it can be easily ported to any device, as Java is widely supported. The requirements on the device, except from the Java support, are that it must be equipped with a display and it must have adequate processing power, e.g. to play mp3 music.

2.9. Conclusion

The method proposed here is a user oriented approach for ensuring secure device pairing, either for devices close together (ViDPSec) or connected via multiple service networks (VIPSec). It relies entirely on the user with no further assumptions (reliable ISPs, PKI etc.). It uses a simple yet effective set of messages exchanged over the channel. It is easy to use because of the human oriented attributes and procedures that are followed (random data, voice and video). Key sizes are user selectable depended on desired security, the default value for normal public use being the minimum accepted levels i.e. 128 bits for the symmetric key and 2048 bits for the asymmetric keys. Furthermore, typical end user terminals today easily meet its computational requirements, making it possible to be used in any environment: from a PC connected to a wireline IP network to a handheld device for use in a wireless IP network. It has been demonstrated that the method is resistant to Man in the Middle attacks, the most sophisticated attacks today's public IP networks suffer from. The protocol is not only resistant to attacks known to compromise today's voice networks, land line or wireless, but also to hypothetical more powerful attacks, like an attack from the network itself. Given the relatively low complexity for implementing the protocol at the end terminals, its effectiveness and user friendliness we expect it will be an attractive candidate solution for ensuring secure device pairing.

Experimental analysis shows that typical end user terminals easily meet its requirements, making it possible to be implemented. Furthermore, by optimizing the application performance its global use can be further ensured, since the device requirements can get even lower.

Chapter 3

Introduction

Handling multiple sets of trouble tickets (TTs) originating from different participants in today's GRID interconnected network environments poses a series of challenges for the involved institutions. Each of the participants follows different procedures for handling trouble incidents in its domain, according to the local technical and linguistic profile. The TT systems of the participants collect, represent and disseminate TT information in different formats. As a result, management of the daily workload by a central Network Operations Centre (NOC) is a challenge on its own. Normalization of TTs to a common format for presentation and storing at the central NOC is mandatory. In the present work we provide a model for automating the collection and normalization of the TT received by multiple networks forming the Grid. Each of the participants is using its home TT system within its domain for handling trouble incidents, whereas the central NOC is gathering the tickets in the normalized format for repository and handling. Our approach is using XML as the common representation language. The model was adopted and used as part of the EGEE project.

3.1. Grid management

Modern telecommunications networks are aimed to provide a plethora of differentiated services to its customers. Networks are becoming more sophisticated by the day, while their offering spans a wide variety of customer types and services. Quality of Service (QoS) [43] and Service Level Agreement (SLA) [44] provisioning are fundamental ingredients.

Multiple interconnected institutions, targeting a common approach to service offering, along with a unified network operation scheme to support these services, form Grid networks. Network Management is crucial for the success of the Grid. Problem reporting, identification and handling as well as trouble information dissemination and delegation of authority are some of the main tasks that have to be implemented by the members of the Grid.

GÉANT2 [45] is an example of a Grid. It is the seventh generation of pan-European research and education network successor to the pan-European multi-gigabit research network GÉANT. The GÉANT2 network connects 34 countries through 30 national research and education networks (NRENs), using multiple 10Gbps wavelengths. GÉANT2 is putting user needs at the forefront of its plans for network services and research.

Usually a central Network Operations Centre (NOC) is established at the core of the network for achieving network and service integration support. Ideally, a uniform infrastructure should be put in place, with interoperating network components and systems, in order to provide services to the users of the Grid and to manage the network. In practice though, this is not the case. Unfortunately, different trouble ticket (TT) systems are used by the participating networks.

There is a wide variety of TT systems available, with differentiated functionality and pricing (respectively) among them. Examples are Keystone [46], ITSM [47], HEAT [48], SimpleTicket [49], OTRS [50]. Moreover, an in-house developed systems, as is the case for GRNET [51], is another option. The advantages of this option are that it offers freedom in design and localization options and that it meets the required functionality in full. It has though the disadvantage of local deployment and maintenance. Nevertheless, it is adopted by many Internet Service Providers (IPSs), both academic and commercial, as the pros of this solution enable service delivery and monitoring with total control over the network.

The current work [82] [83] evolved within the specific Service Activity 2 (SA2) activity of the EGEE European funded project [52]. A central NOC, called the ENOC [53] is responsible for collecting and handling multiple TTs received by the participating institutions TT systems. Various TT systems are used by each of them, delivering TTs in different formats, while TT load is growing proportionally with the network size and the serviced users. TT normalization, i.e. transformation to a common format that is reasonable for all parties and copes with service demands in a dynamic and effective way, is of crucial importance for successful management of the Grid.

In the present work we define a data model for TT normalization for the participating institutions in EGEE. The model is designed in accordance with the specific needs of the participants, meeting requirements of the multiple TT systems used. It is both effective and comprehensive, as it compensates for the core activities of the NOCs. It is also dynamic as it allows other options to be included in the future, according to demand.

3.2. Related work

Whenever multiple organizations and institutions form a Grid, or some other form of cooperative platform for network service deployment, the need arises to define a common understanding over network operations and management issues. Trouble incidents are recorded in case a problem arises, affecting normal network operations or services. Typical problems are failures in network links or other network elements (i.e. routers, servers), security incidents (i.e. intrusion detection) or any other problem that affects normal service delivery (i.e. service overload). The incidents are represented in specific formats, namely TTs. ATT is issued in order for the network operators to record and handle the incident.

RFC 1297 [54], titled NOC Internal Integrated Trouble Ticket System Functional Specification Wishlist, describes general functions of a TT system that could be designed for NOCs, exploring competing uses, architectures, and desirable features of integrated internal trouble ticket systems for Network and other Operations Centres.

Network infrastructure available to EGEE is served by a set of National Research and Education Networks (NRENs) via the GÉANT2 network. Reliable network resource provision to Grid infrastructure highly depends on coherent collaboration between a large numbers of different parties both from NREN/ GÉANT2 and EGEE sides, as described in [55]. Common problems and solutions as well as strategies for investigating problem reports has been presented in [56] [57]. The concept of the Multi-Domain Monitoring (MDM) service, which describes the transfer of end-to-end monitoring services in order to serve the needs of different user groups is discussed in [58].

The OSS Trouble Ticket API (OSS/J API) [59] provides interfaces for creating, querying, updating, and deleting trouble tickets (trouble reports). The Trouble Ticket API focus is on the application of the Java 2 Platform, Enterprise Edition (J2EE), and XML technologies to facilitate the development and integration of OSS components with Trouble Ticket Systems. The Incident Object Description Exchange Format (IODEF) [60] constitutes a format for representing computer security information commonly exchanged between Computer Security Incident Response Teams (CSIRTs). It provides an XML representation for conveying incident information across administrative domains between parties that have an operational responsibility of remediation or a watch-and-warning over a defined constituency. The data model encodes information about hosts, networks, and the services running on these systems; attack methodology and associated forensic evidence; impact of the activity; and limited approaches for documenting workflow.

The EGEE project is heavily using shared resources spanning across more than 45 countries and involving more than 1600 production's hosts. To link these resources together the network infrastructure used by EGEE is mainly served by GÉANT2 and NRENs. NRENs are providing link to sites within a country while GÉANT2, the seventh generation of pan-European research and education network, connects countries. To link Grid and network worlds the ENOC, EGEE Network Operation Centre, has been defined in EGEE as the operational interface between the EGEE Grid, GÉANT2 and the NRENs to check the end-to-end connectivity of Grid sites. Using daily relations with all providers of the network infrastructures on top of which EGEE is built it ensures the complex nexus of domains involved to link Grid sites are performing efficiently. The ENOC deals with network problems troubleshooting, notifications from NRENs, network Service Level Agreement (SLA) installation and monitoring and network usage reporting. The ENOC acts as the network support unit in the Global Grid User Support (GGUS) of EGEE to provide coordinated user support across Grid and network services.

In the next section we describe the trouble ticket harmonization system that was adopted by the EGEE parties for TT normalization. In Appendix A we present the Network Trouble Ticket Data Model in detail.

3.3. Trouble ticket harmonization system

XML [67] was the choice for the implementation schema, due to its powerful mechanisms and its global acceptance. The implemented system operates as depicted in Fig. 3.1 below.

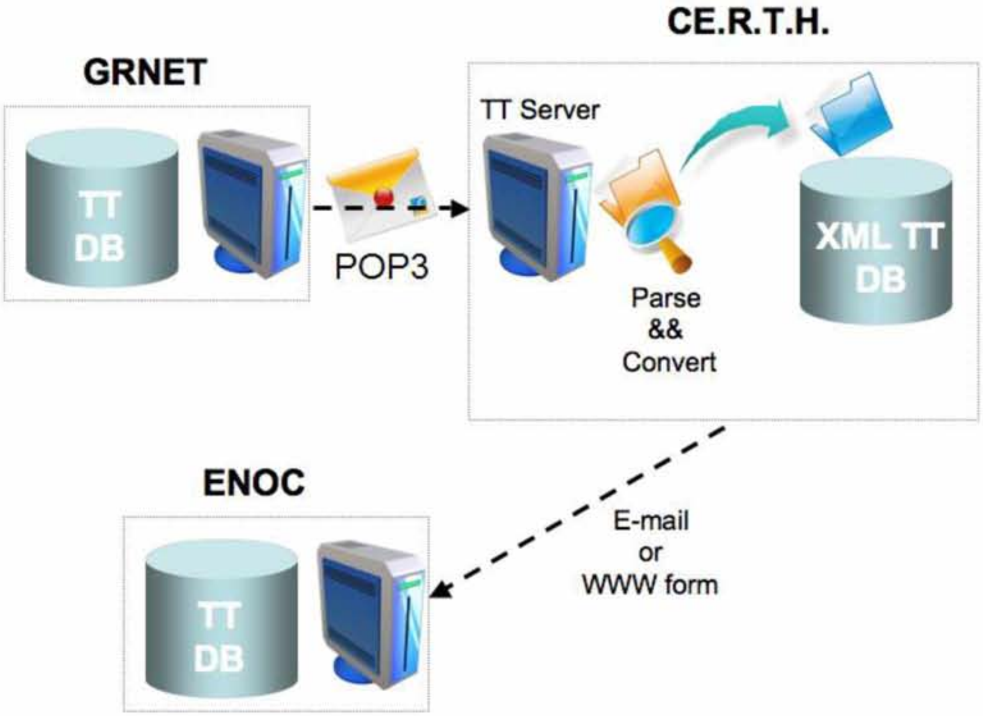


Figure 3.1: Ticket normalization system.

Our system connects to GRNET ticketing system and uses POP e-mail to download the TTs. Following, it converts the TTs according to the data model presented, stores them in a database and finally sends them to ENOC via e-mail to a specified email address. More options are available:

- TTs can be sent via http protocol to a web service or a form.
- TTs can be stored to another database (remote).
- TTs can be sent via email in XML format (not suggested since the XML format is not human readable).

An SMS send option (to mobile phones) is under consideration, since this proves to be vital in case of extreme importance. For this option to work, an SMS server needs to be provided. Linguistic issues are also considered, in order to ease understanding of all fields in a TT, i.e. Greek to English translation needs to be performed for some predefined fields, like TT Type.

Our system offering improves security: most web forms use ASP, PHP or CGI to update the database or perform some other action. This is inherently insecure because the database needs to be accessible from the web server. Our system offers a number of advantages:

- The email can be sent to a completely separate and secure PC.
- Our system can process the email without ever needing access to the web server or be accessible from the Internet.
- If a PC or network connection is down the emails will sit waiting. Our system will 'catch up' when the PC or network is back up.

Moreover we offer increased redundancy: when using web forms to update back-end databases, the problem always arises about what to do if the database is down. Our system resolves this problem, because the email will always get sent. In case the database cannot be updated our system will wait and process the email later

3.4 Conclusions

In the present work, a common format for normalizing trouble tickets from the various regional networks (NRENs) participating in the Grid, implemented in the EGEE project framework, has been designed and implemented. XML was used to represent the common trouble ticket format. The adopted transformation schema is lightweight yet effective and is accepted by all participating partners and the research community. The solution has passed testing and it is already in use, proving to be a valuable tool for handling incidents taking place during the operation of the Grid.

The defined data model has proved to be both efficient and dynamic. It has been submitted as an IEEE Internet draft well ago and has not received any negative comments. It now undergoes review of the RFC editor to become Internet RFC.

Chapter 4

Conclusions

This thesis addresses the issues of security and network management in modern telecommunications networks. For the security aspect, we have focused on user aided secure device pairing among devices. We exploited the various options for doing so and we produced a unique procedure to securely pair two devices, regardless the distance in between, based on the user.

More specifically, we have defined a biometric method for exchanging cryptographic material over the media path and establishing a symmetric key to encrypt the forthcoming communication. By having the user an integral part of the exchange process, through the verification of the exchanged cryptographic elements, we eliminate the need for a trusted third party or for the assumption of a "safe" or "honest" network service provider. Biometry in our case is applied twofold: through the user selected objects to be exchanged as well as through the human attributed verification. The method is in fact a 3-phase handshake procedure among the two devices; at first user selected data and one time asymmetric cryptographic elements are used to create the session signatures, which are then exchanged along with the public parts of the cryptography keys; at the end all the user has to do is to verify the accuracy of the initial user data. The method is simple, easy to implement, user friendly and lightweight. We have formalized the method both for device proximity (ViDPSec) as well as for devices interconnected through public operator networks like the Internet (VIPSec). We have analyzed the overhead of its use and we have developed an implementation of ViDPSec. We have also patented VIPSec as its potential is quite high.

Communications security is an ever important area of research and development. As a future work our method can be extended to support any kind of personal or group communications. Exploitation of our method on various devices and network types is a possible future research direction; implementation on mobile phones over GSM/UMTS/3G is very promising. VoIP is a research area where the exploitation of our method could have significant impact; therefore VoIP is considered as a major future development.

As far as network management is concerned, we elaborated on a form of interconnected networks for the research community and the academia, the Grids. A Grid is a virtual network formed over a number of other networking domains, established to provide a set of services to its users over resources that span all participating network entities. Being so, the Grid itself is based upon its entities for its core operation, i.e. each domain’s links, routers, servers and so on. The services that the Grid offers are different than those of the independent networks and they have to be provisioned in accordance to a unified approach for deployment, delivery and support across the Grid. Identifying trouble incidents as well as communicating and repairing the troubles are a crucial factor for the operation of the Grid.

We have designed a model for automating the collection and normalization of the trouble tickets produced by the independent network domains that form the Grid. Each of the participants is using its home trouble ticket system within its domain for handling trouble incidents, whereas the central Network Operation Centre is gathering the tickets in the normalized format for repository and handling. Our approach is using XML as the common representation language. The model was adopted and used as part of the SA2 activity of the EGEE project. It has been also posted as an IEEE Internet draft.

The use of our proposal to the EGEE network is already a great achievement. For the future, evaluating the impact of the application would prove extremely useful.

Appendix A

Definition of the data model

A.1. General Considerations

A.1.1. Terminology

NTTDM uses specific keywords to describe the various data components. These keywords are: Defined, Free, Multiple, List, Predefined String, String, Datetime, Solved, Cancelled, Inactive, Superseded, Opened/Closed, Operational, Informational, Administrative, Test. Those in this document are to be interpreted as described in Section 2.

A.1.2. Notations

The NTTDM is specified in two ways, as an abstract data model and as an XML Schema. Section 3 provides a Unified Modeling Language (UML) [61] model describing the individual classes and their relationship with each other. The semantics of each class are discussed and their attributes explained. In Section 6, this UML model is converted in an XML Schema [62][63][64][65]. A specific namespace [66] is also defined.

The term "XML document" refers to any instance of an XML document. The term "NTTDM document" refers to specific elements and attributes of the NTTDM schema. Finally, the terms "class", and "element" will be used interchangeably to reference either a given UML class in the data model or its corresponding schema implementation.

A.1.3. About the Network Trouble Ticket Data Model

The NTTDM is a data representation that provides a framework for normalizing and sharing information among network operators and the GOC regarding troubles within the Grid boundaries. There has been a lot of thought processing during the design of the data model:

- The data model serves as a common storage and exchange format.
- Every NOC still uses its home TT system for network management within its area of control.
- As there is no universally adopted definition for a trouble, in the NTTDM definition the term is used with a comprehensive meaning to cover all NOCs.
- Handling every possible definition of a trouble incident would call for an extremely expanded and complex data model. Therefore, the NTTDM purpose is to serve as the basis to normalize and exchange TTs. It is flexible and expressive in order to ensure that specific NOC requirements are met. Specific NOC information is kept outside the NTTDM and external databases can be used to feed it.
- The domain of managing the information is not fully standardized and must rely on free-form textual descriptions. The NTTDM attempts to strike a balance between supporting this free-form content, while still allowing automated processing of incident information.

The NTTDM is only one of feasible TT data representations. The goal of this design was to be as effective and comprehensive to cover for the management of a general grid environment. The already used TT formats influenced the design of the NTTDM.

A.1.4. About the Network Trouble Ticket Implementation

Here we describe an example of typical use case. The Grid project EGEE manages its infrastructure as network overlay over the European NRENs and want to be able to warn EGEE sites of the unavailability of the network. Thanks to collaboration with its network provider the EGEE NOC receive TTs (800 tickets/month, 2500 emails/month) from 20 NRENs and should be able to cope with the heavy TT process. Thanks to the NTTDM the EGEE NOC can automate the TT workflow:

- TT is filtered, sorted and stored in local DB.
- TT impact on the Grid is assessed.
- TT is pushed to dashboard and other tools (EGEE TT system, statistics, etc.)

A.2. NTTDM types and definitions

The various data elements of the TT data model are typed. This section discusses these data types. When possible, native Schema data types were adopted, but for more complicated formats, regular expressions or external standards were used.

A.2.1. Types and Definitions for the TYPE attributes

These types are used to describe the TYPE attributes.

A.2.1.1 Defined

The Defined data type means that the data model provides a mean to compute this value from the rest of the fields.

The Defined data type is implemented as a "Defined" in the schema.

A.2.1.2. Free

The Free data type means that the value can be freely chosen.

All Free string should have as an attribute the language used.

The Free data type is implemented as a "Free" in the schema.

A.2.1.3. Multiple

The Multiple data type consists of one value among multiple fixed values.

The Multiple data type is implemented as a "Multiple" in the schema.

A.2.1.4. List

List means many values among multiple fixed values.

The List data type is implemented as a "List" in the schema.

A.2.2. Types and Definitions for the VALID FORMAT attributes

A.2.2.1. Predefined String

A Predefined String means the different values are predefined in the data model.

Each field that requires a Predefined String contains a specific value. Table 3.1 below shows the allowed values for such fields.

TABLE A.1: The allowed Predefined String values

FIELD NAME	VALUES
TT_TYPE	Operational, Informational, Administrative, Test
TYPE	Scheduled, Unscheduled
TT_PRIORITY	Low, Medium, High
TT_SHORT_DESCRIPTION	Core Line Fault, Access Line Fault, Degraded Service, Router Hardware Fault, Router Software Fault, Routing Problem, Undefined Problem, Network congestion, Client Upgrade, IPv6, QoS, VoIP, Other
TT_IMPACT_ASSESSMENT	No impact, Reduced redundancy, Minor performance impact, Severe performance impact, No connectivity, On backup, At risk, Unknown
TT_STATUS	Opened, Updated, Closed, Solved, Inactive, Cancelled, Reopened, Superseded
TT_SOURCE	Users, Monitoring, Other NOC

The Predefined String data type is implemented as an "xs:string" in the schema with a sequence of enumerations for the allowed values.

A.2.2.1.1. Definitions of the Predefined Values

TT_TYPE

- Operational: for network incident & maintenance only.
- Informational: Information about the TT system or the exchange interface (maintenance, upgrade).
- Administrative: Information about the access to the TTS (credentials) or the exchange interface.
- Test: to test the TT system or the exchange interface, etc.

TYPE

- Scheduled: the incident was scheduled to happen.
- Unscheduled: the incident was unscheduled.

TT_PRIORITY

- Low: the TT priority is low.
- Medium: the TT priority is medium.
- High: the TT priority is high.

TT_SHORT_DESCRIPTION

- Core Line Fault: malfunction of a high bandwidth Core line.
- Access Line Fault: malfunction of a medium bandwidth Access line.
- Degraded Service.
- Router Hardware Fault: malfunction of the router hardware.
- Router Software Fault: malfunction of the router software.
- Routing Problem: incident regarding the routing service.
- Undefined Problem: the nature of the problem is not identified.
- Network congestion: problem due to traffic at the network (blocked).
- Client Upgrade: incidents regarding clients/services upgrade.
- IPv6: incident regarding the IPv6 network.
- QoS: incident regarding the QoS of the network.
- VoIP: incident regarding VoIP.
- Other: non listed incident.

TT_IMPACT_ASSESSMENT

- No impact: the incident does not cause any impacts.
- Reduced redundancy: the incident produces reduction at the redundancy.

- Minor performance impact: the incident causes a minor performance impact.
- Severe performance impact: the incident causes a severe performance impact.
- No connectivity: the incident causes failure of connectivity.
- On backup: the incident produces malfunction on backup services.
- At risk: the incident should not have any impact but possibly it may cause some trouble.
- Unknown: the nature of the impact is not identified.

TT_STATUS

- Opened: the ticket is opened.
- Closed: the ticket is closed.
- Updated: the ticket's contents have been updated.
- Cancelled: the ticket has been opened twice, one of the both tickets is cancelled and a relation is done between them via RELATED_ACTIVITY.
- Solved: the incident is solved but the team prefers to monitor for check.
- Opened/closed stands for tickets that are opened only to report an incident that is already solved.
- Inactive: the ticket is under the responsibility of an external domain and is no more under the domain control.
- Reopened: the ticket was closed by error, or the problem was faulty declared solved. Historical data are very important at this case.
- Superseded: the ticket has been superseded by another one (case of a bigger problem having raised many tickets and being merged in one single incident). The RELATED_ACTIVITY field should include the master ticket reference.

Allowed transitions for TT_STATUS are only those illustrated in Fig. 3.1. Possible final states are indicated with (X).

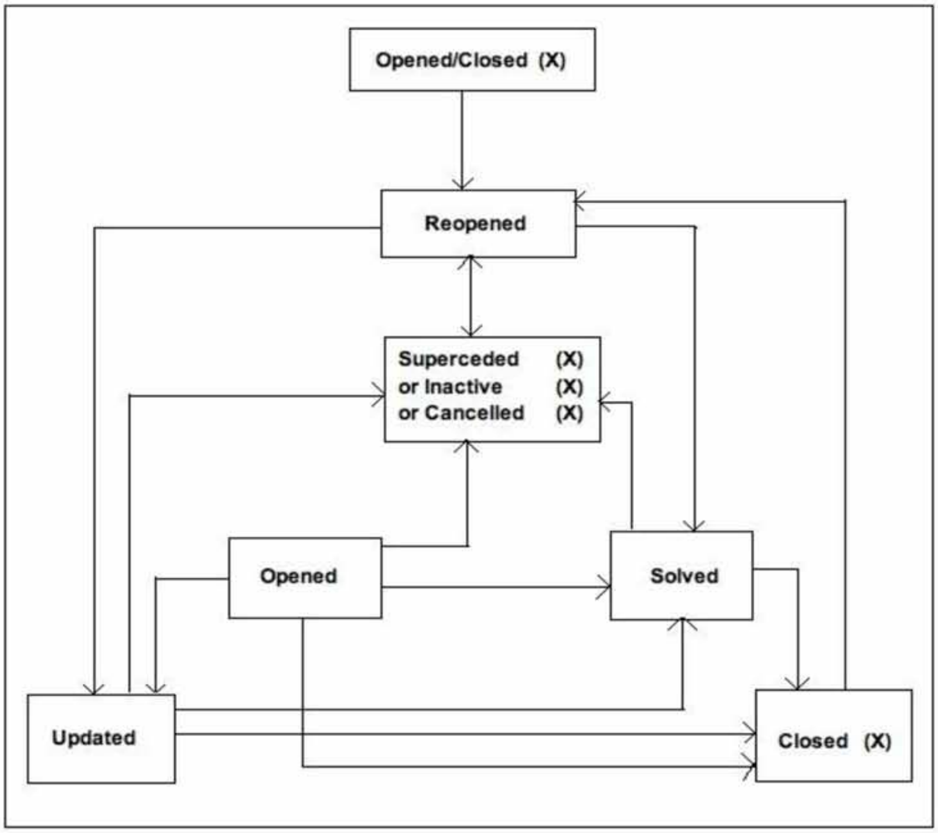


Figure A.1: TT_STATUS transition diagram

A.2.2.2. String

The String value is defined by the user of the model.

The String data type is implemented as an "xs:string" in the schema.

A.2.2.3. Datetime

Date-time strings are represented by the Datetime data type.

Each date-time string identifies a particular instant in time; ranges are not supported.

Date-time strings are formatted according to a subset of ISO 8601: 2000 documented in RFC 3339.

The Datetime data type is implemented as an "xs:dateTime" in the schema.

A.3. NTTDM

In this section, the individual components of NTTDM will be discussed in detail. This class provides a standardized representation for commonly exchanged Field Name data.

A.3.1 NTTDM Components

A.3.1.1 NTTDM Attributes

The Field Name class has four attributes. Each attribute provides information about a Field Name instance. The attributes that characterize one instance constitute all the information required to form the data model.

DESCRIPTION: This field contains a short description of the field name.

TYPE: The TYPE attribute contains information about the type of the field name it depends on. The values that it may contain are: Defined, Free, Multiple, List.

VALID FORMAT: This attribute contains information about the format of each field. The values that it may contain are: PREDEFINED STRING, STRING, DATETIME.

MANDATORY: This attribute indicates if the information of each field is required or is optional. In case the information is required the field MANDATORY contains the word: "YES". On the contrary, when filling the information is optional, the field MANDATORY contains the word "NO".

A.3.2 NTTDM Aggregate Classes

A.3.2.1 NTTDM-Document class

The NTTDM-Document class is the top-level class in NTTDM. All NTTDM documents are an instance of this class.

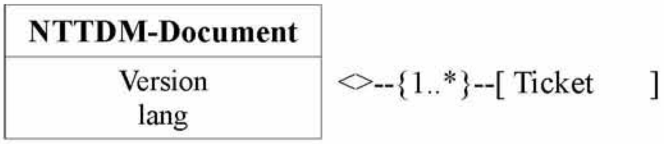


Figure A.2: NTTDM-Document class

The aggregate class that constitute NTTDM-Document is:

- Ticket: One or more. The information related to a single ticket.

The NTTDM-Document class has two attributes:

1. version

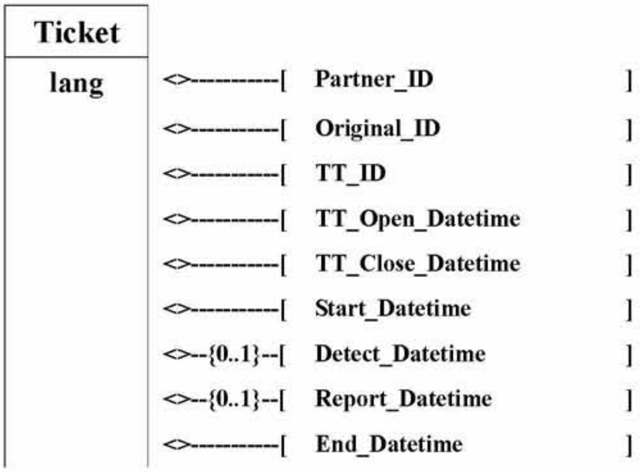
   STRING. The value of this attribute MUST be "1.00"

2. lang

   Required.

A.3.2.2 Ticket class

Every ticket is represented by an instance of the Ticket class. This class provides a standardized representation for commonly exchanged TT data.


Ticket
lang
<>----------[Partner_ID]
<>----------[Original_ID]
<>----------[TT_ID]
<>----------[TT_Open_Datetime]
<>----------[TT_Close_Datetime]
<>----------[Start_Datetime]
<>--{0..1}--[Detect_Datetime]
<>--{0..1}--[Report_Datetime]
<>----------[End_Datetime]

```
<>----------[   TT_Last_Update_Time          ]
<>--{0..1}--[   Time_Window_Start            ]
<>--{0..1}--[   Time_Window_End              ]
<>--{0..1}--[   Work_Plan_Start_Datetime     ]
<>--{0..1}--[   Work_Plan_End_Datetime       ]
<>----------[   TT_Title                     ]
<>----------[   TT_Short_Description         ]
<>--{0..1}--[   TT_Long_Description          ]
<>----------[   Type                         ]
<>----------[   TT_Type                      ]
<>----------[   TT_Impact_Assessment         ]
<>--{0..1}--[   Related_External_Tickets     ]
<>----------[   Location                     ]
<>--{0..1}--[   Network_Node                 ]
<>--{0..1}--[   Network_Link_Circuit         ]
<>--{0..1}--[   End_Line_Location_A          ]
<>--{0..1}--[   End_Line_Location_B          ]
<>--{0..1}--[   Open_Engineer                ]
<>--{0..1}--[   Contact_Engineers            ]
<>--{0..1}--[   Close_Engineer               ]
<>--{0..1}--[   TT_Priority                  ]
<>----------[   TT_Status                    ]
<>--{0..1}--[   Additional_Data              ]
<>--{0..1}--[   Related_Activity             ]
<>----------[   History                      ]
<>--{0..1}--[   Hash                         ]
<>--{0..1}--[   TT_Source                    ]
<>--{0..1}--[   Affected_Community           ]
<>--{0..1}--[   Affected_Service             ]
```

Figure A.3: the Ticket class

- lang: Required.

The Field Names are the Aggregate Classes that constitute the NTTDM and each of them is an element that is characterized by a quadruple (DESCRIPTION, TYPE, VALID FORMAT, MANDATORY).

A.3.2.3 Ticket origin information

A.3.2.3.1 PARTNER_ID

PARTNER_ID	
DESCRIPTION	This field contains the unique ID of the TT source partner.
TYPE	This attribute contains a MULTIPLE value.
VALID FORMAT	This attribute contains a STRING value.
MANDATORY	YES. This information is required.

Figure A.4: Partner_ID Class

A.3.2.3.2 ORIGINAL_ID

ORIGINAL_ID	
DESCRIPTION	This field contains the TT ID that was assigned by the party.
TYPE	This attribute contains a Free value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	YES. This information is required.

Figure A.5: Original_ID Class

A.3.2.4 Ticket information

A.3.2.4.1 TT_ID

TT_ID	
DESCRIPTION	This field contains the unique ID of the TT.
TYPE	It is built the following way: "PARTNER_ID"_"ORIGINAL_ID" (*).
VALID FORMAT	This attribute contains a String value.
MANDATORY	YES. This information is required.

Figure A.6: TT_ID Class

(*): PARTNER_ID and ORIGINAL_ID MUST therefore not contain an underscore character.

A.3.2.4.2 TT_TITLE

TT_TITLE	
DESCRIPTION	This field contains the title of the TT.
TYPE	This attribute contains a Defined value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	YES. This information is required.

Figure A.7: TT_Title Class

A.3.2.4.3 TT_TYPE

TT_TYPE	
DESCRIPTION	This field contains the type of the TT.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Predefined String value.
MANDATORY	YES. This information is required.

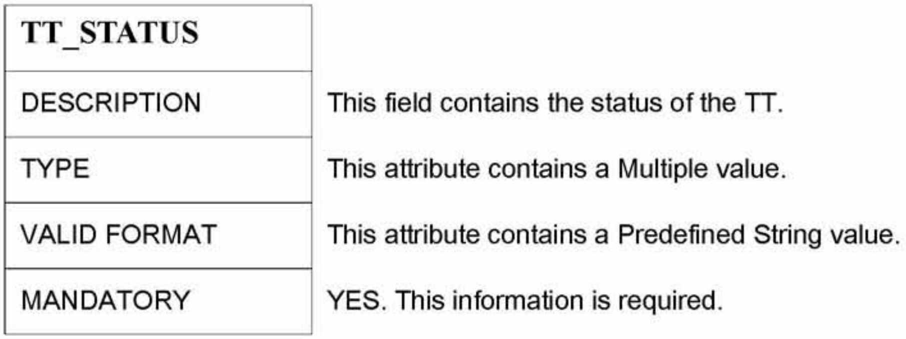
Figure A.8: Type Class

A.3.2.4.4 TT_PRIORITY

TT_PRIORITY	
DESCRIPTION	This field contains the priority of the TT.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Predefined String value.
MANDATORY	NO. This information is optional.

Figure A.9: TT_Priority Class

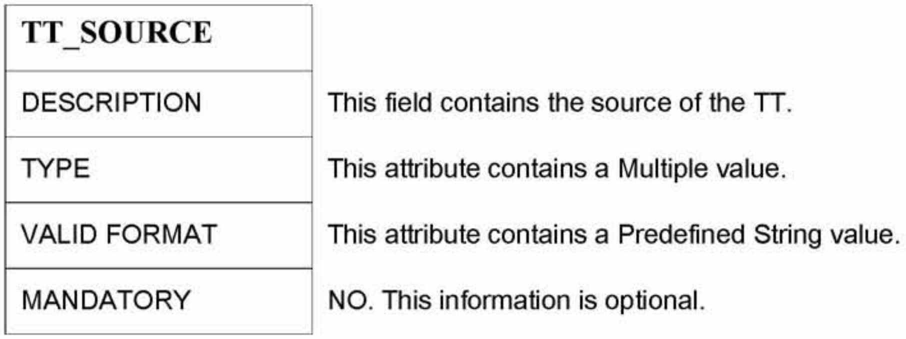
A.3.2.4.5 TT_STATUS



TT_STATUS	
DESCRIPTION	This field contains the status of the TT.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Predefined String value.
MANDATORY	YES. This information is required.

Figure A.10: TT_Status Class

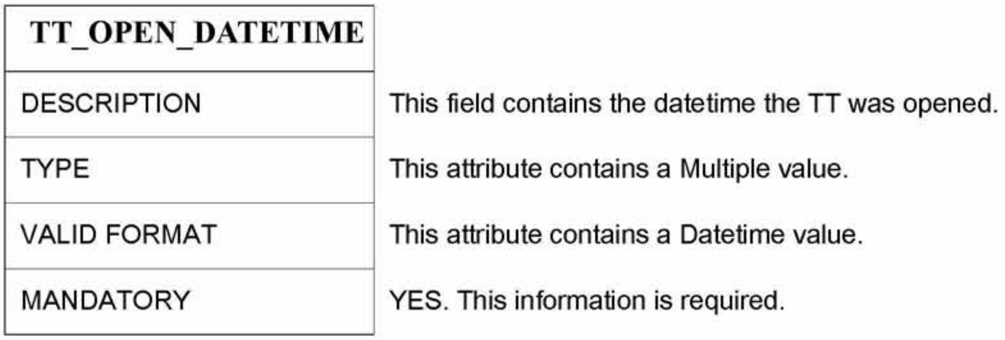
A.3.2.4.6 TT_SOURCE



TT_SOURCE	
DESCRIPTION	This field contains the source of the TT.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Predefined String value.
MANDATORY	NO. This information is optional.

Figure A.11: TT_Source Class

A.3.2.4.7 TT_OPEN_DATETIME



TT_OPEN_DATETIME	
DESCRIPTION	This field contains the datetime the TT was opened.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	YES. This information is required.

Figure A.12: TT_Open_Datetime Class

A.3.2.4.8 TT_CLOSE_DATETIME

TT_CLOSE_DATETIME	
DESCRIPTION	This field contains the datetime the TT was closed.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	YES. This information is required.

Figure A.13: TT_Close_Datetime Class

A.3.2.5 Trouble details

A.3.2.5.1 TT_SHORT_DESCRIPTION

TT_SHORT_DESCRIPTION	
DESCRIPTION	This field contains the short description of the incidence/maintenance reported in the TT.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Predefined String value.
MANDATORY	YES. This information is required.

Figure A.14: TT_Short_Description Class

A.3.2.5.2 TT_LONG_DESCRIPTION

TT_LONG_DESCRIPTION	
DESCRIPTION	This field contains the long description of the incidence/maintenance reported in the TT.
TYPE	This attribute contains a Free value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	YES. This information is required.

Figure A.15: TT_Long_Description Class

A.3.2.5.3 TYPE

TYPE	
DESCRIPTION	This field contains the type of the trouble.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Predefined String value.
MANDATORY	YES. This information is required.

Figure A.16: Type Class

A.3.2.5.4 TT_IMPACT_ASSESSMENT

TT_IMPACT_ASSESSMENT	
DESCRIPTION	This field contains the impact of the incidence/maintenance.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Predefined String value.
MANDATORY	YES. This information is required.

Figure A.17: TT_Impact_Assessement Class

A.3.2.5.5 START_DATETIME

START_DATETIME	
DESCRIPTION	This field contains the datetime that the incident/maintenance started.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	YES. This information is required.

Figure A.18: Start_Datetime Class

A.3.2.5.6 DETECT_DATETIME

DETECT_DATETIME	
DESCRIPTION	This field contains the datetime that the incident was detected.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	NO. This information is optional.

Figure A.19: Detect_Datetime Class

A.3.2.5.7 REPORT_DATETIME

REPORT_DATETIME	
DESCRIPTION	This field contains the datetime that the incident was reported.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	NO. This information is optional.

Figure A.20: Report_Datetime Class

A.3.2.5.8 END_DATETIME

END_DATETIME	
DESCRIPTION	This field contains the datetime that the incident/maintenance ended.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	YES. This information is required.

Figure A.21: End_Datetime Class

A.3.2.5.9 TT_LAST_UPDATE_TIME

TT_LAST_UPDATE_TIME	
DESCRIPTION	This field contains the datetime that the TT was updated.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	NO. This information is optional.

Figure A.22: TT_Last_Update_Time Class

A.3.2.5.10 TIME_WINDOW_START

TIME_WINDOW_START	
DESCRIPTION	This field contains the window start time in which planned maintenance may occur.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	NO. This information is optional. (*)

Figure A.23: Time_Window_Start Class

(*): However, it is mandatory if the TYPE element contains the string "Scheduled".

A.3.2.5.11 TIME_WINDOW_END

TIME_WINDOW_END	
DESCRIPTION	This field contains the window end time in which planned maintenance may occur.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	NO. This information is optional. (*)

Figure A.24: Time_Window_End Class

(*): However, it is mandatory if the TYPE element contains the string "Scheduled".

A.3.2.5.12 WORK_PLAN_START_DATETIME

WORK_PLAN_START_DATETIME	
DESCRIPTION	This field contains the work planned (expected) start time in case of maintenance.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	NO. This information is optional.

Figure A.25: Work_Plan_Start_Datetime Class

A.3.2.5.13 WORK_PLAN_END_DATETIME

WORK_PLAN_END_DATETIME	
DESCRIPTION	This field contains the work planned (expected) end time in case of maintenance.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a Datetime value.
MANDATORY	NO. This information is optional.

Figure A.26: Work_Plan_End_Datetime Class

Note: The period delimited by WORK_PLAN_START_DATETIME and WORK_PLAN_END_DATETIME must be included in the period delimited by TIME_WINDOW_START and TIME_WINDOW_END, duplicated with {START, END}_DATETIME, even in case of maintenance.

A.3.2.6 Related data

A.3.2.6.1 RELATED_EXTERNAL_TICKETS

RELATED_EXTERNAL_TICKETS	
DESCRIPTION	This field contains the NOC entity related to the incident.
TYPE	This attribute contains a List value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.27: Related_External_Tickets Class

A.3.2.6.2 ADDITIONAL_DATA

ADDITIONAL_DATA	
DESCRIPTION	This field contains additional information.
TYPE	This attribute contains a Free value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.28: Additional_Data Class

A.3.2.6.3 RELATED_ACTIVITY

RELATED_ACTIVITY	
DESCRIPTION	This field contains the trouble TT IDs of the related incidents.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.29: Related_Activity Class

A.3.2.6.4 HISTORY

HISTORY	
DESCRIPTION	This field contains the necessary Actions/events log.
TYPE	This attribute contains a Free value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	YES. This information is required. (*)

Figure A.30: History Class

(*) This field must NOT be empty when the VALID FORMAT attribute of the TT_STATUS field is different from "OPENED" or "OPENED/CLOSED".

A.3.2.7 Localization and Impact

A.3.2.7.1 AFFECTED_COMMUNITY

AFFECTED_COMMUNITY	
DESCRIPTION	This field contains information about the community that was affected by the trouble incident.
TYPE	This attribute contains a Free value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.31: Affected_Community Class

A.3.2.7.2 AFFECTED_SERVICE

AFFECTED_SERVICE	
DESCRIPTION	This field contains the service that was affected by the trouble incident.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.32: Affected_Service Class

A.3.2.7.3 LOCATION

LOCATION	
DESCRIPTION	This field contains the location (Pop site, city, etc.) of the incident/maintenance.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	YES. This information is required.

Figure A.33: Location Class

A.3.2.7.4 NETWORK_NODE

NETWORK_NODE	
DESCRIPTION	This field contains the NOC network node related to the incident.
TYPE	This attribute contains a List value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.34: Network_Node Class

A.3.2.7.5 NETWORK_LINK_CIRCUIT

NETWORK_LINK_CIRCUIT	
DESCRIPTION	This field contains the name of the network line related to the incident.
TYPE	This attribute contains a List value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

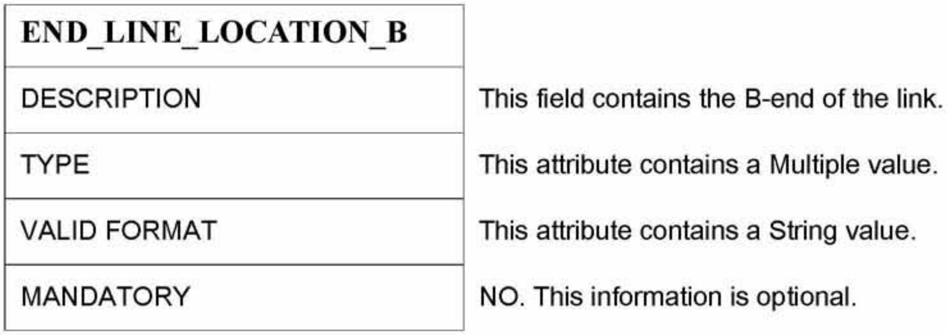
Figure A.35: Network_Link_Circuit Class

A.3.2.7.6 END_LINE_LOCATION_A

END_LINE_LOCATION_A	
DESCRIPTION	This field contains the A-end of the link.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.36: End_Line_Location_A Class

A.3.2.7.7 END_LINE_LOCATION_B

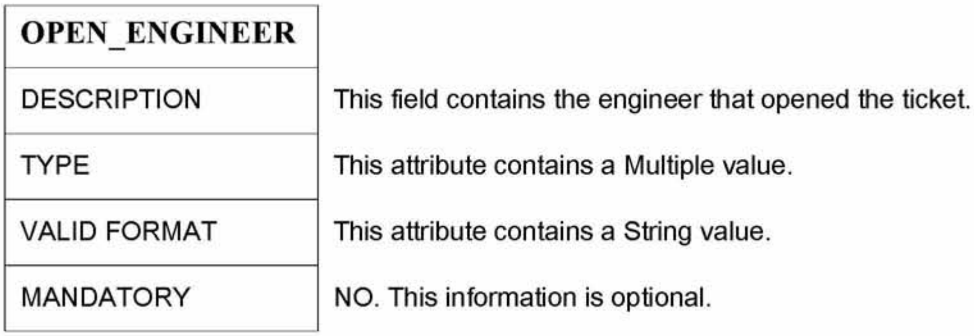


END_LINE_LOCATION_B	
DESCRIPTION	This field contains the B-end of the link.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.37: End_Line_Location_B Class

A.3.2.8 Contact information

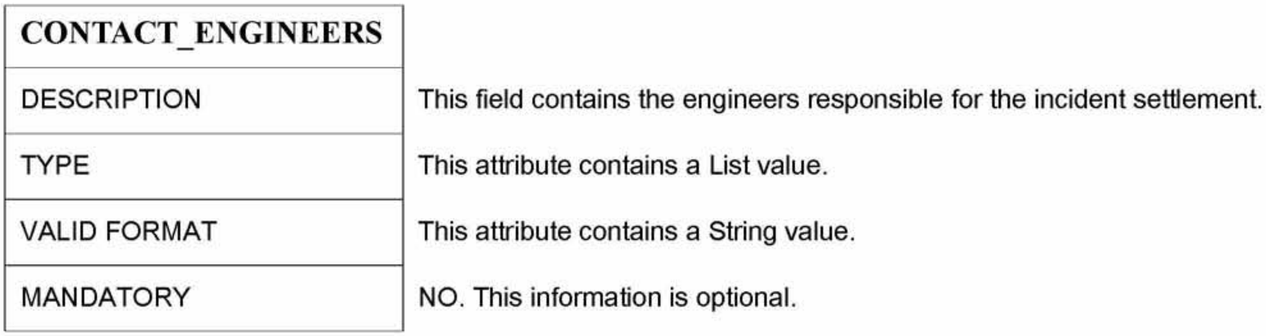
A.3.2.8.1 OPEN_ENGINEER



OPEN_ENGINEER	
DESCRIPTION	This field contains the engineer that opened the ticket.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.38: Open_Engineer Class

A.3.2.8.2 CONTACT_ENGINEERS



CONTACT_ENGINEERS	
DESCRIPTION	This field contains the engineers responsible for the incident settlement.
TYPE	This attribute contains a List value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.39: Contact_Engineers Class

A.3.2.8.3 CLOSE_ENGINEER

CLOSE_ENGINEER	
DESCRIPTION	This field contains the engineer that closed the ticket.
TYPE	This attribute contains a Multiple value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.40: Close_Engineer Class

A.3.2.9 Security

A.3.2.9.1 HASH

HASH	
DESCRIPTION	This field contains the encrypted message hash.
TYPE	This attribute contains a Defined value.
VALID FORMAT	This attribute contains a String value.
MANDATORY	NO. This information is optional.

Figure A.41: Hash Class

A.3.3 NTTDM Representation

The collected and processed TTs received from multiple telecommunications networks are adjusted in a normalized NTTDM.

The representation of this normalized Data Model is shown in Table 3.2. The "DESCRIPTION" attribute is implied.

Table A.2: the Field Name class

FIELD NAME	**TYPE**	**VALID FORMAT**	**MANDATORY**
PARTNER_ID	MULTIPLE	STRING	YES
ORIGINAL_ID	FREE	STRING	YES
TT_ID	DEFINED	STRING	YES
TT_OPEN_DATETIME	MULTIPLE	DATETIME	YES
TT_CLOSE_DATETIME	MULTIPLE	DATETIME	YES
START_DATETIME	MULTIPLE	DATETIME	YES
DETECT_DATETIME	MULTIPLE	DATETIME	NO
REPORT_DATETIME	MULTIPLE	DATETIME	NO
END_DATETIME	MULTIPLE	DATETIME	YES
TT_LAST_UPDATE_TIME	MULTIPLE	DATETIME	YES
TIME_WINDOW_START	MULTIPLE	DATETIME	NO
TIME_WINDOW_END	MULTIPLE	DATETIME	NO
WORK_PLAN_START_DATETIME	MULTIPLE	DATETIME	NO
WORK_PLAN_END_DATETIME	MULTIPLE	DATETIME	NO
TT_TITLE	DEFINED	STRING	YES
TT_SHORT_DESCRIPTION	MULTIPLE	PREDEFINED STRING	YES
TT_LONG_DESCRIPTION	FREE	STRING	NO
TYPE	MULTIPLE	PREDEFINED STRING	YES
TT_TYPE	MULTIPLE	PREDEFINED STRING	YES
TT_IMPACT_ASSESSMENT	MULTIPLE	PREDEFINED STRING	YES
RELATED_EXTERNAL_TICKETS	LIST	STRING	NO
LOCATION	MULTIPLE	STRING	YES
NETWORK_NODE	LIST	STRING	NO
NETWORK_LINK_CIRCUIT	LIST	STRING	NO
END_LINE_LOCATION_A	MULTIPLE	STRING	NO
END_LINE_LOCATION_B	MULTIPLE	STRING	NO
OPEN_ENGINEER	MULTIPLE	STRING	NO
CONTACT_ENGINEERS	LIST	STRING	NO
CLOSE_ENGINEER	MULTIPLE	STRING	NO
TT_PRIORITY	MULTIPLE	PREDEFINED STRING	NO
TT_STATUS	MULTIPLE	PREDEFINED STRING	YES
ADDITIONAL_DATA	FREE	STRING	NO
RELATED_ACTIVITY	MULTIPLE	STRING	NO
HISTORY	FREE	STRING	YES

HASH	DEFINED	STRING	NO
TT_SOURCE	MULTIPLE	STRING	NO
AFFECTED_COMMUNITY	FREE	STRING	NO
AFFECTED_SERVICE	MULTIPLE	STRING	NO

A.4. Internationalization Issues

Internationalization and localization is of specific concern to the NTTDM, since it is only through collaboration, often across language barriers, that certain incidents be resolved. The NTTDM supports this goal by depending on XML constructs, and through explicit design choices in the data model.

The main advantage of the model is that it provides a normalized data type that is implemented fully in the English language and can be used conveniently. It also supports Free formed text that can be written in any language. In the future it will provide translation services for all the free-formed text.

A.5. Examples

A.5.1 Link Failure

In this section an example is provided of network TTs exchanged using the proposed format. This is an actual GRNET ticket normalized according to TTDM. Fields that were not included in the ticket are left blank.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This example describes a link failure that was detected -- >
<NTTDM-Document version="1.00" lang="gr"
  xmlns="urn:ietf:params:xml:ns:NTTDM-1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
 xsi:schemaLocation="urn:ietf:params:xml:schema:NTTDM-1.0">
 <Ticket>
     <Partner_ID> 01 </Partner_ID>
     <Original_ID> 5985 </Original_ID>
     <TT_ID> 01_5985 </TT_ID>
     <TT_Open_Datetime> 2008-12-16T10:01:15+0200
</TT_Open_Datetime>
     <TT_Close_Datetime> 2008-12-16T15:05:00+0200
</TT_Close_Datetime>
     <Start_Datetime> 2008-12-16T09:55:00+0200
</Start_Datetime>
     <End_Datetime> 2008-12-16T15:01:21+0200 </End_Datetime>
     <TT_Last_Update_Time> 2008-12-16T15:00:34+0200
</TT_Last_Update_Time>
     <TT_Title> Forth Link Failure </TT_Title>
     <TT_Short_Description> Core Line Fault
</TT_Short_Description>
     <TT_Long_Description> Forth Link Failure
</TT_Long_Description>
     <Type> Unscheduled </Type>
     <TT_Type> Operational   </TT_Type>
     <TT_Impact_Assessment> No Connectivity
</TT_Impact_Assessment>
     <Location> HERAKLION </Location>
     <Network_Node> FORTH </Network_Node>
     <Network_Link_Circuit> FORTH-2 </Network_Link_Circuit>
     <Open_Engineer> Dimitris Zisiadis </Open_Engineer>
     <Close_Engineer> Dimitris Zisiadis </Close_Engineer>
```

```
      <TT_Priority> High </TT_Priority>
      <TT_Status> Closed </TT_Status>
   </Ticket>
  </NTTDM-Document>
```

A.6. Sample implementation: XML schema

This section provides the XML Schema of the NTTDM.

```
 <?xml version="1.0" encoding="utf-8"?>
 <xs:schema xmlns="urn:ietf:params:xml:ns:nttdm-0.1"
 xmlns:nttdm="urn:ietf:params:xml:ns:nttdm-1.0"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="urn:ietf:params:xml:ns:nttdm-1.0"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">
  <xs:annotation>
     <xs:documentation>
    Trouble Ticket Data Model v-1.0
     </xs:documentation>
  </xs:annotation>
  <!--
================================================================
 == NTTDM-Document class                                          ==
================================================================
  -->
  <xs:element name="NTTDM-Document">
      <xs:complexType>
```

```
           <xs:sequence>
               <xs:element ref="nttdm:Incident" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="version" type="xs:string" fixed="1.00"/>
            <xs:attribute name="lang" type="xs:language" use="required"/>
       </xs:complexType>
  </xs:element>
  <!--
================================================================
 ==  Incident class                                                ==
================================================================
  -->
  <xs:element name="Incident">
      <xs:complexType>
           <xs:sequence>
               <xs:element ref="nttdm:Partner_ID"/>
               <xs:element ref="nttdm:Original_ID"/>
               <xs:element ref="nttdm:TT_ID"/>
               <xs:element ref="nttdm:TT_Open_Datetime"/>
               <xs:element ref="nttdm:TT_Close_Datetime"/>
               <xs:element ref="nttdm:Start_Datetime"/>
               <xs:element ref="nttdm:Detect_Datetime"/>
               <xs:element ref="nttdm:Report_Datetime"/>
               <xs:element ref="nttdm:End_Datetime"/>
               <xs:element ref="nttdm:TT_Last_Update_Time"/>
               <xs:element ref="nttdm:Time_Window_Start"/>
               <xs:element ref="nttdm:Time_Window_End"/>
               <xs:element ref="nttdm:Work_Plan_Start_Datetime"/>
```

```
        <xs:element ref="nttdm:Work_Plan_End_Datetime"/>
        <xs:element ref="nttdm:TT_Title"/>
        <xs:element ref="nttdm:TT_Short_Description"/>
        <xs:element ref="nttdm:TT_Long_Description"/>
        <xs:element ref="nttdm:Type"/>
        <xs:element ref="nttdm:TT_Type"/>
        <xs:element ref="nttdm:TT_Impact_Assessment"/>
        <xs:element ref="nttdm:Related_External_Tickets"/>
        <xs:element ref="nttdm:Location"/>
        <xs:element ref="nttdm:Network_Node"/>
        <xs:element ref="nttdm:Network_Link_Circuit"/>
        <xs:element ref="nttdm:End_Line_Location_A"/>
        <xs:element ref="nttdm:End_Line_Location_B"/>
        <xs:element ref="nttdm:Open_Engineer"/>
        <xs:element ref="nttdm:Contact_Engineers"/>
        <xs:element ref="nttdm:Close_Engineer"/>
        <xs:element ref="nttdm:TT_Priority"/>
        <xs:element ref="nttdm:TT_Status"/>
        <xs:element ref="nttdm:Additional_Data"/>
        <xs:element ref="nttdm:Related_Activity"/>
        <xs:element ref="nttdm:History"/>
        <xs:element ref="nttdm:Hash"/>
        <xs:element ref="nttdm:TT_Source"/>
        <xs:element ref="nttdm:Affected_Community"/>
        <xs:element ref="nttdm:Affected_Service"/>
    </xs:sequence>
    <xs:attribute name="lang" type="xs:language"/>
</xs:complexType>
```

```
  </xs:element>
  <!--
============================================================
==  Partner_ID Class                                          ==
============================================================
  -->
  <xs:element name="Partner_ID" type="nttdm:Partner_IDType"/>
    <xs:complexType name="Partner_IDType">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="Description" type="xs:string"
              fixed="The unique ID of the TT source partner"/>
          <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
          <xs:attribute name="Valid_Format" type="xs:string" use="required"/>
          <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
==  Original_ID class                                         ==
============================================================
    -->
    <xs:element name="Original_ID" type="nttdm:Original_IDType"/>
    <xs:complexType name="Original_IDType">
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="Description" type="xs:string"
```

```
            fixed="The trouble ticket ID that was assigned by the party"/>
        <xs:attribute name="Type" type="nttdm:Free" use="required"/>
        <xs:attribute name="Valid_Format" type="xs:string" use="required"/>
        <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
=  TT_ID class                                                ==
============================================================
  -->
  <xs:element name="TT_ID" type="nttdm:TT_IDType"/>
  <xs:complexType name="TT_IDType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
            fixed="The unique ID of the TT"/>
        <xs:attribute name="Type" type="nttdm:Defined" use="required"/>
        <xs:attribute name="Valid_Format" type="xs:string" use="required"/>
        <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
==  TT_Open_Datetime Class                                    ==
============================================================
  -->
  <xs:element name="TT_Open_Datetime"
        type="nttdm:TT_Open_DatetimeType"/>
  <xs:complexType name="TT_Open_DatetimeType">
     <xs:simpleContent>
        <xs:extension base="xs:string">
           <xs:attribute name="Description" type="xs:string"
              fixed="The datetime that the TT was opened"/>
           <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
           <xs:attribute name="Valid_Format" type="xs:dateTime"
              use="required"/>
           <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
        </xs:extension>
     </xs:simpleContent>
  </xs:complexType>
  <!--
=============================================================
==  TT_Close_Datetime Class                                         ==
=============================================================
  -->
  <xs:element name="TT_Close_Datetime"
       type="nttdm:TT_Close_DatetimeType"/>
  <xs:complexType name="TT_Close_DatetimeType">
     <xs:simpleContent>
        <xs:extension base="xs:string">
           <xs:attribute name="Description" type="xs:string"
              fixed="The datetime that the TT was closed"/>
           <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
        <xs:attribute name="Valid_Format" type="xs:dateTime"
           use="required"/>
        <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
== Start_Datetime Class                                        ==
============================================================
  -->
  <xs:element name="Start_Datetime" type="nttdm:Start_DatetimeType"/>
  <xs:complexType name="Start_DatetimeType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
           fixed="The datetime that the incident/maintenance started"/>
        <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
        <xs:attribute name="Valid_Format" type="xs:dateTime"
           use="required"/>
        <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
== Detect_Datetime Class                                       ==
============================================================
    -->
    <xs:element name="Detect_Datetime" type="nttdm:Detect_DatetimeType"/>
    <xs:complexType name="Detect_DatetimeType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                      fixed="The datetime that the incident was detected"/>
                <xs:attribute name="Type" type="nttdm:Multiple"/>
                <xs:attribute name="Valid_Format" type="xs:dateTime"/>
                <xs:attribute name="Mandatory" type="xs:string" default="No"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
= Report_Datetime Class                                            ==
============================================================
    -->
    <xs:element name="Report_Datetime" type="nttdm:Report_DatetimeType"/>
    <xs:complexType name="Report_DatetimeType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                      fixed="The datetime that the incident was reported"/>
                <xs:attribute name="Type" type="nttdm:Multiple"/>
                <xs:attribute name="Valid_Format" type="xs:dateTime"/>
                <xs:attribute name="Mandatory" type="xs:string" default="No"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
= End_Datetime Class                                           ==
============================================================
    -->
    <xs:element name="End_Datetime" type="nttdm:End_DatetimeType"/>
    <xs:complexType name="End_DatetimeType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                    fixed="The datetime that the incident ended"/>
                <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
                <xs:attribute name="Valid_Format" type="xs:dateTime"
                    use="required"/>
                <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
== TT_Last_Update_Time Class                                   ==
============================================================
    -->
    <xs:element name="TT_Last_Update_Time"
        type="nttdm:TT_Last_Update_TimeType"/>
    <xs:complexType name="TT_Last_Update_TimeType">
```

```
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute name="Description" type="xs:string"
               fixed="The datetime that the TT was updated"/>
            <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
            <xs:attribute name="Valid_Format" type="xs:dateTime"
               use="required"/>
            <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <!--
=========================================================
== Time_Window_Start Class                                      ==
=========================================================
   -->
   <xs:element name="Time_Window_Start"
       type="nttdm:Time_Window_StartType"/>
   <xs:complexType name="Time_Window_StartType">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute name="Description" type="xs:string"
         fixed="The window start time in which planned maintenance may occur"/>
            <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
            <xs:attribute name="Valid_Format" type="xs:dateTime"
               use="optional"/>
            <xs:attribute name="Mandatory" type="xs:string" default="No"/>
         </xs:extension>
```

```
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
= Time_Window_End Class                                                      ==
============================================================
    -->
    <xs:element name="Time_Window_End"
           type="nttdm:Time_Window_EndType"/>
    <xs:complexType name="Time_Window_EndType">
        <xs:simpleContent>
             <xs:extension base="xs:string">
                 <xs:attribute name="Description" type="xs:string"
              fixed="The window end time in which planned maintenance may occur"/>
                 <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
                 <xs:attribute name="Valid_Format" type="xs:dateTime"
                       use="optional"/>
                 <xs:attribute name="Mandatory" type="xs:string" default="No"/>
             </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
 == Work_Plan_Start_Datetime Class                                           ==
============================================================
    -->
    <xs:element name="Work_Plan_Start_Datetime"
           type="nttdm:Work_Plan_Start_DatetimeType"/>
```

```
  <xs:complexType name="Work_Plan_Start_DatetimeType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
            fixed="Work planned start time in case of maintenancer"/>
        <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
        <xs:attribute name="Valid_Format" type="xs:dateTime"
            use="optional"/>
        <xs:attribute name="Mandatory" type="xs:string" default="No"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
==============================================================
 == Work_Plan_End_Datetime Class                                  ==
==============================================================
  -->
  <xs:element name="Work_Plan_End_Datetime"
      type="nttdm:Work_Plan_End_DatetimeType"/>
  <xs:complexType name="Work_Plan_End_DatetimeType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
            fixed="Work planned end time in case of maintenancer"/>
        <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
        <xs:attribute name="Valid_Format" type="xs:dateTime"
            use="optional"/>
        <xs:attribute name="Mandatory" type="xs:string" default="No"/>
```

```
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
==============================================================
== TT_Title Class                                                          ==
==============================================================
    -->
    <xs:element name="TT_Title" type="nttdm:TT_TitleType"/>
    <xs:complexType name="TT_TitleType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                      fixed="The title of the TT"/>
                <xs:attribute name="Type" type="nttdm:Defined" use="required"/>
                <xs:attribute name="Valid_Format" type="xs:string" use="required"/>
                <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
==============================================================
== TT_Short_Description Class                                              ==
==============================================================
    -->
    <xs:element name="TT_Short_Description"
          type="nttdm:TT_Short_DescriptionType"/>
    <xs:complexType name="TT_Short_DescriptionType">
    <xs:simpleContent>
        <xs:extension base="xs:string">
            <xs:attribute name="Description" type="xs:string"
                 fixed="The short description of the TT"/>
            <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
            <xs:attribute name="ValidFormat" use="required">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="Core Line Fault"/>
                        <xs:enumeration value="Access Line Fault"/>
                        <xs:enumeration value="Degraded Service"/>
                        <xs:enumeration value="Router Hardware Fault"/>
                        <xs:enumeration value="Router Software Fault"/>
                        <xs:enumeration value="Routing Problem"/>
                        <xs:enumeration value="Undefined Problem"/>
                        <xs:enumeration value="Network Congestion"/>
                        <xs:enumeration value="Client Upgrade"/>
                        <xs:enumeration value="IPv6"/>
                        <xs:enumeration value="QoS"/>
                        <xs:enumeration value="Other"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
            <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<!--
```

```
============================================================
== TT_Long_Description Class                                ==
============================================================
  -->
  <xs:element name="TT_Long_Description"
      type="nttdm:TT_Long_DescriptionType"/>
  <xs:complexType name="TT_Long_DescriptionType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
            fixed="The detailed description of the
                  problem/incident/maintenance reported in the TT"/>
        <xs:attribute name="Type" type="nttdm:Free" use="required"/>
        <xs:attribute name="Valid_Format" type="xs:string" use="required"/>
        <xs:attribute name="Mandatory" type="xs:string" default="No"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
=  Type Class                                                 ==
============================================================
  -->
  <xs:element name="Type" type="nttdm:TypeType"/>
  <xs:complexType name="TypeType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
```

```
                      fixed="The type of the trouble"/>
                <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
                <xs:attribute name="ValidFormat" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Scheduled"/>
                            <xs:enumeration value="Unscheduled"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
== TT_Type Class                                          ==
============================================================
    -->
    <xs:element name="TT_Type" type="nttdm:TT_TypeType"/>
    <xs:complexType name="TT_TypeType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                      fixed="The type of the TT"/>
                <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
                <xs:attribute name="ValidFormat" use="required">
                    <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="Operational"/>
              <xs:enumeration value="Informational"/>
              <xs:enumeration value="Administrative"/>
              <xs:enumeration value="Test"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
=============================================================
== TT_Impact_Assessment Class                                          ==
=============================================================
  -->
  <xs:element name="TT_Impact_Assessment"
     type="nttdm:TT_Impact_AssessmentType"/>
  <xs:complexType name="TT_Impact_AssessmentType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
           fixed="The impact of the incident/maintenance"/>
        <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
        <xs:attribute name="ValidFormat" use="required">
          <xs:simpleType>
            <xs:restriction base="xs:string">
```

```
              <xs:enumeration value="No Impact"/>
              <xs:enumeration value="Reduced Redundancy"/>
              <xs:enumeration value="Minor Performance Impact"/>
              <xs:enumeration value="Severe Performance Impact"/>
              <xs:enumeration value="No Connectivity"/>
              <xs:enumeration value="On Backup"/>
              <xs:enumeration value="At Risk"/>
              <xs:enumeration value="Unknown"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
=============================================================
 == Related_External_Tickets Class                                  ==
=============================================================
  -->
  <xs:element name="Related_External_Tickets"
     type="nttdm:Related_External_TicketsType"/>
  <xs:complexType name="Related_External_TicketsType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
           fixed="The NOC entity related to the incident"/>
        <xs:attribute name="Type" type="nttdm:List" use="optional"/>
        <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
        <xs:attribute name="Mandatory" type="xs:string" default="No"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
== Location Class                                          ==
============================================================
  -->
  <xs:element name="Location" type="nttdm:LocationType"/>
  <xs:complexType name="LocationType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
          fixed="Location (Pop site, city, etc.) of the incident/maintenance"/>
        <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
        <xs:attribute name="Valid_Format" type="xs:string" use="required"/>
        <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
== Network_Node Class                                      ==
============================================================
  -->
  <xs:element name="Network_Node" type="nttdm:Network_NodeType"/>
```

```
   <xs:complexType name="Network_NodeType">
      <xs:simpleContent>
          <xs:extension base="xs:string">
              <xs:attribute name="Description" type="xs:string"
                   fixed="The network node related to the incident"/>
              <xs:attribute name="Type" type="nttdm:List" use="optional"/>
              <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
              <xs:attribute name="Mandatory" type="xs:string" default="No"/>
          </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <!--
==============================================================
= Network_Link_Circuit Class                                    ==
==============================================================
   -->
   <xs:element name="Network_Link_Circuit"
        type="nttdm:Network_Link_CircuitType"/>
   <xs:complexType name="Network_Link_CircuitType">
      <xs:simpleContent>
          <xs:extension base="xs:string">
              <xs:attribute name="Description" type="xs:string"
                   fixed="Name of the network line related to the incident"/>
              <xs:attribute name="Type" type="nttdm:List" use="optional"/>
              <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
              <xs:attribute name="Mandatory" type="xs:string" default="No"/>
          </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <!--
============================================================
 == End_Line_Location_A Class                                                              ==
============================================================
   -->
   <xs:element name="End_Line_Location_A"
        type="nttdm:End_Line_Location_AType"/>
   <xs:complexType name="End_Line_Location_AType">
       <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                     fixed="A-end of the link"/>
                <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
                <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
                <xs:attribute name="Mandatory" type="xs:string" default="No"/>
            </xs:extension>
       </xs:simpleContent>
   </xs:complexType>
   <!--
============================================================
  ==  End_Line_Location_B Class                                                            ==
============================================================
   -->
   <xs:element name="End_Line_Location_B"
        type="nttdm:End_Line_Location_BType"/>
   <xs:complexType name="End_Line_Location_BType">
       <xs:simpleContent>
```

```
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                      fixed="B-end of the link"/>
                <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
                <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
                <xs:attribute name="Mandatory" type="xs:string" default="No"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
 == Open_Engineer Class                                          ==
============================================================
    -->
    <xs:element name="Open_Engineer" type="nttdm:Open_EngineerType"/>
    <xs:complexType name="Open_EngineerType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                      fixed="The engineer that opened the ticket"/>
                <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
                <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
                <xs:attribute name="Mandatory" type="xs:string" default="No"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
 == Contact_Engineers Class                                           ==
============================================================
  -->
  <xs:element name="Contact_Engineers" type="nttdm:Contact_EngineersType"/>
  <xs:complexType name="Contact_EngineersType">
     <xs:simpleContent>
        <xs:extension base="xs:string">
           <xs:attribute name="Description" type="xs:string"
                 fixed="The engineers responsible for the incident settlement"/>
           <xs:attribute name="Type" type="nttdm:List" use="optional"/>
           <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
           <xs:attribute name="Mandatory" type="xs:string" default="No"/>
        </xs:extension>
     </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
== Close_Engineer Class                                               ==
============================================================
  -->
  <xs:element name="Close_Engineer" type="nttdm:Close_EngineerType"/>
  <xs:complexType name="Close_EngineerType">
     <xs:simpleContent>
        <xs:extension base="xs:string">
           <xs:attribute name="Description" type="xs:string"
              fixed="The engineer that closed the ticket"/>
           <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
           <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
        <xs:attribute name="Mandatory" type="xs:string" default="No"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
=================================================================
 == TT_Priority Class                                              ==
=================================================================
  -->
  <xs:element name="TT_Priority" type="nttdm:TT_PriorityType"/>
  <xs:complexType name="TT_PriorityType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
           fixed="The trouble ticket priority"/>
        <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
        <xs:attribute name="ValidFormat" use="optional">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="Low"/>
              <xs:enumeration value="Medium"/>
              <xs:enumeration value="High"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="Mandatory" type="xs:string" default="No"/>
      </xs:extension>
    </xs:simpleContent>
```

```
   </xs:complexType>
   <!--
==============================================================
 == TT_Status Class                                                                            ==
==============================================================
   -->
   <xs:element name="TT_Status" type="nttdm:TT_StatusType"/>
   <xs:complexType name="TT_StatusType">
       <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                      fixed="The trouble ticket status"/>
                <xs:attribute name="Type" type="nttdm:Multiple" use="required"/>
                <xs:attribute name="ValidFormat" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="Opened"/>
                            <xs:enumeration value="Updated"/>
                            <xs:enumeration value="Solved"/>
                            <xs:enumeration value="Closed"/>
                            <xs:enumeration value="Inactive"/>
                            <xs:enumeration value="Opened/Closed"/>
                            <xs:enumeration value="Cancelled"/>
                            <xs:enumeration value="Superseded"/>
                            <xs:enumeration value="Reopened"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
= Additional_Data Class                                                       ==
============================================================
    -->
    <xs:element name="Additional_Data" type="nttdm:Additional_DataType"/>
    <xs:complexType name="Additional_DataType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                    fixed="Additional information"/>
                <xs:attribute name="Type" type="nttdm:Free" use="optional"/>
                <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
                <xs:attribute name="Mandatory" type="xs:string" default="No"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
 == Related_Activity Class                                                ==
============================================================
    -->
    <xs:element name="Related_Activity" type="nttdm:Related_ActivityType"/>
    <xs:complexType name="Related_ActivityType">
```

```
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                    fixed="The TT IDs of the related incidents"/>
                <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
                <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
                <xs:attribute name="Mandatory" type="xs:string" default="No"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
 == History Class                                                ==
============================================================
    -->
    <xs:element name="History" type="nttdm:HistoryType"/>
    <xs:complexType name="HistoryType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                    fixed="The necessary Actions/events log"/>
                <xs:attribute name="Type" type="nttdm:Free" use="required"/>
                <xs:attribute name="Valid_Format" type="xs:string" use="required"/>
                <xs:attribute name="Mandatory" type="xs:string" default="Yes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
```

```
============================================================
 == Hash Class                                                 ==
============================================================
  -->
  <xs:element name="Hash" type="nttdm:HashType"/>
  <xs:complexType name="HashType">
    <xs:simpleContent>
       <xs:extension base="xs:string">
         <xs:attribute name="Description" type="xs:string"
            fixed="Encrypted message hash"/>
         <xs:attribute name="Type" type="nttdm:Defined" use="optional"/>
         <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
         <xs:attribute name="Mandatory" type="xs:string" default="No"/>
       </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
== TT_Source Class                                             ==
============================================================
  -->
  <xs:element name="TT_Source" type="nttdm:TT_SourceType"/>
  <xs:complexType name="TT_SourceType">
    <xs:simpleContent>
       <xs:extension base="xs:string">
         <xs:attribute name="Description" type="xs:string"
            fixed="The source of the ticket"/>
         <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
```

```
        <xs:attribute name="ValidFormat" use="optional">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="Users"/>
              <xs:enumeration value="Monitoring"/>
              <xs:enumeration value="Other NOC"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="Mandatory" type="xs:string" default="No"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!--
============================================================
 == Affected_Community Class                                          ==
============================================================
  -->
  <xs:element name="Affected_Community"
     type="nttdm:Affected_CommunityType"/>
  <xs:complexType name="Affected_CommunityType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="Description" type="xs:string"
          fixed="The community that was affected by the trouble incident"/>
        <xs:attribute name="Type" type="nttdm:Free" use="optional"/>
        <xs:attribute name="Valid_Format" type="xs:string" use="optional"/>
        <xs:attribute name="Mandatory" type="xs:string" default="No"/>
```

```
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
 == Affected_Service Class                                   ==
============================================================
    -->
    <xs:element name="Affected_Service" type="nttdm:Affected_ServiceType"/>
    <xs:complexType name="Affected_ServiceType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Description" type="xs:string"
                    fixed="The service that was affected by the trouble incident"/>
                <xs:attribute name="Type" type="nttdm:Multiple" use="optional"/>
                <xs:attribute name="ValidFormat" type="xs:string" use="optional"/>
                <xs:attribute name="Mandatory" type="xs:string" default="No"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--
============================================================
 == Data Types                                                  ==
============================================================
    -->
    <xs:simpleType name="Defined">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
```

```
  <xs:simpleType name="Free">
      <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <xs:simpleType name="Multiple">
      <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <xs:simpleType name="List">
      <xs:restriction base="xs:string"/>
  </xs:simpleType>
</xs:schema>
```

A.7. Security Considerations

Some security issues should be kept in mind as network TTs could carry sensitive information (IP addresses, contact details, authentication details, commercial providers involved etc.) about flagship institutions (military, health centre...).

- Integrity: The HASH field is intended to ensure integrity of received tickets.
- Confidentiality: could be ensured by encrypting whole tickets or only some parts. This could allow having meaningful tickets to be disclosed while only sensitive information protected.
- Peer entity authentication: message sender authentication must be provided in order to establish session with data origin authentication regardless of the form in which the TTs are exchanged, being either delivered through email, web forms or through a SOAP service. The latter is considered the better choice, the model itself though does not specify the communications requirements.

A.8. List of acronyms

TT:	Trouble Ticket
NTTDM:	Network Trouble Ticket Data Model
DB:	Data Base
EGEE:	Enabling Grid for E-sciencE
ENOC:	EGEE NOC
NOC:	Network Operation Centre
GOC:	Grid Operation Centre
NREN:	National Research and Educational Networks
QoS:	Quality of service
SLA:	Service level Agreement
UML:	Unified Modeling Language
XML:	Extensible Markup Language

BIBLIOGRAPHY

1. http://www.tcpdump.org
2. http://www.ethereal.com
3. http://www.sniff-em.com
4. Scott R. Fluhrer, Itsik Mantin, Adi Shamir, *"Weaknesses in the Key Scheduling Algorithm of RC4"*, Selected Areas in Cryptography 2001: 1-24
5. Me G., Verdone, D., *"An overview of some techniques to exploit VoIP over WLAN"*, International Conference on Digital Telecommunications, 2006. ICDT '06. 29-31 Aug. 2006
6. Ulrike Meyer, Susanne Wetzel, *A man-in-the-middle attack on UMTS*. In Proceedings of ACM Workshop on Wireless Security (WiSE 2004), ACM, October 2004
7. Ulrike Meyer, Susanne Wetzel, *On the Impact of GSM Encryption and Man-in-the-Middle Attacks on the Security of Interoperating GSM/UMTS Networks*, Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004 15th IEEE International Symposium on September 2004
8. F.C. Piper and M. Walker, *Cryptographic solutions for voice telephony and GSM*, In Proceedings of COMPSEC '98, London, UK, Elsevier. 15, November 1998
9. Suvada Myagmar and Vineet K. Gupta, *3G Security Principals*, http://choices.cs.uiuc.edu/MobilSec/posted_docs/3G_Security_Overview.ppt
10. Grecas, C.F., Maniatis, S.I., Venieris, I.S., *Towards the introduction of the asymmetric cryptography in GSM, GPRS and UMTS networks*, Proceedings, Sixth IEEE Symposium on Computers and Communications 2001, July 2001
11. Ulrike Meyer, Susanne Wetzel, *Wireless monitoring and denial of service: A man-in-the-middle attack on UMTS*, Proceedings of the 2004 ACM workshop on Wireless security, ACM Press, October 2004
12. Badra, M., Hajjeh, I., *Key-Exchange Authentication Using Shared Secrets*, Computer Volume 39, Issue 3, March 2006
13. Amit Parnerkar, Dennis Guster, Jayantha Herath, *Secret key distribution protocol using public key cryptography*, Journal of Computing Sciences in Colleges, Volume 19 Issue 1, Consortium for Computing Sciences in Colleges , October 2003
14. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Fifth Printing, August 2001
15. Niels Ferguson, Bruce Schneier, *Practical Cryptography*, Wiley, 2003
16. Steve Burnett, RSA *Security's Official Guide to Cryptography*, McGraw-Hill/Osborne Media, 2001

17. W. Diffie, and M. E. Hellman, “New directions in cryptography,” IEEE Transactions on Information Theory, IT-22:644–654, 1976

18. D. P. Maher, “Secure communication method and apparatus,” U.S. Patent Number 5,450,493, September 1995. Filed on 29th December 1993

19. International Organization for Standardization, Geneve, Switzerland. ISO/IEC 10118–3: 2003, Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash- functions, 2nd edition, 2003

20. C. Gehrmann, C. J. Mitchell, and K. Nyberg, “Manual authentication for wireless devices,” RSA Cryptobytes, 7(1):29–37, January 2004

21. C. Gehrmann et al. SHAMAN Deliverable: Detailed Technical Spec ification of Mobile Terminal System Security, May 2002. Available at www.isrc.rhul.ac.uk/shaman/docs/d10v1.pdf

22. [22] J-H Hoepman, “The ephemeral pairing problem,” In Proc. Int. Conf. Financial Cryptography, number 3110 in Lecture Notes in Computer Science, pages 212–226. Springer, 2004

23. S. Vaudenay, “Secure communications over insecure channels based on short authenticated strings,” In Advances in Cryptology - CRYPTO 2005, number 3621 in Lecture Notes in Computer Science, pages 309 – 326. Springer Verlag, 2005

24. S. Laur, N. Asokan, and K. Nyberg, “Efficient mutual data authentication based on short authenticated strings,” IACR Cryptology ePrint Archive: Report 2005/424 available at http://eprint.iacr.org/2005/424, November 2005

25. D. Kirovski, M. Sinclair, and D. Wilson, “The Martini Synch: Device Pairing via Joint Quantization,” Information Theory, 2007. ISIT 2007. IEEE International Symposium on , vol., no., pp.466-470, 24-29 June 2007

26. J. M. McCune, A. Perrig, and M. K. Reiter, “Seeing-Is-Believing: using camera phones for human-verifiable authentication,” In Proceedings of the 2005 IEEE Symposium on Security and Privacy (May 08 - 11, 2005). SP. IEEE Computer Society, Washington, DC, 110-124

27. M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun, “Loud and Clear: Human-Verifiable Authentication Based on Audio,” Distributed Computing Systems, 2006. ICDCS 2006. 26th IEEE International Conference on , vol., no., pp. 10-10, 2006

28. American National Standards Institute, ANSI X9.30.1-1997: Public-Key Cryptography for the Financial Services Industry - Part 1: The Digital Signature Algorithm (DSA), American Bankers Association, 1997

29. V.S. Miller, “Use of elliptic curves in cryptography,” Advances in Cryptology - Crypto '85, Springer-Verlag (1986), 417-426

30. N. Koblitz, “Elliptic curve cryptosystems,” Mathematics of Computation 48 (1997), 203-209.

31. B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," Fast Software

Encryption: Second International Workshop, Leuven, Belgium, December 1994, Proceedings, Springer-Verlag, 1994, pp. 191-204

32. American National Standards Institute. ANSI X9.52-1998, Triple Data Encryption Algorithm Modes of Operation. 1998
33. B. Schneier, “Applied cryptography,” John Wiley & Sons, 2nd edition (1996)
34. http://vipsec.inf.uth.gr/ PresentApplet1.html
35. Phil Zimmerman, “ZRTP: Extensions to RTP for Diffie-Hellman Key Agreement for SRTP”, Internet Draft
36. http://zfoneproject.com
37. M. Cagalj, S. Capkun, and J. Hubaux, “*Key agreement in peer-to-peer wireless networks*”, IEEE Special Issues on Cryptography and Security, 2006
38. Phil Zimmerman, Jon Callas, *“ZRTP and ZFone”*, NOMS 2006, 3rd April 2006
39. Vlastimil Klíma, *“Finding MD5 Collisions – a Toy For a Notebook”*, IACR ,2005
40. John Kelsey, and Bruce Schneier, *“Second Preimages on n-bit Hash Functions for Much Less than* 2^n *Work”*, SANE 2006
41. http://blogs.msdn.com/tomarcher/archive/2006/05/10/594204.aspx
42. http://www.java.com
43. Quality of Service (QoS), http://en.wikipedia.org/wiki/Quality of service
44. Service Level Agreement (SLA), http://en.wikipedia.org/wiki/Service Level Agreement
45. GÉANT2, http://www.geant2.net/
46. Keystone, http://www.stonekeep.com/keystone.php/
47. ITSM, http://www.bmc.com/products/products services detail/0,0,19052 19426 52560284,00.html
48. HEAT, http://www.frontrange.com/ProductsSolutions/SubCategory.aspx?id=35&ampccid=6
49. SimpleTicket, http://www.simpleticket.net/
50. OTRS, http://otrs.org/
51. GRNET, http://www.grnet.gr/
52. Service Activity 2 (SA2), http://www.eu-egee.org/
53. ENOC, http://egee-sa2.web.cern.ch/egee-sa2/ENOC.html
54. RFC 1297, http://www.ietf.org/rfc/rfc1297.txt
55. EGEE-SA2-TEC-503527- NREN Interface-v2-7
56. PERTKB Web, http://pace.geant2.net/cgi-bin/twiki/view/PERTKB/WebHome
57. GÉANT2 Performance Enhancement and Response Team (PERT) User Guide and Best Practice Guide, http://www.geant2.net/upload/pdf/GN2-05-176v4.1.pdf
58. Implementing a Multi-Domain Monitoring Service for the European Research Networks, http://tnc2007.terena.org/core/getfile.php?file id=148

59. The OSS Trouble Ticket API (OSS/J API), http://jcp.org/aboutJava/communityprocess/first/jsr091/index.html

60. The Incident Object Description Exchange Format (IODEF), http://tools.ietf.org/html/draft-ietf-inch-implement-01

61. Rumbaugh, J., Jacobson, I., and G. Booch, "The Unified Modeling Language Reference Model," ISBN 020130998X, Addison-Wesley, 1998

62. World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, October 2000, <http://www.w3.org/TR/2000/REC-xml-20001006>.

63. XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001 http://www.w3.org/TR/xmlschema-0/

64. World Wide Web Consortium, "XML XML Schema Part 1:Structures Second Edition", W3C Recommendation, October 2004, <http://www.w3.org/TR/xmlschema-1/>

65. World Wide Web Consortium, "XML Schema Part 2: Datatypes Second Edition", W3C Recommendation, October 2004, <http://www.w3.org/TR/xmlschema-2/>

66. World Wide Web Consortium, "Namespaces in XML", W3C Recommendation, January 1999, <http://www.w3.org/TR/REC-xml-names/>

67. XML, http://en.wikipedia.org/wiki/XML

68. Gebhardt, M., Illies, G. and Schindler, W. (2006) 'A note on the practical value of single hash collisions for special file formats', in J. Dittmann (Ed). *Sicherheit*, volume 77 of LNI, GI pp. 333-344

69. A. Lenstra and B. de Weger, "On the Possibility of Constructing Meaningful Hash Collisions for Public Keys," 10th Australasian Conf. Information Security and Privacy (ACISP 05), LNCS 3574, C. Boyd and J. Manuel González Nieto, eds., Springer-Verlag, 2005, pp. 267–279

70. RFC4270, Attacks on Cryptographic Hashes in Internet Protocols

71. Sven Schge, “Finding Hash Collisions”, Publisher Vdm Verlag, Published 04082008, ISBN 3639066138

72. X. Wang, D. Feng, X. Lai, and H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", August 2004, <http://eprint.iacr.org/2004/199>

73. RFC3174, Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001

74. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu, "Collision Search Attacks on SHA1", February 2005, <http://theory.csail.mit.edu/~yiqun/shanote.pdf>

75. RFC1889, RTP: A Transport Protocol for Real-Time Applications

76. RFC3711, The Secure Real-time Transport Protocol (SRTP)

77. RFC3261, SIP: Session Initiation Protocol

78. D. Zisiadis, S. Kopsidas and L. Tassiulas, "Voice Interactive Personalized Security Protocol: Definition

and Security Analysis", 3rd workshop on Secure Network Protocols - NPSec (in conjunction with the IEEE International Conference on Network Protocols - ICNP 2007), October 16 2007, Beijing, China

79. Dimitris Zisiadis, Spyros Kopsidas, Leandros Tassiulas, “ViDPSec: Visual Device Pairing Security”, International Symposium on Privacy and Security Applications (PSA09), August 31 2009, Vancoover, Canada.

80. Dimitris Zisiadis, Spyros Kopsidas, Leandros Tassiulas, VIPSec defined, Computer NetworksVolume 52, Issue 13, , (1) Research and Trials for Reliable VoIP Applications; (2) Wireless Multimedia Sensor Networks, 17 September 2008, Pages 2518-2528. (http://www.sciencedirect.com/science/article/B6VRG-4SGD4YG-1/2/de7d129b526b86d5bc42ad45f9d102aa)

81. D. Zisiadis, S. Kopsidas, L. Tassiulas, *"An Architecture for Secure VoIP and Collaboration Applications",* Third International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing - SecPerU (in conjunction with the IEEE International Conference on Pervasive Services - ICPS 2007), July 15-20 2007, Istanbul, Turkey

82. D. Zisiadis, S. Kopsidas, M. Tsavli, L. Tassiulas, L. Georgiadis, C. Tziouvaras and F. Karayannis, "Grid Management: Data Model Definition for Trouble Ticket Normalization", The Second International Conference on Networks for Grid Applications (Gridnets 2008), October 8-10 2008, Beijing, China

83. D. Zisiadis et al, Internet draft, "The Network Trouble Ticket Data Model", http://tools.ietf.org/html/draft-dzis-nwg-nttdm