Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Σχεδιασμός και Ανάπτυξη Συστήματος Conditional Access σε Συστήματα IPTV

Μιχαηλίδης Μιχαήλ

Επιβλέπων Καθηγητής

Τάσος Νταγιούκλας

Ιανουάριος 2013

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Σχεδιασμός και Ανάπτυξη Συστήματος Conditional Access σε Συστήματα IPTV

Μιχαηλίδης Μιχαήλ

Επιβλέπων Καθηγητής
Τάσος Νταγιούκλας

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Ιανουάριος 2013

Περίληψη

Το περιβάλλον IPTV παρουσιάζει ένα τελείως διαφορετικό περιβάλλον από μια συμβατική τηλεόραση. Βασική προϋπόθεσή είναι η ύπαρξη μιας persistant σύνδεσης μεταξύ του τελικού χρήστη και του παραγωγού των υπηρεσιών. Η διαδικασία αυτή δημιουργεί πολλές νέες δυνατότητες για αυτό που μέχρι σήμερα ονομάζουμε τηλεόραση. Παράλληλα πρέπει να εξασφαλιστούν και κάποια βασικά θέματα όπως η χρήση του περιεχομένου των υπηρεσιών να παραμένει στα όρια που έχουν οριστεί από τον παραγωγό δηλαδή ασφάλεια από κλοπή και πειρατεία, η προστασία της ιδιωτικότητας κάθε τελικού χρήστη, όπως επίσης το δίκτυο μέσω του οποίου θα γίνεται η μετάδοση των υπηρεσιών θα πρέπει να είναι ασφαλές και να μην μπορεί να δεχθεί κακόβουλες επιθέσεις πχ DoS.

Σκοπός της διατριβής είναι να σχεδιαστεί και να αναπτυχθεί μηχανισμός ασφαλείας για υπηρεσίες IPTV και να συγκριθούν οι διάφοροι τρόποι κρυπτογράφησης για να εξαχθούν συμπεράσματα για τον τρόπο και το είδος της κρυπτογράφησης που απαιτείται.

Η Κλιμακωτή Κωδικοποίηση Video (Scalable video coding) σκοπό έχει την κωδικοποίηση μιας υψηλής ποιότητας ροής video (video bitstream) με τέτοιον τρόπο ώστε να περιέχει ένα ή περισσότερα υποσύνολα ροών (επίπεδα ή layers), τα οποία να μπορούν να αποκωδικοποιηθούν αυτόνομα, με πολυπλοκότητα και ποιότητα ανακατασκευής παρόμοιες με αυτές που θα προέκυπταν εάν το κάθε υποσύνολο κωδικοποιούταν ξεχωριστά από τον αντίστοιχο μη κλιμακωτό κωδικοποιητή (encoder- decoder). Το πρότυπο κωδικοποίησης H.264 θεωρείται σαν το πιο ενδεδειγμένο και πλέον το πιο σύγχρονο για διαδικασίες streaming αφού παρέχει εξαιρετικά αποτελέσματα συμπίεσης και ποιότητας.

Ένα σύστημα CAS (Conditional Access System) μπορεί να εγκατασταθεί αμέσως μετά τον streamer και με την χρήση μεθόδων κρυπτογράφησης ιδιαίτερα διαδεδομένων όπως αυτός των AES, DES, 3DES μπορεί να δώσει τα απαιτούμενα αποτελέσματα ασφαλείας χωρίς να δημιουργεί μεγάλο overhead φορτίο κατά την μετάδοση των υπηρεσιών IPTV. Η δημιουργία ενός εξομοιωτή μιας τέτοιας διαδικασίας ήταν απαραίτητη για την διερεύνηση της βιωσιμότητας του συστήματος CAS που προτείνεται. Οι μετρήσεις στην ολική καθυστέρηση που δημιουργείται έδειξαν ότι ένα τέτοιο σύστημα είναι ικανό να λειτουργήσει αξιόπιστα.

Summary

The IPTV environment presents a completely different environment than a conventional TV. Basic condition is a persistent connection between the client and the service provider. This creates new possibilities for what we call TV, today. Furthermore, some basic issues have to be resolved like the service security that must be according to provider needs, protection from theft, piracy as well as the client's privacy. Finally, it is important the network the IPTV is broadcasting must stay secure and protected from any attacks like DoS.

The purpose of this thesis is to design and develop a security mechanism for IPTV service and to compare the different cryptography ways in order to come to conclusions regarding the development of a CAS system.

The Scalable Video Coding (SVC) is a way coding a high quality bitstream video in a way that contains subsets of video streaming in layers that can be decoded separately in complexity and quality similar to a conventional way. The H.264 coding is considered appropriate and modern tool for video encoding since it offers exceptional results in quality and video compression.

A CAS can be installed after the streamer and with popular cryptography methods like AES, DES, DES3 can give the nesecery security results without high overhead load during the IPTV broadcasting. The creation of a simulator of a CAS is needed to explore the viability of the proposed system. The final results on the total video delay show that this CAS system can work with high reliability.

Ευχαριστίες

Ευχαριστώ θερμά τον κύριο Τάσο Νταγιούκλα για την συνεργασία του και την πολύτιμη βοήθεια του.

Περιεχόμενα

Κεφάλαιο 1

IPTV – Συστήματα CAS

1.1 Τι είναι IPTV – Σύντομη περιγραφή

Η Τηλεόραση μέσω διαδικτύου (IPTV) έχει αναπτυχθεί εμπορικά για σχεδόν 5 χρόνια τώρα. Στο διάστημα αυτό είναι αντιληπτό ότι δημιουργήθηκαν πολλές υπηρεσίες video και αναδείχθηκε το εμπορικό τμήμα της IPTV. Ακόμη και με την ύπαρξη όλων αυτών των ανταγωνιστικών συστημάτων όμως, η IPTV διατηρεί το ρόλο της ως σημαντική δύναμη στην βιομηχανία τηλεόρασης.

Οι νέες εξελίξεις στην IPTV σημαίνει ότι θα διατηρηθεί ο ρόλος της παραδοσιακής τηλεόρασης αλλά προστίθενται νέες εφαρμογές και δυνατότητες μέσω δικτύου. Βασιζόμενη σε υπηρεσίες επικοινωνίας μέσω IP, η εμπειρία της τηλεόρασης έχει επεκταθεί σε πολλές από τις υπηρεσίες που παρέχονται από τις εφαρμογές web. Την ίδια στιγμή οι εξελίξεις στην τεχνολογία IPTV επιτρέπουν την παροχή βίντεο μέσω του διαδικτύου αλλά και την προστασία των εμπορικών δικαιωμάτων του περιεχομένου του.

Υπάρχουν πλέον σημαντικές αλλαγές στη λειτουργία της IPTV σε σχέση με την συμβατική TV. Η εκπομπή της TV δεν αποσκοπεί σε εξυπηρέτηση συσκευών περιορισμένων δυνατοτήτων ("TV" ή "αναπαραγωγής μουσικής"). Αντ' αυτού, έχουν στόχο την πρόσβαση σε πολλαπλών χρήσεων έξυπνες συσκευές. Επίσης, η εκπομπή των δεδομένων δεν γίνεται μέσα από δίκτυα αποκλειστικής πρόσβασης. Αλλά πραγματοποιείται από οποιοδήποτε συνδυασμό των καλωδίων, ψηφιακή συνδρομητική γραμμή (DSL), Fiber to the Home (FTTH) ή κινητή πρόσβαση. Οι δυνατότητες αυτές υποδηλώνουν ότι οι μορφές κωδικοποίησης των δεδομένων και μεταφοράς πρέπει να είναι αναγνώσιμες από διαφορετικούς τύπους συσκευών και διαφορετικών πρόσβασης δικτύων. Για παράδειγμα, ένας θεατής με μια σταθερή γραμμή ADSL μπορεί να έχει εγγραφή στην υπηρεσία IPTV και μπορεί να έχει πρόσβαση στο περιεχόμενό της και για την προβολή σε τηλεόραση, σε ένα προσωπικός υπολογιστής (PC) ή ένα smartphone , που διανέμονται μέσω ενός καλωδιακού δικτύου στο σπίτι, IP σύνδεση δεδομένων ή ασύρματη σύνδεση ευρείας ζώνης.

Η οπτική εμπειρία δεν είναι σύμφωνη με προκαθορισμένα χρονοδιαγράμματα ή ραδιοτηλεοπτικά κανάλια. Τώρα το περιεχόμενο είναι προσαρμοσμένο στον κάθε χρήστη. Εξατομικευμένο περιεχόμενο επιλεγμένο και διαμορφωμένο από τον κάθε χρήστη, ο οποίος μπορεί να ελέγχει τα δικαιώματα πρόσβασης και τον τρόπο παρουσίασης. Αυτές οι δυνατότητες απαιτούν interactive user interface (IUS) και αντίστοιχες εφαρμογές για να παραλάβουν και να διαχειριστούν τις προτιμήσεις του θεατή. Έτσι αντί για μαζικό χειρισμό των προτιμήσεων όλων των χρηστών δίνεται η δυνατότητα των προσωπικών ρυθμίσεων για κάθε χρήστη.

Η IPTV δεν περιορίζεται στο σπίτι του θεατή ή του δικτύου από τον πάροχο των υπηρεσιών του. Στο περιεχόμενο μπορεί να έχει πρόσβαση από οποιαδήποτε θέση στο Internet που διαθέτει. Επιπλέον, όταν έχει πρόσβαση σε ασύρματα δίκτυα, τότε και η τηλεόραση γίνεται ασύρματη. Οι δυνατότητες αυτές απαιτούν προηγμένες δυνατότητες ασφαλείας και πρόσβαση υπό όρους Conditional Access (CA), Digital Rights Management (DRM), καθώς και την προστασία της ιδιοτικότητας του θεατή. Αυτά απαιτούν προηγμένο Quality of Service (QoS) που λειτουργούν πέρα από τα ειδικά δίκτυα πρόσβασης[03].

Η τηλεοπτική εμπειρία δεν περιορίζεται στην επιλογή ενός καναλιού και παθητικά την προβολή περιεχομένου. Η συσκευή με τις εφαρμογές της, widgets, επιτρέπει την αλληλεπίδραση του τηλεθεατή να προσαρμόζει την προβολή του περιεχομένου κατά την βούληση του. Αυτές οι δυνατότητες απαιτούν προηγμένα εργαλεία αλληλεπίδρασης, πέρα από ένα απλό

τηλεχειριστήριο, και ευφυείς συσκευές πύλης, που μπορεί να χειριστεί την ανταλλαγή μηνυμάτων μεταξύ της συσκευής ελέγχου και του δικτύου.

Στο σύνολό τους αυτές οι δυνατότητες επιτρέπουν την κοινωνική τηλεόραση. Οι χρήστες μπορούν να προσωποποιήσουν τις συνδρομές τους σε κοινωνικά δίκτυα, και να ανταλλάξουν απόψεις για τις συνήθειες τους και προτιμήσεις τους. Αυτές μπορούν να αποσταλούν οπουδήποτε και σε άλλα μέλη της ίδιας κοινωνικής ομάδας και σε οποιονδήποτε συνδυασμό συσκευών.

Η διαχείριση περιεχομένου και δημιουργία υπηρεσιών είναι βασικές λειτουργίες του κάθε συστήματος διανομής τηλεόραση. Όσον αφορά την Υπηρεσία Δημιουργίας IPTV προσφέρει πολλά πλεονεκτήματα, συμπεριλαμβανομένων αυτών:

• Με την IPTV, η προσαρμογή του περιεχομένου μπορεί να ελεγχθεί με τον απαιτούμενο κατακερματισμό και μπορεί να προσωποποιηθεί για τους εκάστοτε θεατές και για συγκεκριμένες συσκευές. Αυτά είναι δυνατόν επειδή η IPTV έχει κληρονομήσει τη διαχείριση δεδομένων και τις δυνατότητες χειρισμού από τις εφαρμογές Web.

• Ως υπηρεσία IP, η IPTV μπορεί να παραδοθεί σε οποιαδήποτε συσκευή με δυνατότητα IP που μπορεί να εμφανίσει το περιεχόμενο, όπως στο σπίτι μια smart TV, PC, smartphone.

• Οι εξελίξεις στη διαχείριση ταυτοτήτων (IDM) επιτρέπουν την ίδια την ταυτότητα του χρήστη και τις προτιμήσεις που θα συνδέεται με πολλές συσκευές, ανεξάρτητα από τη συσκευή που χρησιμοποιείται.

• Οι επιλογές διαφήμισης μπορούν να προσαρμοστούν για τον συγκεκριμένο συνδρομητή και να παραδοθούν με το IP stream σε πολλαπλά σημεία.

Σε μια IPTV το middleware διαχειρίζεται τη ροή του περιεχομένου από τον πάροχο στον καταναλωτή. Στην πορεία, το περιεχόμενο μπορεί να έχει την ικανότητα:

1. Να προσαρμόζεται στις δυνατότητες της συσκευής λήψης
2. Να παρουσιάζεται στους θεατές ως μέρος των προσφερόμενων υπηρεσιών
3. Να κωδικοποιείται για να ελέγχεται η πρόσβαση,
4. Να προσαρμόζεται στις προτιμήσεις, τα προνόμια των θεατών ή τις γεωγραφικές τους ιδιαιτερότητες.

Αυτές οι τέσσερις μορφές προσαρμογής εμφανίζονται στο μεγαλύτερο μέρος των υπηρεσιών IPTV. Όπως και με άλλα συστήματα λογισμικού, κάθε στρώμα της στοίβας πρωτοκόλλου για middleware servers επικοινωνεί με το αντίστοιχο επίπεδο του πελάτη σε ένα STB (set top box), PC, smartphone, ή άλλη συσκευή του χρήστη. Επιπλέον, μηνύματα ελέγχου ταξιδεύουν κατά μήκος μιας διαδρομής επιστροφής (back channel) από τον πελάτη προς το διακομιστή στα αντίστοιχα στρώματα του πρωτοκόλλου. Τα μηνύματα ελέγχου εξυπηρετούν ανάγκες του θεατή για περιεχόμενο βίντεο (VoD), εξειδικευμένες υπηρεσίες, ελέγχουν την αναπαραγωγή της ροής βίντεο, ή διαχειρίζονται τις προτιμήσεις του χρήστη[02].

Ο κύριος τρόπος μετάδοσης IPTV είναι της γραμμικής τηλεόρασης. Όλες οι άλλες περιπτώσεις χρήσης είναι παραλλαγές ή επεκτάσεις αυτής. Στον κεντρικό server, μεταφέρεται το περιεχόμενο από τους ραδιοτηλεοπτικούς φορείς συμπεριλαμβανομένου και των απαραιτήτων δεδομένων για το περιεχόμενο που διατίθεται από τα παραδοσιακά κανάλια καθώς επίσης και τα ειδικά κανάλια σύμφωνα με τις απαιτήσεις του τηλεθεατή. Για τα ειδικά κανάλια, οι φορείς IPTV έχουν εμπορικές σχέσεις με τους παρόχους περιεχομένου για τα δικαιώματα για αυτό το περιεχόμενο. Στα κεντρικά, το περιεχόμενο κωδικοποιείται όταν είναι απαραίτητο (για παράδειγμα, από MPEG-2 σε MPEG-4 που χρησιμοποιείται ευρέως σε IPTV). Επειδή η λήψη βίντεο, κωδικοποίηση, κρυπτογράφηση και ομαδοποίηση λαμβάνουν χώρα στα κεντρικά, εκεί βρίσκεται και τα μεγάλα έργα υποδομής βίντεο καθώς και συνδέσεις με VoD servers, η υποδομή διαχείρισης περιεχομένου και η διαχείριση των χρηστών για τη χρέωση, τιμολόγηση και εξυπηρέτηση του πελάτη. Μόλις λάβει το περιεχόμενο ο κεντρικός server κωδικοποιείται, πακετάρετε για παράδοση IP, και είτε ετοιμάζεται για μετάδοση, ή αποθηκεύεται για VoD. Έτσι τα βήματα για την IPTV είναι τα εξής:

1) Προσδιορισμός των συσκευών που χρησιμοποιούνται για πρόσβαση στο δίκτυο, έλεγχος ταυτότητας των συσκευών (Authentication (AuthN)), και επιβεβαίωση πρόσβασης (Authorization (AuthZ)) στο δίκτυο και στην υπηρεσία IPTV.

2) Προσδιορισμός της συνδρομής για την συγκεκριμένη συσκευή και επιβεβαίωση των υπηρεσιών που είναι εξουσιοδοτημένες να έχουν πρόσβαση.

3) Αναγνώριση και πιστοποίηση του κάθε χρήστη (για εξατομικευμένη IPTV), και έλεγχος αδείας.

4) Διανομή κλειδιών κρυπτογράφησης στο διακομιστή κρυπτογράφησης, και τα σχετικά κλειδιά αποκρυπτογράφησης στις εξουσιοδοτημένες συσκευή.

Ο θεατής ελέγχει την πρόσβαση του στο περιεχόμενο της IPTV. Στη βασική λειτουργία, το Set Top Box (STB) της IPTV δέχεται πληροφορίες EPG από την πύλη διαχείρισης περιεχομένου, και οι χρήστες επιλέγουν το περιεχόμενο από το EPG. Επιπλέον, τα STB έχουν την δυνατότητα να συλλέξουν δεδομένα διαμόρφωσης και να εκτελέσει δημοφιλή widgets για την εμφάνιση πληροφοριών σχετικά με τις καιρικές συνθήκες, την κυκλοφορία, διαφημίσεις, ειδικές προσφορές. Αυτά τα widgets δίνουν μια υποτυπώδη μορφή της διαδραστικής τηλεόρασης (ITV)[02].

Όπως και οι προκάτοχοί του στην ψηφιακή καλωδιακή τηλεόραση ή δορυφορική, έτσι και στην IPTV πρέπει να ελέγχεται η πρόσβαση των συσκευών στο δίκτυο. Απαιτείτε δηλαδή ένα σύστημα για την παροχή πρόσβασης υπό όρους, ένας μηχανισμός που εμποδίζει το σύνολο του περιεχομένου που προβάλλεται σε μια μη εξουσιοδοτημένη συσκευή. Στην Ευρώπη, ακολουθεί τα πρότυπα DVB υπό όρους πρόσβασης. Επίσης, έχει καθορίσει το πρότυπο CableCARD. Όλα τα συστήματα πρόσβασης υπό όρους βασίζονται στην ίδια αρχή: το περιεχόμενο είναι κωδικοποιημένα με ένα τυχαίο δυναμικό κλειδί το οποίο είναι κρυπτογραφημένο και μεταδίδεται σε κάθε STB ώστε το περιεχόμενο να αποκωδικοποιηθεί. Η αποκρυπτογράφηση του κλειδιού στο STB γίνεται με τη χρήση μιας συσκευής αποκωδικοποίησης. Το κλειδί συχνά αναφέρεται ως κλειδί ελέγχου και στέλνεται μέσω ενός μηνύματος Ελέγχου δικαίωμα (ECM) και διαχειρίζεται μέσω ενός μηνύματος Διαχείρισης δασμών (EMM) του οποίου ο ρόλος είναι να επιτρέψει την αποκρυπτογράφηση βάση της άδειας χρήσης του τελικού χρήστη.

Το Digital Rights Management σχετίζεται με την πρόσβαση υπό όρους CA αλλά είναι διαφορετικό. Κύριος στόχος του είναι να αποτρέψει την παράνομη χρήση του ψηφιακού περιεχομένου από τους τελικούς χρήστες, όπως και για την προστασία κατά της αντιγραφής. Το DRM είναι πλέον εφαρμογή για κάθε ψηφιακό βίντεο και κάθε αναπαραγωγή μουσικής, καθώς και για eBook. Μια σειρά από γνωστά συστήματα DRM έχουν αναπτυχθεί αλλά τελικά το DRM έχει αποδειχθεί ότι είναι ένα εμπόδιο στη διανομή του περιεχομένου. Οι προσπάθειες όπως από το Coral Consortium προσπαθούν να παρέχουν λειτουργικότητα μεταξύ των συστημάτων ώστε πιο διαφανή κατανομή σε εξουσιοδοτημένες συσκευές[08].

Τόσο τα συστήματα CA όσο και τα DRM βοηθούν στην επιβολή κανόνων θέασης του περιεχομένου και σιγά σιγά γίνονται ένα. Όλο και περισσότερο στην πλευρά του STB του συστήματος CA υλοποιείται μέσω αποκωδικοποιητή λογισμικού μαζί με ένα ειδικό chip ασφαλείας. Σύστημα Πρόσβασης DCAS (Downloadable Conditional Access System) είναι το επόμενο βήμα ώστε να επιτρέπει να εγκαταστήσετε ένα νέο CA σε περιόδους παραβιάσεις της

ασφάλειας και θα επιτρέψει τη φορητότητα των STBs από τον ένα φορέα στον άλλο. Το ίδιο το λογισμικό μπορεί να σταλεί ως μήνυμα DRM ενσωματωμένο στο βίντεο, το οποίο με τη σειρά του επιτρέπει στους παρόχους περιεχομένου να χρησιμοποιούν ένα διαφορετικό σύστημα CA[07].

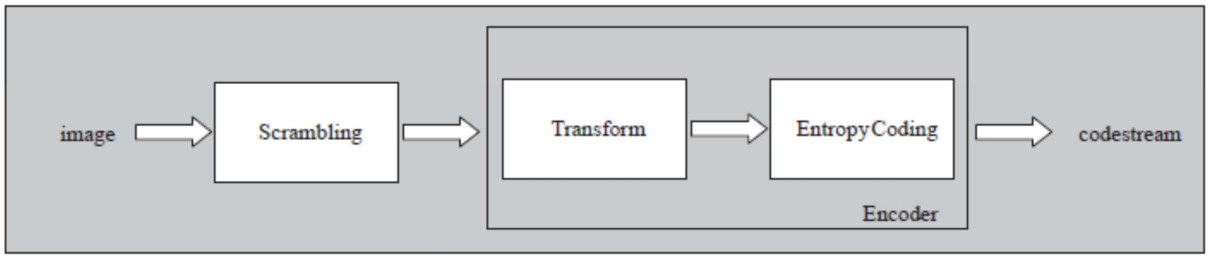
Αυτή η δυνατότητα είναι ιδιαίτερα ενδιαφέρουσα αφού η IPTV φέρει DRM μηνύματα IP από άκρο σε άκρο. Στο συνηθισμένο δίκτυο διανομής IPTV το STB θα είναι διαφορετικό από πάροχο σε πάροχο ως εκ τούτου, ο διαχωρισμός του υλικού της άδειας (που συνδέεται με το chip ασφαλείας) και το περιεχόμενο μέσω του DRM παρέχει δυνατότητες για ευέλικτες λύσεις. Ειδικότερα, όπως η IPTV εξελίσσεται, τα μηνύματα DRM μπορούν να συνδεθούν με νέους μηχανισμούς ασφαλείας που συνδέονται με IDM σε δίκτυα κινητής τηλεφωνίας για παράδειγμα [01].

1.2 Scrambling

Για την μετάδοση video σήμερα συνήθως χρησιμοποιούνται μέθοδοι scrambling. Αυτό γιατί η εγκατάσταση ενός συστήματος κρυπτογράφησης – ασφαλείας κοστίζει πολύ περισσότερο από τα δεδομένα που διακινούνται. Επίσης έχουν μεγάλο bit rate και χαμηλή διαφημιστική αξία. Αλλά και οι τρόποι κρυπτογράφησης δεν είναι πάντα εύκολο να προσαρμοστούν σε ένα τέτοιο σύστημα.

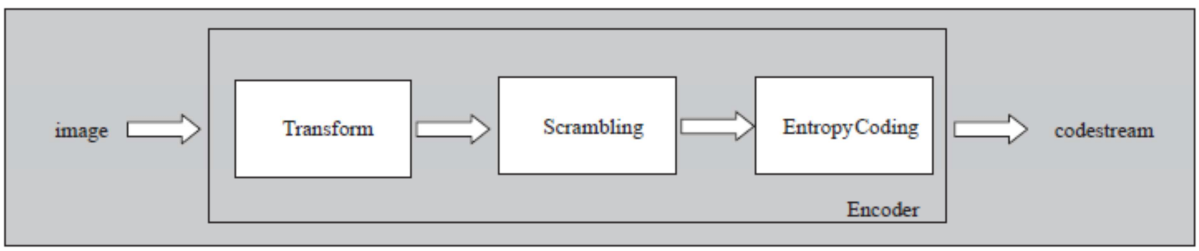
Η συνήθης μέθοδος ασφαλείας είναι η χρήση αλγορίθμων scrambling. Αυτοί μπορούν να κατηγοριοποιηθούν σε:

1) Ψευδό – θόρυβο στα δεδομένα όπου οι authorized χρήστες ξέρουν σε πιο σημείο βρίσκονται και μπορούν να το αφαιρέσουν
2) Διαδικασία αλλαγής των frames (flip, inverse)
3) Κωδικοποίηση βασικών παραμέτρων των δεδομένων



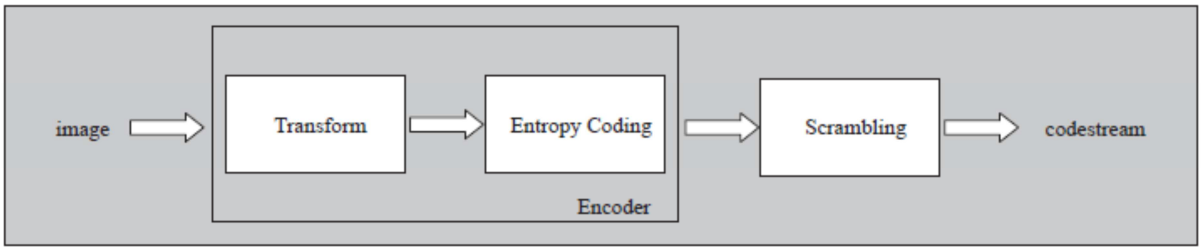
Εικόνα 1-1: Image domain scrambling

Έτσι μπορεί να γίνει αναστροφή κάποιων bits μέσα στον encoder, όπως φαίνεται στην εικόνα 1-1. Αυτό είναι ένας εύκολος τρόπος scrambling που δεν χρειάζεται πολύπλοκες διαδικασίες , αλλά δεν περιέχει υψηλό επίπεδο ασφαλείας και το επίπεδο συμπίεσης είναι σαφώς μικρότερο.



Εικόνα 1-2: Transform domain scrambling

Σε ένα δεύτερο επίπεδο μπορεί να γίνει scrambling μέσα στον encoder όπως φαίνεται στην εικόνα 1-2. Αυτό σημαίνει ότι το scrambling είναι ελεγχόμενο και δεν επηρεάζει το streaming, όμως υπάρχει επιβάρυνση στον encoder.



Εικόνα 1-3: codestream domain scrambling

Σε τρίτο επίπεδο μπορεί να γίνει το scrambling μετά τον encoder αυτό σημαίνει το ίδιο εύκολη διαδικασία, εικόνα 1-3, με το image scrambling αλλά μπορεί να δημιουργήσει πρόβλημα στον decoder μιας και η διαδικασία εκεί γίνεται πιο πολύπλοκη [14].

1.3 CAS

Ένα σύστημα ασφαλείας σε σύστημα IPTV πρέπει να παρέχει:

1. Αξιοπιστία: Κρυπτογραφία για να προστατευθεί το αυθεντικό περιεχόμενο
2. Επιβεβαίωση ακεραιότητας τόσο σε επίπεδο δεδομένων όσο και σε περιεχόμενο
3. Πρόσβαση υπό όρους (conditional access)
4. Διαδικασία authentication και authorization

Για τον περιορισμό της πολυπλοκότητας ενός συστήματος Conditional Access αλλά και την αντίστοιχη αύξηση της απόδοσης του συστήματος μπορούν να χρησιμοποιηθούν συστήματα που βασίζονται σε downloadable CAS system αλλά και σε αφιερωμένους (dedicated) Cas servers.

1.3.1 Downloadable CAS system.

Το πρωτόκολλο authentication protocol που χρησιμοποιείται σε τέτοια συστήματα απαιτεί την ακόλουθη διαδικασία.

1. IPTV φορτώνει με το κατάλληλο software (συνήθως το CAS client software και CAS S/W) με ασφαλές τρόπο ώστε να προστατεύονται τόσα τα αντίστοιχα κλειδιά όσο και άλλες σημαντικές πληροφορίες λειτουργίας.

2. Ο χρήστης λαμβάνει ένα Token. Όταν γίνεται αναφορά σε ένα CA token το σύστημα ελέγχου του χρήστη δημιουργεί ένα token το οποίο συνδυάζεται με τις πληροφορίες του χρήστη έτσι ώστε μόνο εξουσιοδοτημένοι χρήστες μπορούν να το χρησιμοποιήσουν.

3. Με στόχο το set top box (STB) να δεχθεί σωστά το CAS S/W το STB πρέπει να αποδείξει ότι είναι αυθεντικό. Τότε λαμβάνει το CAS S/W κωδικοποιημένο και το public key[04].

4. Το STB αποθηκεύει το CAS S/W και λειτουργεί βασιζόμενο στο SVM. Στη συνέχεια το σύστημα θεωρείται ότι λειτουργεί με ασφάλεια. Η διαδικασία δηλαδή λειτουργίας του αποτελείται από 5 βήματα.

α. CA token και η χρήση του από το συγκεκριμένο STB

β. authentication του STB

γ. authentication του χρήστη

δ. επιβεβαίωση του CAS S/W – ασφαλή μεταφορά του στο STB

ε. εκτέλεση του λογισμικού

Σημειώνεται ότι κάθε CAS S/W διανέμεται σε κάθε STB με δυο κλειδιά που αποκωδικοποιούν και κωδικοποιούν δεδομένα που μεταφέρονται από και προς τον CAS server. Τα κλειδιά μπορεί να είναι συμμετρικά μπορεί και να μην είναι ώστε η επικοινωνία τους να δίνει και ένα επίπεδο ασφάλειας. Επίσης το κανάλι επικοινωνίας μεταξύ του CAS και του streaming server μπορεί να θεωρηθεί ασφαλές γιατί χρησιμοποιούν το ίδιο ζευγάρι από κλειδιά[09].

1.3.2 Dedicated CAS system

Ένα STB είναι αφιερωμένο σε ένα συγκεκριμένο CAS Server. Σε αυτή την περίπτωση το λογισμικό είναι εγκαταστημένο στο STB από την αρχή. Η εταιρία που παρέχει τόσο τον CAS server όσο και το STB διανέμει και τα αντίστοιχα κλειδιά κωδικοποίησης και αποκωδικοποίησης με μια κάρτα ή ενσωματωμένα σε κάποια μνήμη του STB. Σε αντίθεση με τις ψηφιακές εκπομπές IPTV χρησιμοποιεί μια IP και επομένως το STB πρέπει να έχει μια αντίστοιχη MAC address. Αυτή η MAC address είναι μοναδική, και αυτός είναι ένας τρόπος ελέγχου της αυθεντικότητας της συσκευής. Μετά τον έλεγχο μπορεί η συσκευή να επικοινωνεί με ασφάλεια με τον αφιερωμένο CAS server χρησιμοποιώντας τα κλειδιά που έχουν ήδη δοθεί.

Η IPTV αποτελείται από 3 σημαντικά μέρη τον streaming server, τον CAS Server και τον χρήστη CAS S/W Η λειτουργία τους είναι η ακόλουθη:

– Ο πάροχος του περιεχομένου (streaming server) : διαχειρίζεται το περιεχόμενο προς εκπομπή και το εκπέμπει μέσω του δικτύου. Το περιεχόμενο είναι κωδικοποιημένο με ένα κλειδί που δημιουργείται ξεχωριστά για κάθε εκπομπή.

– CAS server: Ο CAS server έχει το ρόλο να κάνει authentication κάθε εμπλεκόμενου στην διαδικασία τόσο τον πάροχο όσο και τον χρήστη. Πως και να δίνει κλειδιά αποκρυπτογράφησης και κρυπτογράφησης τόσο στους χρήστες όσο και τον πάροχο.

– Χρήστης: Με το κλειδί από τον CAS server μπορεί να αποκωδικοποιεί το περιεχόμενο για τον θεατή.

Το πρωτόκολλο επικοινωνίας μπορεί να ορισθεί με τα ακόλουθα βήματα:

Βήμα1: Ο πάροχος δέχεται μήνυμα (seed) να δημιουργήσει το πρώτο OTP από τον CAS Server.

Βήμα 2: Ο πάροχος δημιουργεί το πρώτο OTP.

Βήμα 3: Ο CAS Server στέλνει το ίδιο seed στον χρήστη που θέλει να δει το streaming.

Βήμα 4: Ο πάροχος κωδικοποιεί το STL με το OTPseed

Βήμα 5: Ο χρήστης δημιουργεί και αυτός το αντίστοιχο OTPseed και αποκωδικοποιεί το STL και το αποθηκεύει.

Βήμα 6: Ο χρήστης και ο πάροχος με την χρήση του OTPseed δημιουργούν το OTP.

Βήμα 7: Ο πάροχος κωδικοποιεί το streaming με το OTP0 και το στέλνει στον χρήστη. Ο χρήστης αποκωδικοποιεί το streaming με το αντίστοιχο OTP0.

Βήμα 8: Η διαδικασία στο βήμα 7 επαναλαμβάνεται όσο ισχύει το STL.

Βήμα 9: Όταν τελειώσει ο χρόνος του STL ο πάροχος δημιουργεί νέο OTP το στέλνει στον CAS Server και αυτός με την σειρά του το στέλνει στο αντίστοιχο χρήστη όπως έγινε στο βήμα 3. Τα διάφορα OTP που δημιουργούνται αποθηκεύονται ώστε να μπορούν να επανασταλλούν όταν απαιτείται πχ στην απώλεια δεδομένων.

Βήμα 10: Ο χρήστης δημιουργεί το αντίστοιχο OTP. Τα βήματα 7-10 επαναλαμβάνονται μέχρι το τέλος του streaming[09].

1.3.3 Dynamic CAS Server

Ένα τέτοιο σύστημα μπορούμε να πούμε ότι αποτελείται από τον demultiplexer, που μπορεί να ελέγξει τον CA downloader και το διαχειριστή λογισμικού του CA. Ο demultiplexer μεταφέρει τα

μηνύματα EMM και ECM στο διαχειριστή λογισμικού. Επίσης αναλύει τις ιδιαίτερες πληροφορίες του προγράμματος από το stream και μεταδίδει τις πληροφορίες στον διαχειριστή λογισμικού. Ο downloader κατεβάζει τα δεδομένα για το λογισμικό του CA μέσω ασφαλούς σύνδεσης με πρωτόκολλο IP. Γίνεται επιβεβαίωση του λογισμικού με την ψηφιακή του υπογραφή. Ο διαχειριστής λογισμικού τρέχει το αντίστοιχο λογισμικό με βάσει τις πληροφορίες που έχει για το τρέχον stream. Αν αλλάξει η κατάσταση του stream (αλλαγή προγράμματος, ανανέωση λογισμικού) ο διαχειριστής λογισμικού τρέχει το αντίστοιχο λογισμικό, Αν ο διαχειριστής δεν βρίσκει το λογισμικό που απαιτείται τότε ζητεί από τον downloader να φέρει το αντίστοιχο λογισμικό [06].

Κεφάλαιο 2

Πρότυπο Η. 264

2.1 Κωδικοποίηση Η.264

κατά την εγγραφή βίντεο περίπου 9000 frames βίντεο μεταφέρονται από την κάμερα και στέλνονται στον εσωτερικό κωδικοποιητή Η.264 για συμπίεση. Κάθε frame βίντεο είναι συμπιέζεται σε έναν από τους δύο τρόπους: σαν ένα I-frame ή ένα p-frame.

Ένας I-frame είναι ένα frame κωδικοποιημένο που έχει, χωρίς αναφορά σε οποιοδήποτε άλλο frame του βίντεο. Ένα video frame ξεκινά πάντα με ένα I-frame και περιέχει τακτικά I-frame σε όλο το stream. Αυτά τα frame λέγονται και intra frames ή key frames ή σημεία πρόσβασης και είναι ζωτικής σημασίας για την τυχαία προσπέλαση του video Η.264, όπως με rewind κατά τη διάρκεια της αναπαραγωγής. Τα i-frames είναι στα τακτά διαστήματα μέσα στο video σε

τμήματα που λέγονται i-frame interval. Το μειονέκτημα των I-frames είναι ότι μπορεί να συμπιεσθεί λιγότερο από το P-frame.

Τα P-frame είναι αντισταθμιζόμενα frames κίνησης δηλαδή ο κωδικοποιητής κάνει σύγκριση του παρόντος και του προηγούμενου frame για να διαπιστώσει ποιές πληροφορίες δεν έχουν αλλάξει π.χ. Ένα στατικό φόντο δεν είναι ανάγκη να επαναλαμβάνεται η μεταφορά του. Σε αντίθεση με βασικούς codecs που ελέγχουν μόνο τη διαφορά ο H.264 δεν εξετάζει μόνο την διαφορά αλλά και την κίνηση συμβαίνει από frame σε frame. Έτσι η κίνηση αντισταθμίζεται και η ποιότητα είναι καλύτερη ενώ το μέγεθος της συμπίεσης είναι ψηλό.

2.2 Λειτουργία

Η συμπίεση του Video είναι ουσιαστικά μια διαδικασία, για να μειωθεί και να αφαιρεθεί σημαντικός αριθμός περιττών πληροφοριών που περιέχονται σε ένα video, ώστε αυτό να πάρει μορφή που θα διευκολύνει την αποθήκευση και την μεταφορά του. Η διαδικασία απαιτεί την εφαρμογή ενός αλγορίθμου στο αρχικό video για να δημιουργηθεί ένα συμπιεσμένο αρχείο. Για την αναπαραγωγή του εφαρμόζεται η διαδικασία συμπίεσης αντίστροφα. Ο χρόνος για την συμπίεση του αποστολή και αποσυμπίεση του ονομάζεται καθυστέρηση. Όσο πιο προηγμένη είναι η μέθοδος συμπίεσης τόσο μεγαλύτερη η καθυστέρηση για συγκεκριμένη υπολογιστική δύναμη. Ένα ζεύγος αλγορίθμων που λειτουργούν μαζί ονομάζεται video codec. Διαφορετικοί codecs είναι προφανές ότι είναι και ασύμβατοι μεταξύ τους. Συνήθως όμως αυτοί προσφέρονται σε πακέτο και μπορούν να αποσυμπιέσουν διαφορετικά ήδη συμπίεσης.

Αποτελέσματα από encoders που χρησιμοποιούν τον ίδιο αλγόριθμο συμπίεσης μπορεί επίσης να διαφέρει γιατί ο σχεδιαστής του encoder επέλεξε διαφορετικά εργαλεία που ορίζονται από το πρωτόκολλο του αλγορίθμου. Παρόλα αυτά το decode μπορεί να γίνει από κάθε decoder που έχει τον βασικό αλγόριθμο που χρησιμοποιήθηκε. Αυτό είναι πού χρήσιμο γιατί έτσι μπορούν να γίνουν πολλές παραλλαγές ανάλογα με του στόχους της συμπίεσης καλύτερη ποιότητα, χαμηλότερο κόστος, βελτίωση μεγέθους αρχείων κλπ.

Βασικός στόχος κατά την δημιουργία του H.264 ήταν η δημιουργία μιας απλής και καθαρής λύσης περιορίζοντας τις επιλογές και τις δυνατότητες στο ελάχιστο. Σαν βασικό στοιχείο του H.264 είναι να υποστηρίζει δυνατότητες και επίπεδα απόδοσης ώστε να λειτουργούν με τα πιο δημοφιλή προϊόντα στην αγορά. Το H.264 έχει 7 profiles που το κάθε ένα στοχεύει σε

διαφορετική οικογένεια εφαρμογών. Κάθε profile ορίζει τις δυνατότητες έχει ο encoder και μειώνει την πολυπλοκότητα λειτουργίας του decoder.

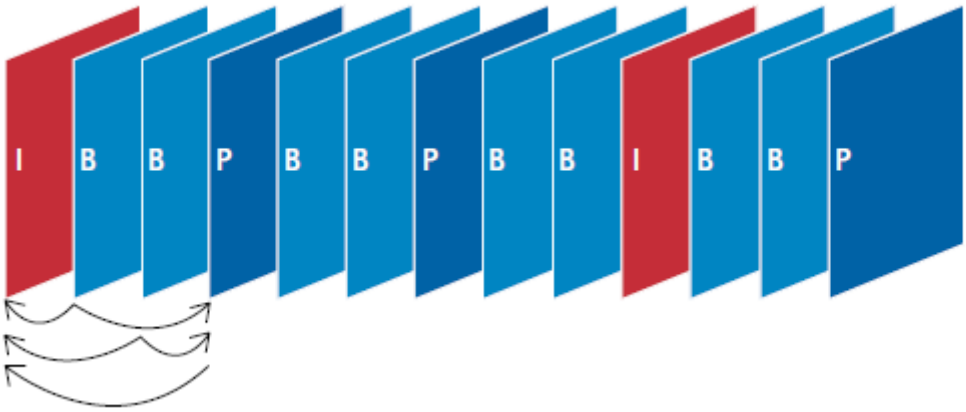
Οι web-κάμερες και οι encoders χρησιμοποιούν το βασικό profile το οποίο προορίζεται κυρίως για εφαρμογές για συστήματα με χαμηλούς πόρους χρήσης. Το βασικό profile είναι ιδανικό για encoding πραγματικού χρόνου για διακίνηση video στο δίκτυο. Το profile επίσης ενεργοποιεί την χαμηλή καθυστέρηση η οποία είναι απαραίτητη για δυνατότητες πραγματικού χρόνου όπως η χρήση cameras zoom/pan/tilt. Επίσης το H.264 έχει 11 επίπεδα για έλεγχο της απόδοσης αλλάζοντας την ποιότητα δηλαδή την ανάλυση του video ανάλογα με την χρήση.

2.2.1 frames

Ανάλογα με το profile υπάρχουν διαφορετικά ήδη frames όπως τα I frames, P frames και B frames. Τα I-frames είναι frames τα οποία μπορούν να αποκωδικοποιηθούν χωρίς αναφορές σε άλλα frames. Το πρώτο frame σε μια σειρά από frames είναι πάντα I-frame. I-frames είναι τα σημεία αναφοράς για νέους χρήστες που τώρα συγχρονίζονται με το video streaming πχ σε περίπτωση σφάλματος δικτύου και επανασύνδεση. Τα I-frames μπορούν να χρησιμοποιηθούν για υπάρχει η δυνατότητα από τον χρήστη για fast forward, rewind. Ο encoder αυτόματο εισάγει I-frames κατά την συμπίεση σε κανονικά χρονικά διαστήματα ανάλογα με τις απαιτήσεις. Το μειονέκτημα είναι ότι αυξάνουν σημαντικά το μέγεθος του αρχείου.

Τα P-frames έχουν αναφορές σε τμήματα από προηγούμενα I ή P-frames για να γίνει decode το frame. Συνήθως χρειάζονται λιγότερα bits από τα I frames αλλά είναι πολύ ευαίσθητα σε λάθη κατά την αποστολή τους εξαιτίας της πολυπλοκότητας τους και τις αναφορές τους σε προηγούμενα P και I-frames.

Ένα B-frame είναι ένα frame με αναφορές τόσο σε προηγούμενα frame όσο και σε επόμενα frames. Μια τυπική ακολουθία από frames μπορεί να είναι I B B P B B P B B I B B P. Όταν ξεκινήσει η διαδικασία decoding πρέπει αυτή πάντα να αρχίζει με I frame. P-frames και B-frames γίνονται decoded μαζί με το frame αναφοράς. Στο βασικό profile που είναι ιδανικό για μεταφορά video streaming μέσω δικτύου αφού μειώνεται η καθυστέρηση αισθητά με την μη χρήση B frames.



Εικόνα 2.1: Τυπική ακολουθία από I- B- και P- frames.

Το H.264 παρουσιάζει ένα τεράστιο βήμα στην συμπίεση video. Προσφέρει τεχνικές που δίνουν καλύτερη συμπίεση λόγο της καλύτερης πρόβλεψης όσο και μείωσης των λαθών. Παρέχει νέες δυνατότητες για την δημιουργία νέου video encoder που μπορούν να δώσουν video ψηλότερης ανάλυσης διατηρώντας το ίδιο bit rate. Η ευελιξία του δίνει τη δυνατότητα να έχει εφαρμογή σε διαφορετικά ήδη που απαιτούν video από DVD υψηλής ανάλυσης μέχρι και online video (YouTube), χρήση σε ασύρματα δίκτυα (τηλέφωνα). Με την υποστήριξη από πολλές βιομηχανίες και εφαρμογές τόσο επαγγελματικές όσο και γενικής χρήσης το H.264 αναμένεται να αντικαταστήσει τα παλαιότερα πρωτόκολλα συμπίεσης σε όλα τα επίπεδα .

Με την εξέλιξη της τεχνολογίας και του δικτύου είναι πλέον απαραίτητο οι πάροχοι video να δίνουν το video σε μορφή και σε ανάλυση συμβατές με όλες τις «έξυπνες συσκευές». Παλιότερα υπήρχαν οι επιλογές ανάλογα με την ταχύτητα που υποστήριζε το δίκτυο του κάθε χρήστη, να του αποσταλεί και αντίστοιχης ποιότητας video (ξεχωριστό αρχείο). Αυτό σημαίνει αυξημένος αποθηκευτικός χώρος και διαδικασία με καθυστέρηση. Για το λόγο αυτό οι σύγχρονες εταιρίες διανομής video προσπάθησαν να βρουν τρόπους να περιορίσουν αυτό το πρόβλημα. Η λύση δόθηκε με το να περιέχονται όλες οι αναλύσεις ενός video στο ίδιο αρχείο. Αυτό όχι μόνο έκανε την διαχείριση των αρχείων ευκολότερη αλλά έλυσε και πολλά προβλήματα προβολής στον τελικό χρήστη αφού τώρα υπήρχε η δυνατότητα από τον server μόλις διαπιστώσει απώλεια πακέτων κατά την μεταφορά να μεταβεί σε αποστολή χαμηλότερης ποιότητας video.

Τρεις είναι οι βασικές μέθοδοι streaming:

- Adaptive streaming

- Dynamic streaming

- Smooth streaming

Όλες αυτές οι μέθοδοι θεωρούνται ικανοποιητικές για κάθε μορφή αποστολής video.

H.264 AVC συμπιέζει το video σε layers. Ξεκινώντας από το βασικό layer που περιέχει το επίπεδο της ανάλυσης του video (resolution), τα frames per sec , και το επίπεδο των λεπτομερειών (higher detail). Τα επιπλέον layer μπορούν να βελτιώσουν αισθητά την ποιότητα του video χρησιμοποιώντας αυτούς τους παράγοντες. Για παράδειγμα το βασικό layer μπορεί να έχει συμπιεσθεί με 15 frames per sec και σε ανάλυση 320χ240 με data rate 300kbps. Τα επιπλέον layers μπορούν να φέρουν το video το 720p και 3mbps που ταιριάζει σε set top box με ικανό αριθμό I frames για καλύτερη χρήση του video από τον πελάτη. Σε σύγκριση με άλλες μεθόδους H.264 SVC είναι πολύ αποδοτικός encoder αφού μπορεί να είναι μόνο 20% μεγαλύτερο από το αρχείο με την καλύτερη δυνατή ποιότητα. Με άλλα λόγια αν η ανώτερη ποιότητα έχει data rate 3mbps τότε το SVC video θα έχει 3.6mbps. Επίσης με τους νέους encoders μπορεί αυτό να γίνει της ώρα της αποστολής που σημαίνει ότι δεν χρειάζεται αυτό να έχει προετοιμασία για την αποστολή του[10] .

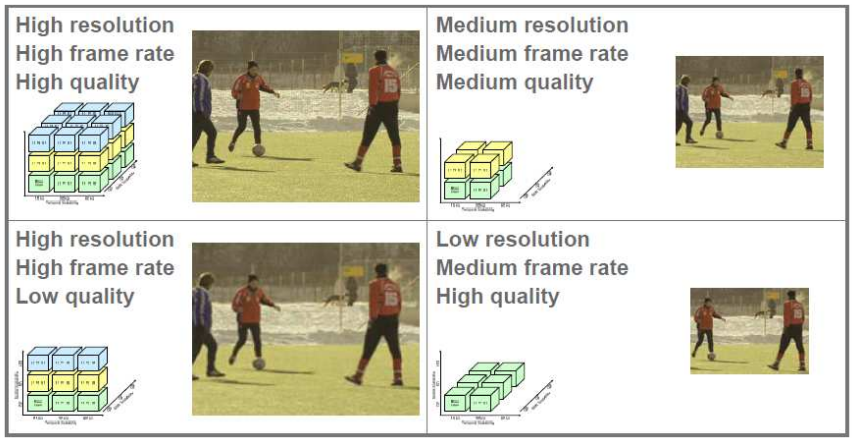
Ο Κωδικοποιητής H.264 χρησιμοποιείται για τη συμπίεση βίντεο για να μειωθεί το εύρος ζώνης που απαιτείται για τη μεταφορά του, ή να μειώσει το χώρο αποθήκευσης που απαιτείται για την αρχειοθέτηση τους. Το κόστος συμπίεσης αυξάνεται ανάλογα με τις απαιτήσεις: Όσο υψηλότερη είναι η αναλογία συμπίεσης, τόσο πιο υπολογιστική ισχύς που απαιτείται.

Για την οριοθέτηση της ανταλλαγή μεταξύ του εύρους ζώνης και της υπολογιστικής απαίτησης καθορίζεται τόσο το ελάχιστο εύρος ζώνης καναλιού που απαιτείται για να φέρει το κωδικοποιημένο codestream και την ελάχιστη προδιαγραφή της συσκευής αποκωδικοποίησης. Σε παραδοσιακά συστήματα βίντεο, όπως τηλεοπτική εκπομπή, η ελάχιστη προδιαγραφή ενός αποκωδικοποιητή (στην περίπτωση αυτή ένα set-top box) ορίζεται εύκολα.

Σήμερα, ωστόσο, το βίντεο χρησιμοποιείται όλο και περισσότερο σε ποικίλες εφαρμογές με ένα διαφορετικό σύνολο αντιστοιχιών πελάτη συσκευών-υπολογιστές από την προβολή βίντεο στο Διαδίκτυο σε φορητό ψηφιακοί βοηθοί (PDA) και ακόμη και το ταπεινό κινητό τηλέφωνο. Οι ροές βίντεο για αυτές τις συσκευές είναι κατ 'ανάγκη διαφορετικές.

Σε ένα ιδανικό σενάριο, το βίντεο θα κωδικοποιείται μόνο μία φορά με μία υψηλή απόδοση κωδικοποιητή. Έτσι όταν απαιτείται μικρότερη ανάλυσης video τότε ένα τμήμα του stream θα αποκωδικοποιείται που ανταποκρίνεται στις προδιαγραφές του τελικού χρήστη- συσκευής. Αυτό το τμήμα της ροής θα είναι πολύ πιο εύκολο να αποκωδικοποιηθεί. Έτσι ουσιαστικά η ροή μπορεί να προσαρμόζεται στις απαιτήσεις ανά περίπτωση.

Η δυνατότητα κλιμάκωσης επέκταση Codec βίντεο με το πρότυπο H.264 (H.264 SVC) έχει σχεδιαστεί για να προσφέρει τα οφέλη που περιγράφηκαν στην προηγούμενη ιδανικό σενάριο. Με βάση το βίντεο H.264 Codec ή το σύνθετο πρότυπο (H.264 AVC) σε μεγάλο βαθμό αξιοποιεί τα εργαλεία και τις έννοιες του αρχικού codec. Το κωδικοποιημένο ρεύμα που παράγει, ωστόσο, είναι επεκτάσιμο: χρονικά, χωρικά, και όσον αφορά την ποιότητα του βίντεο. Δηλαδή, μπορεί να αποφέρει αποκωδικοποίηση βίντεο με διαφορετικούς ρυθμούς καρέ, ψηφίσματα, ή τα επίπεδα ποιότητας.



Εικόνα 2.2: Χαρακτηριστικά SVC σε συνδυασμό με ποιότητα, frame rate και ανάλυση σε τυπική ροή video

Η επέκταση SVC εισάγει μια έννοια που δεν υπάρχει στο αρχικό κωδικοποιητή H.264. Ένα βασικό στρώμα κωδικοποιεί τη βασική ποιότητα video ως προς την ανάλυση την ποιότητα και το frame rate. Στοιβάδες Enhancement κωδικοποιούν πρόσθετες πληροφορίες χρησιμοποιώντας το στρώμα βάσης ως σημείο εκκίνησης, μπορούν να χρησιμοποιηθούν για την ανακατασκευή υψηλότερης ποιότητας ανάλυση, ή χρονικές εκδόσεις του βίντεο κατά τη διαδικασία αποκωδικοποίησης. Με αποκωδικοποίηση του στρώματος βάσεως και μόνο τις επόμενες στρώσεις απαιτούν ανώτερη ποιότητας αποκωδικοποιητή που μπορεί να παράγει μία ροή βίντεο με ορισμένα επιθυμητά χαρακτηριστικά. Η εικόνα 2.2 δείχνει τη στρωματική δομή ενός ρεύματος H.264 SVC. Κατά τη διάρκεια της διαδικασίας κωδικοποιούν, λαμβάνεται μέριμνα για την κωδικοποίηση ενός συγκεκριμένου στρώματος χρησιμοποιώντας αναφορά μόνο στις

κατώτερες στρώσεις επίπεδο. Με τον τρόπο αυτό, το κωδικοποιημένο codestream μπορεί να είναι τοποθετημένο σε οποιοδήποτε αυθαίρετο σημείο και να εξακολουθεί να παραμένει ένα έγκυρο και αποκωδικοποιήσιμο codestream.

2.2.2 Διαδικασία streaming

Η διαδικασία της επικοινωνίας αρχίζει από το σημείο όπου ένα ασυμπίεστο αναλογικό ή ψηφιακό σήμα συμπιέζεται και μια στοιχειώδης ροή (elementary stream) στην έξοδο του κωδικοποιητή. Η στοιχειώδης ροή ορίζεται ως μια συνεχής ροή σε πραγματικό χρόνο ψηφιακού σήματος. Μία στοιχειώδης ροή είναι ουσιαστικά η πρώτη έξοδος από τον κωδικοποιητή. Η ροή είναι οργανωμένη κατά κανόνα σε βίντεο καρέ σε αυτό το στρώμα του μοντέλου. Το είδος των πληροφοριών σε μία στοιχειώδη ροή μπορεί να περιλαμβάνει το είδος του πλαισίου και το bitrate. Είναι σημαντικό να σημειωθεί ότι αυτό το στρώμα χωρίζεται ουσιαστικά σε δύο υπό επίπεδα από την προδιαγραφή του H.264/AVC: Στρώμα κωδικοποίησης βίντεο (Video coding layer-VCL) και Στρώμα αφηρημένου δικτύου (Network abstract layer-NAL). Το VCL υπόστρωμα καθορίζει τη συμπίεση του βίντεο. Η έξοδος από αυτό το στρώμα αυτό είναι μια σειρά από φέτες (slices) εικόνας. Η ροή των bit στο NAL στρώμα οργανώνεται σε διάφορα επιμέρους πακέτα που ονομάζονται μονάδες NAL. Επίσης είναι δυνατό αυτές οι μονάδες να περιλαμβάνουν και άλλα είδη περιεχομένου όπως πληροφορίες ελέγχου.

Προκειμένου οι στοιχειώδης ροές ήχου, δεδομένων και βίντεο να μεταδοθούν μέσω του ψηφιακού δικτύου, κάθε στοιχειώδης ροή μετατρέπεται σε πακέτα (Packetized Elementary Stream). Κάθε ροή περιέχει μόνο ένα είδος δεδομένων από μία πηγή. Τα πακέτα μίας ροή μπορεί να είναι σταθερά ή όχι σε μεγέθους. Αυτό περιλαμβάνει τη διάθεση των 6bytes για την επικεφαλίδα και το υπόλοιπο του πακέτου για τη μεταφορά του περιεχομένου.

Λόγω της φύσης της δικτύωσης η σειρά ή ο αριθμός των καρέ του βίντεο που μεταδίδονται από τον αποστολέα μπορεί να διαφέρει από την σειρά ή τον αριθμό των πακέτων που λαμβάνονται από τον πελάτη Έτσι, για να βοηθήσει το συγχρονισμό, το H.264 συχνά δίνει πληροφορία χρονισμού στα πακέτα της στοιχειώδους ροής (PES) που αποτελούν μέρος του συγκεκριμένου βίντεο. Υπάρχουν δύο τύποι που μπορούν να εφαρμοστούν σε κάθε πακέτο PES – Presentation time stamps (PTS) και Decode time stamps (DTS).

PTS-Είναι μία τιμή των 33ών bit η οποία τοποθετείται στην κεφαλίδας . Ο λόγος ύπαρξής είναι ο καθορισμός της σειράς και του χρόνου που κάθε πακέτο θα υποβληθεί στο θεατή. DTS-Χρησιμοποιούνται για να ενημερώσουν τον αποκωδικοποιητή πότε ακριβώς πρέπει να αρχίσει την επεξεργασία των πακέτων. Το επίπεδο αυτό χρησιμοποιείται από ένα ευρύ φάσμα IPTV εφαρμογών. Ενεργεί ως μεσάζων μεταξύ των H.264/AVC, MPEG-2, ή VC-1 κωδικοποιήσεων των υψηλότερων επιπέδων με τα κατώτερα τμήματα του IPTVCM. Το πρωτόκολλο RTP (Ref 4) αντιπροσωπεύει τον πυρήνα αυτού του στρώματος και είναι συχνά το θεμέλιο για την αποστολή σε πραγματικό χρόνο της ροής σε ένα δίκτυο IP. Το RTP προσφέρει end-to-end μεταφορά των ροών ήχου και βίντεο ενθυλακώνοντας το περιεχόμενο σε πακέτα. Κάθε πακέτο αποτελείται από μια κεφαλίδα και το ωφέλιμο φορτίο δεδομένων. Για να βελτιωθεί η αποδοτικότητα εύρους ζώνης, το ωφέλιμο φορτίο συνήθως περιλαμβάνει περισσότερες από ένα MPEG-TS πακέτο. Η κεφαλίδα περιέχει τις λειτουργίες που είναι απαραίτητες για την επιτυχή διαβίβαση των δεδομένων σε πραγματικό χρόνο σε όλο το δίκτυο. Μια επικεφαλίδα RTP είναι αναγνωρίσιμη με την τιμή 5004 στην User Datagram Protocol (UDP) κεφαλίδα και περιέχει αρκετά μεγάλο αριθμό τομέων. Λεπτομέρειες των διαφόρων αυτών τομέων φαίνεται στο σχήμα και στον πίνακα που ακολουθούν. Πρέπει επίσης να επισημάνουμε ότι το RTP δεν έχει πεδίο μήκους στην κεφαλίδα του διότι εξαρτάται από τα υπάρχοντα πρωτόκολλα μεταφοράς για την παροχή αυτού του είδους πληροφορίας. Τα δύο βασικά πλεονεκτήματα εισαγωγής συμπιεσμένου βίντεο σε RTP πακέτα είναι:

(1) Προσθέτει έναν αριθμό σειράς για το πακέτο για να βοηθήσει τόσο το server και όσο και το IPTVCD να εντοπίσουν απώλειες πακέτων. Επιπλέον, ο αριθμός αυτός μπορεί επίσης να χρησιμοποιηθεί από τον αποκωδικοποιητή IPTVCD για την επανακατάταξη των πακέτων που φθάνουν από το δίκτυο IP σε λάθος σειρά.

(2) Το πεδίο σήμανσης χρόνου συμβάλλει στην αντιμετώπιση θεμάτων όπως το jitter και ο εσφαλμένος χρονισμός ρολογιού μεταξύ πηγής και προορισμού. κεφαλίδων στα πακέτα αλλά και κατά την μετατροπή από τη μία δικτυακή τεχνολογία σε άλλη.

Όπως υποδηλώνει το όνομα αυτού του μηχανισμού ορίζει τον κατακερματισμό της ενιαίας NAL μονάδας σε πολλά πακέτα RTP. Είναι σημαντικό να σημειωθεί ότι η κατακερματισμένη μονάδα NAL πρέπει να αποσταλεί στο δίκτυο σε μια συνεχή και διαδοχική σειρά. Αυτό είναι δυνατό μέσω της χρήσης των αριθμών που περιέχονται μέσα στην κεφαλίδα RTP. Ο μηχανισμός αυτός έχει πλεονεκτήματα για τους παρόχους υπηρεσιών IPTV. Πρώτα απ 'όλα να συμβάλλει στην επίτευξη μετάδοσης μεγαλύτερης ποσότητας υψηλής ευκρίνειας περιεχομένου βίντεο και δεύτερον να

συμβάλλει στη βελτίωση της αποτελεσματικότητα των τεχνικών διόρθωσης σφαλμάτων, όπως είναι Forward Error Correction (FEC).

Κεφάλαιο 3

Κρυπτογράφηση

3.1 SSL – Secure Socket Layer

Το πρωτόκολλο SSL χρησιμοποιείται για κρυπτογράφηση δεδομένων που πρόκειται να μεταδοθούν. Το πρωτόκολλο αυτό συναντάται σε τυπικές συνδέσεις δικτύου για την ασφαλή μεταφορά δεδομένων είτε κατά την εκτέλεση μιας εφαρμογής είτε για την εκτέλεσε πολλών εφαρμογών ταυτοχρόνως SSL tunnels.

3.1.1 SSL τυπικές συνδέσεις

Χωρίς αμφιβολία η πιο διαδεδομένη μέθοδος VPN είναι βασισμένη στο web-based SSL. Πολλά sites χρησιμοποιούν SSL για να παρέχουν ασφαλή σύνδεση μεταξύ των web-server και των χρηστών. Οι περισσότεροι χρήστες δεν γνωρίζουν για το SSL εκτός από την κλειδαριά που

εμφανίζεται στον web-browser όταν χρησιμοποιείται από όλους του δημοφιλείς web browsers. Η τυπική TCP θύρα για web based επικοινωνία είναι η 80, αντίστοιχα για SSL κρυπτογράφηση χρησιμοποιείται η θύρα TCP 443. Αν και συνήθως το SSL συσχετίζεται με το HTTP μπορεί να χρησιμοποιηθεί για κρυπτογράφηση και σε άλλα πρωτόκολλα όπως SMTP, NNTP, LDAPS, IMAP, POP3,

3.1.2 SSL εγκατάσταση στον τελικό χρήστη

SSL είναι εύκολη η εγκατάσταση στον τελικό χρήστη. Οι περισσότερες web-based εφαρμογές έχουν επιλογή για ενεργοποίηση SSL. Οι ρυθμίσεις και η διαδικασία είναι σχετικά απλή. Σε περιπτώσεις όπου η εφαρμογή δεν υποστηρίζει SSL τότε η χρήση SSL-tunneling είναι η πιο εύκολη και η πιο διαδεδομένη.

3.1.3 SSL εγκατάσταση στον server

Η ενεργοποίηση του SSL σε ένα SSL web-based server είναι μια απλή διαδικασία. Το SSL δημιουργεί ένα πιστοποιητικό με ψηφιακή υπογραφή, οπότε η βασική δουλειά είναι να μπορεί να δημιουργεί και να εγκαταστήσει τα σωστά πιστοποιητικά. Αρχικά το πρόγραμμα δημιουργεί ένα ζευγάρι κλειδιών και μια αίτηση για πιστοποιητικό, το οποίο περιέχει το κλειδί για την κωδικοποίηση. Στη συνέχεια αποστέλλεται η αίτηση στους αντίστοιχους χρήστες που ελέγχει τις αιτήσεις αν είναι αυθεντικές. Ο τελικός χρήστης υπογράφει το πιστοποιητικό και αποστέλλει κωδικοποιημένο πια το κλειδί επιβεβαιώνοντας την ορθότητα της κρυπτογράφησης. Έτσι από το σημείο αυτό και μετά η επικοινωνία βασίζεται στο SSL και θεωρείται ασφαλής.

Όταν εγκαθίσταται μια σύνδεση SSL είναι σημαντικό να τηρείται ανάλογα με την περίπτωση το επίπεδο της κρυπτογράφησης. Οι πρόσφατοι web-based εφαρμογές υποστηρίζουν κρυπτογράφηση 128bit ή μεγαλύτερη αλλά παλιότερες εκδόσεις υποστηρίζουν 40-bit και 56-bit κρυπτογράφηση. Έτσι χρήστες με παλιότερες εκδόσεις ίσως να μην έχουν πρόσβαση στις εφαρμογές.

3.1.4 Πότε χρησιμοποιείται SSL

η πιο προφανή περίπτωση χρήσης SSL είναι όταν θέλουμε κρυπτογράφηση για μια HTTP επικοινωνία. Ωστόσο αυτό μπορεί να επεκταθεί και να γίνει το SSL πολύ πιο χρήσιμο. Για παράδειγμα το Outlook της microsoft έχει αντίστοιχο πρόγραμμα web based. Έτσι απαιτείται browser που να υποστηρίζει 128-bit κλειδιά κρυπτογράφησης. Το SSL σε αυτές τις περιπτώσεις είναι αόρατο σε ένα απλό χρήστη.

Αντίστοιχα σε διαδικασίες που δεν βασίζονται σε web-based εφαρμογές όπως το POP και IMAP υπάρχουν μεγαλύτερες δυσκολίες. Παρόλο που πολλές εφαρμογές υποστηρίζουν SSL υπάρχουν αρκετές που δεν το υποστηρίζουν. Για παράδειγμα το Outlook της Microsoft έχει επιλογές για ενεργοποίηση του SSL για την προστασία SMTP και IMAP.

3.1.5 SSL tunnels

Ένας λιγότερο γνωστός τρόπος για εγκατάσταση SSL είναι η χρήση SSL tunneling. Αφού το SSL tunneling έχει εγκατασταθεί μπορούν να τρέξουν πολλά πρωτόκολλα, Η εγκατάσταση γίνεται με τον βασικό κώδικα να τρέχει στο server και αντίστοιχος client στους χρήστες. Χρησιμοποιείται τεχνολογία port fowrwarding όπου τα κρυπτογραφημένα δεδομένα προωθούνται για αποστολή και λαμβάνονται από την αντίστοιχη θύρα του client όπου και γίνεται η αποκωδικοποίηση τους.

3.1.6 Χρήσεις SSL tunneling

Χρησιμοποιείται συνήθως για την εκτέλεση ασφαλούς πρωτοκόλλου σε κοινά δίκτυα επικοινωνίας. Το βασικό πλεονέκτημα είναι ότι μπορεί να εγκατασταθεί πολύ εύκολα και με ελάχιστο κόστος.

3.2 Αλγόριθμοι Κρυπτογράφησης

3.2.1 AES

AES είναι ακρωνύμιο του Advanced Encryption Standard και θεσπίστηκε από το Αμερικάνικο Ινστιτούτο Τεχνολογίας και standards (NIST) το 2001. Τώρα ο AES χρησιμοποιείται παγκοσμίως και δημιουργήθηκε για να αντικαταστήσει τον DES. Ο AES είναι αλγόριθμος που χρησιμοποιεί συμμετρικό κλειδί. Αυτό σημαίνει ότι το κλειδί που χρησιμοποιείται για την κωδικοποίηση χρησιμοποιείται και για την αποκωδικοποίηση.

Ο AES βασίζεται σε block μεγέθους 128bits και σε κλειδιά 128 – 192 – 256bits. Το μέγεθος των κλειδιών υποδηλώνει και το πλήθος των επαναλήψεων που θα συμβούν για την κωδικοποίηση των δεδομένων.

10 κύκλοι επαναλήψεων για κλειδιά 128bit

12 κύκλοι επαναλήψεων για κλειδιά 192bit

14 κύκλοι επαναλήψεων για κλειδιά 256bit

Κάθε κύκλος αποτελείται από πολλά βήματα, συμπεριλαμβανομένου και ένα βήμα που εξαρτάται από το ίδιο το κλειδί. Τα ίδια αντίστροφα βήματα χρησιμοποιούνται για την αποκωδικοποίηση των δεδομένων.

1. Key- expansion- δημιουργία κλειδιών που προέρχονται από το αρχικό κλειδί

2. Αρχικός κύκλος

    1. κάθε byte ενώνεται με το κυκλικό κλειδί με την χρήση biwise xor.

3. κύκλοι

    1. Subbytes μη γραμμική αντικατάσταση όπου κάθε byte αντικαθίσταται με ένα άλλο

    2. shiftrows όπου οι γραμμές των δεδομένων αλλάζουν κυκλικά θέση

3. mixcolumns όπου γίνεται συνδυασμός των 4 bytes ανά στήλη.

4. προσθήκη κυκλικού κλειδιού

4. Τελικός κύκλος

1.subbytes

2.shiftrounds

3. προσθήκη κυκλικού κλειδιού

3.2.2 DES

Η Data Encryption Standard (DES), είναι το όνομα της Federal Information Processing Standard (FIPS) 46-3, το οποίο περιγράφει τον αλγόριθμο κρυπτογράφησης δεδομένων (DEA). Ο DEA επίσης ορίζεται με το πρότυπο ANSI X3.92. επίσης είναι μια βελτίωση του αλγορίθμου Lucifer που αναπτύχθηκε από την IBM στις αρχές του 1970. Η IBM, η Υπηρεσία Εθνικής Ασφάλειας (NSA) και το Εθνικό Γραφείο Προτύπων (NBS ,σήμερα Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας NIST) ήταν οι υπηρεσίες που ανέπτυξαν τον αλγόριθμο. Το DES έχει μελετηθεί εκτενώς από τη δημοσίευσή του και είναι ο πιο ευρέως χρησιμοποιούμενος Συμμετρικός αλγόριθμος στον κόσμο. Το DES είναι 64-bit και χρησιμοποιεί ένα 56-bit κλειδί κατά τη διάρκεια της εκτέλεσης (έχει 8 bits ισοτιμίας από το πλήρες κλειδί 64-bit).Ο DES είναι ένα συμμετρικό κρυπτογραφικό σύστημα, και συγκεκριμένα ένα 16-γύρο cipher Feistel. Όταν χρησιμοποιείται για την επικοινωνία, τόσο αποστολέας και ο παραλήπτης πρέπει να γνωρίζει το ίδιο μυστικό κλειδί, το οποίο μπορεί να χρησιμοποιηθεί για την κρυπτογράφηση και την αποκρυπτογράφηση του μηνύματος, ή για τη δημιουργία και την επαλήθευση ενός κώδικα ταυτότητας μηνυμάτων (MAC). Ο DES μπορεί επίσης να χρησιμοποιηθεί για μεμονωμένους χρήστες κρυπτογράφησης, όπως για την αποθήκευση αρχείων σε έναν σκληρό δίσκο σε κρυπτογραφημένη μορφή.

Ο DES είναι αρχετυπικός block cipher, δηλαδή, ένας πρωτότυπος κρυπταλγόριθμος συμμετρικού κλειδιού, που λαμβάνει μια σειρά από bits απλού κειμένου (plaintext bits) σταθερού μήκους και την μετατρέπει, μέσω μιας σειράς πολύπλοκων ενεργειών, σε μια άλλη σειρά bits, το κρυπτοκείμενο (chiphertext) με το ίδιο μήκος. Στην περίπτωση του DES το μέγεθος μπλοκ (block size: Η σειρά των bits σταθερού μήκους) είναι 64 bits. Ο DES χρησιμοποιεί, επίσης, ένα κλειδί για

να προσαρμόσει την μετατροπή, ώστε η αποκρυπτογράφηση να μπορεί, υποθετικά, να πραγματοποιηθεί μόνο από εκείνους που γνωρίζουν το συγκεκριμένο κλειδί που χρησιμοποιήθηκε για την κρυπτογράφηση. Το κλειδί φαινομενικά αποτελείται από 64 bits. Ωστόσο, στην πραγματικότητα μόνο 56 από αυτά χρησιμοποιήθηκαν από τον αλγόριθμο. Τα υπόλοιπα 8 bits χρησιμοποιούνται αποκλειστικά για τον έλεγχο της ισοτιμίας (parity) και στη συνέχεια απορρίπτονται (αυτά καλούνται parity bits), εξ ου και αναφέρεται συνήθως ως κλειδί μήκους 56 bits. Όπως οι άλλοι block αλγόριθμοι κρυπτογράφησης, έτσι και ο DES από μόνος του δεν είναι ασφαλής τρόπος κρυπτογράφησης αλλά, αντίθετα, πρέπει να χρησιμοποιηθεί με ειδικό τρόπο λειτουργίας (mode of operation).

Ο αλγόριθμος που δημιουργεί τα υποκλειδιά. Αρχικά, 56 bits του κλειδιού επιλέγονται από τα αρχικά 64 από τη μεταλλαγμένη επιλογή 1 (Permuted Choice 1 : PC-1) — τα υπόλοιπα οκτώ bits είτε απορρίπτονται είτε χρησιμοποιούνται ως parity bits (για τον έλεγχο ισοτιμίας). Τα 56 bits διαιρούνται έπειτα σε δύο τμήματα των 28 και κάθε μισό αντιμετωπίζεται έκτοτε χωριστά. Στους διαδοχικούς γύρους και τα δύο μισά περιστρέφονται αριστερά κατά ένα ή δύο bits (που διευκρινίζονται για κάθε γύρο) και έπειτα, 48 bits του υποκλειδιού επιλέγονται από τη μεταλλαγμένη επιλογή 2 (Permuted Choice 2 : PC-2) — 24 bits από το αριστερό μισό και 24 από το bits μισό. Οι περιστροφές σημαίνουν ότι ένα διαφορετικό σετ από bits χρησιμοποιείται σε κάθε υποκλειδί. Κάθε bit χρησιμοποιείται σε περίπου 14 από τα 16 υποκλειδιά.

Το πρόγραμμα κλειδιού για την αποκρυπτογράφηση είναι παρόμοιο — τα υποκλειδιά είναι σε αντίστροφη διάταξη έναντι αυτή της κρυπτογράφησης. Αν, δηλαδή, κατά την κρυπτογράφηση το πρόγραμμα κλειδιού είναι {k1, k2, k3 ... k16}, τότε το πρόγραμμα κλειδιού της αποκρυπτογράφησης θα είναι {k16 ... k3, k2, k1}. Πέραν αυτής της αλλαγής, η διαδικασία είναι η ίδια όπως για την κρυπτογράφηση.

3.2.3 3DES

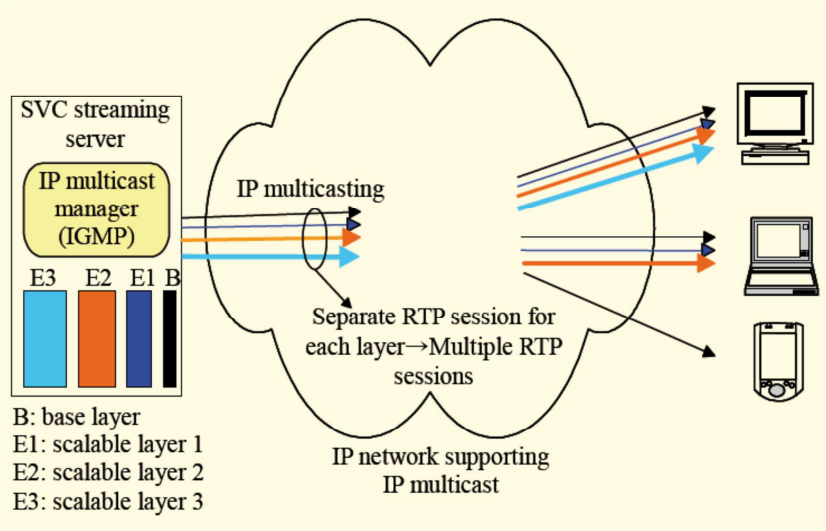
Το σύστημα αυτό δεν είναι τίποτα άλλο παρά 3 συστήματα DES σε σειρά. Η έξοδος δηλαδή του πρώτου συστήματος είναι η είσοδος του δευτέρου του οποίου η έξοδος είναι η είσοδος του τρίτου. Προφανώς αποδίδει καλύτερη κρυπτογράφηση διότι πρέπει να υποκλαπούν 3 κλειδιά και όχι ένα.

Κεφάλαιο 4

Εξομοιωτής CAS - Ανάλυση

Για την μελέτη του συστήματος CAS θα πρέπει να επιλεγεί ένας αλγόριθμος κρυπτογράφησης ο οποίος θα παρέχει ικανοποιητικό επίπεδο ασφάλειας χωρίς να δημιουργεί μεγάλο overhead κατά την μετάδοση των δεδομένων του video. Η διαδικασία βασίζεται στη λογική να χρησιμοποιηθεί ένα πρόγραμμα κρυπτογράφησης για να κωδικοποιεί το video και να χρονομετρηθεί η διαδικασία κρυπτογράφησης ενώ παράλληλα θα γίνεται σύγκριση με το μέγεθος του video ώστε να διαπιστωθεί η καθυστέρηση που δημιουργείται. Έτσι χρησιμοποιείται ο ανοικτός κώδικας openssl για τα διαφορετικά είδη κρυπτογράφησης προς έλεγχο.

Η μεταφορά πακέτων μέσω IP streaming χρησιμοποιείται το μοντέλο όπως αυτό φαίνεται στην εικόνα 4-1. Ο server ανοίγει ένα κανάλι επικοινωνίας πραγματικού χρόνου (RTP) για να μεταφέρει τα layers όπως αυτά δημιουργήθηκαν κατά την H.264 συμπίεση του video. Για κάθε τελικό χρήστη ο server δημιουργεί ένα bitstream σύμφωνα με τις ανάγκες του τελικού χρήστη με τα αντίστοιχα NAL πακέτα . Σε αυτό το κανάλι επικοινωνίας γίνεται η μεταφορά όλων των πακέτων που εμπεριέχουν όλα τα layers. Στη συνέχεια κάθε συσκευή δέχεται και αποκωδικοποιεί των αριθμό των layers ανάλογα με τις δυνατότητες που έχει και έχουμε την τελική εικόνα. Έτσι ουσιαστικά ο CAS που προτείνεται μπορεί να εγκατασταθεί αμέσως μετά των streamer αποφεύγοντας έτσι μεγαλύτερο φορτίο κατά την αποστολή των δεδομένων[13].



Εικόνα 4-1: Μεταφορά SVC streaming με πολλαπλά sessions πραγματικού χρόνου

Τα δεδομένα για την εξομοίωση του CAS server που χρησιμοποιήθηκαν ήταν πακέτα από video streaming. Κάθε frame αποτελείται από 3 διαφορετικά layers. Το video έχει καταγραφεί με 30fps(frames per sec). Έτσι κάθε frame μπορεί να ειπωθεί ότι μεταφέρει το 1/30 sec του video. Αυτό είναι απαραίτητο για να δούμε πόση είναι η συνολική καθυστέρηση εξαιτίας της κρυπτογράφησης. Κάθε πακέτα για λόγους εξομοίωσης μπορεί να εκφρασθεί με ένα string αντιστοίχων bytes και να ζητηθεί από το OpenSSL να το κρυπτογραφήσει. Έπειτα μετρώντας το χρόνο για κάθε κρυπτογράφηση να εξαχθεί και ο χρόνος που απαιτείται για ένα αντίστοιχο streaming[05].

Για το χειρισμό του openssl χρησιμοποιήθηκε γλώσσα προγραμματισμού C, ο κώδικας συμπεριλαμβάνεται στο παράρτημα. Τα αποτελέσματα μελετήθηκαν με τη χρήση Excel και παρουσιάζονται παρακάτω.

Οι αλγόριθμοι κρυπτογράφησης που χρησιμοποιήθηκαν είναι ο αλγόριθμος AES-CBC ο αλγόριθμος DES-CBC και ο DES3-CBC με την χρήση 128bit, 192bit και 256bit keys. Οι αλγόριθμοι αυτοί είναι ευρέως διαδεδομένοι ο AES ως ποιο σύγχρονος ενώ ο DES και 3DES είναι πιο παλαιοί συνεχίζουν όμως να δίνουν ικανοποιητικά αποτελέσματα. Είναι αλγόριθμοι που τους υποστηρίζει το openssl και μπορούν εύκολα να χρησιμοποιηθούν για την κρυπτογράφηση πακέτων video.

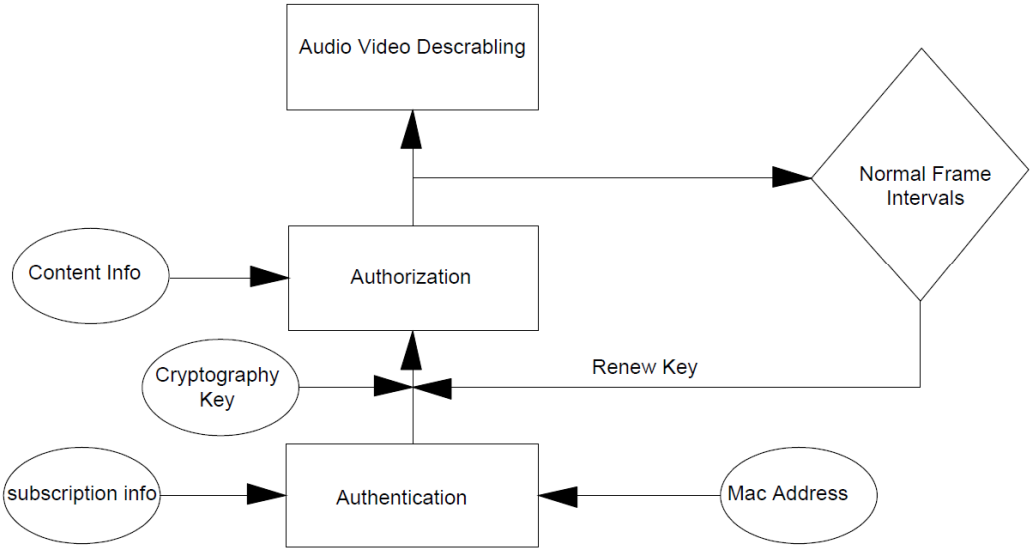
Σε επίπεδο κρυπτογράφησης η χρήσης της openssl μπορούμε να μελετήσουμε διαφορετικές περιπτώσεις προς σύγκριση. Επιλέχθηκαν οι ακόλουθες μορφές κρυπτογράφησης:

1. Κρυπτογράφηση με την χρήση είτε με 128bit, είτε με 192 bit, είτε με 256bit key. Η κρυπτογράφηση γίνεται σε όλα τα layers. Με τον τρόπο αυτό πετυχαίνεται μεγάλο επίπεδο ασφαλείας ειδικά με την χρήση κλειδιού 256bit αλλά υπάρχει ο κίνδυνος δημιουργίας μεγάλης καθυστέρησης για την κωδικοποίηση και την αποκωδικοποίηση όλων των layers.

2. Κρυπτογράφηση χρησιμοποιώντας διαφορετικά επίπεδα ασφαλείας για κάθε layer βασιζόμενοι στο γεγονός ότι το layer 0 είναι το βασικότερο για να μπορεί ο τελικός χρήστης να δει την εικόνα. Έτσι χρησιμοποιούμε κλειδιά 256bit για να κρυπτογραφηθεί το layer 1 και κλειδιά 192 και 128bit αντίστοιχα για τα layers 1 και 2. Με τον τρόπο αυτό διατηρείτε σε ψηλό επίπεδο η ασφάλεια αλλά κερδίζουμε σε επίπεδο καθυστέρησης κρυπτογράφησης.

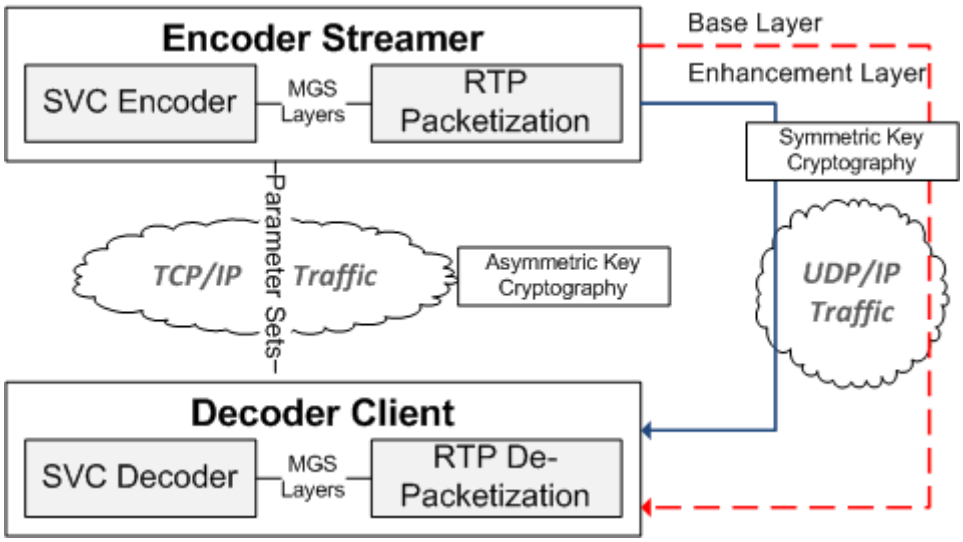
3. Κρυπτογράφηση μόνο του layer 0 με την χρήση 256bit key. Με αυτή την περίπτωση ελέγχεται κατά πόσο μπορεί να μειωθεί αισθητά ο χρόνος κρυπτογράφησης διατηρώντας επίπεδα ασφαλείας τα ελάχιστα δυνατά, χωρίς να διακινδυνεύει αποκωδικοποίηση της εκπομπής από μη εξουσιοδοτημένους χρήστες.

Με βάσει τις προηγούμενες περιπτώσεις κρυπτογράφησης, δημιουργήθηκε ο απαραίτητος κώδικας σε C. Για την εξομοίωση ενός συστήματος CAS πρέπει να αποφασισθεί μια διαδικασία κωδικοποίησης. Συγκεκριμένα:



Διάγραμμα 1: Διάγραμμα ροής CAS

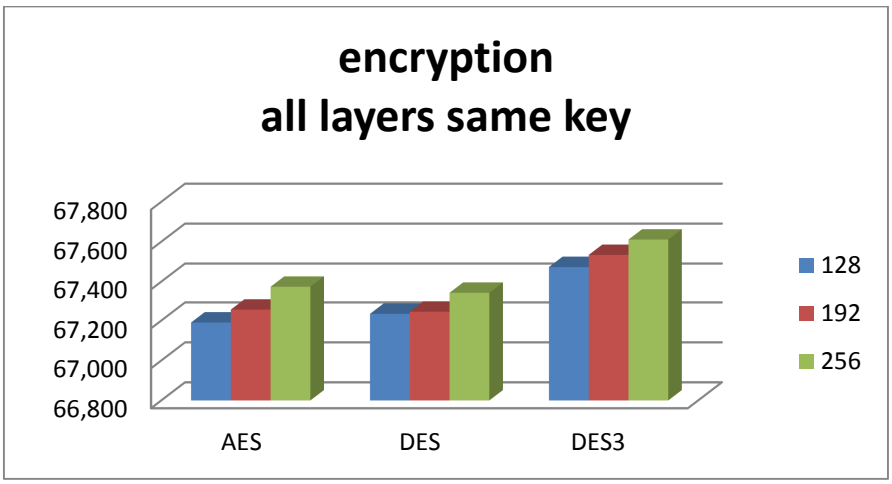
Ο CAS server μετά το απαραίτητο handshake όπου γίνεται authentication και authorization του client, αποστέλλει τα απαραίτητα κλειδιά για την αποκωδικοποίηση του video με RSA κρυπτογράφηση αφού αυτή θεωρείται ασφαλέστερη για μια τέτοια διαδικασία (Διάγραμμα 1). Ο χρήστης παραλαμβάνει τα κωδικοποιημένα πακέτα τα αποκωδικοποιεί και παραλαμβάνει τα κλειδιά. Στην συνέχεια ο server αφού διαπιστώσει ότι ο χρήστης είναι έτοιμος να δεχθεί τα πακέτα του video ξεκινάει την αποστολή τους. Ο χρήστης λαμβάνει τα πακέτα τα αποκωδικοποιεί και προβάλει το video. Για λόγους ασφαλείας τα κλειδιά αποκρυπτογράφησης ανανεώνονται μετά από ένα συγκεκριμένο χρονικό διάστημα. Στο διάγραμμα 2 φαίνεται με απλό τρόπο η διαδικασία κωδικοποίησης και αποκωδικοποίησης σε video codestream.



Διάγραμμα 2: Διάγραμμα κωδικοποίησης και αποκωδικοποίησης video

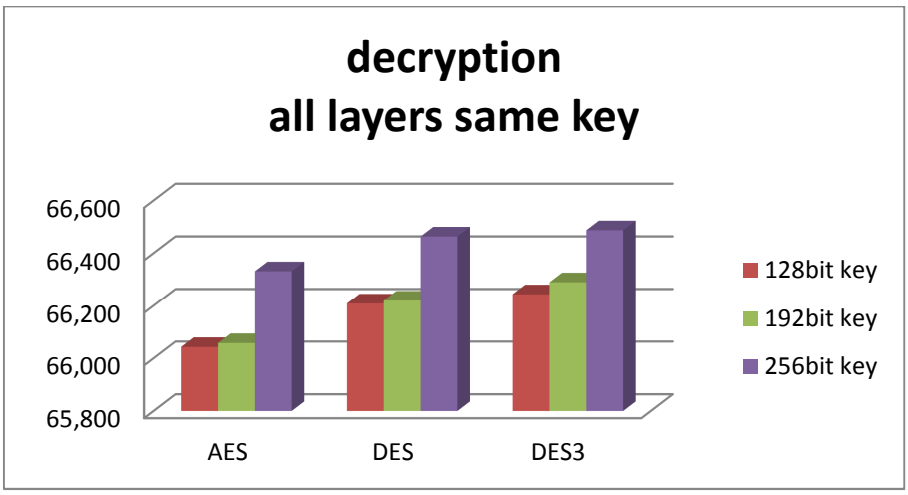
Για την εξομοίωση ενός τέτοιου συστήματος CAS μελετούμε τους χρόνους κωδικοποίησης στον server και τους χρόνους αποκωδικοποίησης στον χρήστη. Έτσι χρησιμοποιώντας δεδομένα αποστολής πακέτων για την τυπική μετάδοση ενός video 1373 frames με χρήση 3 layers ένα βασικό το layer 0 και 2 δευτερεύοντα layer 1, layer 2 με συνολική διάρκεια 45.7sec πρέπει να γίνει μελέτη των χρόνων κωδικοποίησης των πακέτων. Συγκεκριμένα το αρχείο των δεδομένων περιέχει τον αύξων αριθμό του πακέτου αποστολής, τον αύξων αριθμό του frame, το layerid, και το μέγεθος του πακέτου. Ο κώδικας έχει στόχο να διαβάσει τις τιμές από το αρχείο και να δημιουργεί πακέτο ανάλογου μεγέθους προς κωδικοποίηση. Στην συνέχεια κάνοντας χρήση του ανοιχτού κώδικα openssl γίνεται η κωδικοποίηση του πακέτου. Για την εξομοίωση της διαδικασίας ορίζεται ότι μετά από 12 frames γίνεται ανανέωση των κλειδιών. Η όλη διαδικασία χρονομετρείται ώστε να μπορούν να εξαχθούν συμπεράσματα.

Τρέχοντας τον κώδικα προέκυψαν μετρήσεις που φαίνονται στον ακόλουθο πίνακα για κάθε περίπτωση. Όπως φαίνεται από τον πίνακα η μείωση του bit κλειδιού δεν έδωσε εξαιρετικά καλύτερα αποτελέσματα καθυστέρησης αφού οι διαφορές είναι ελάχιστες.



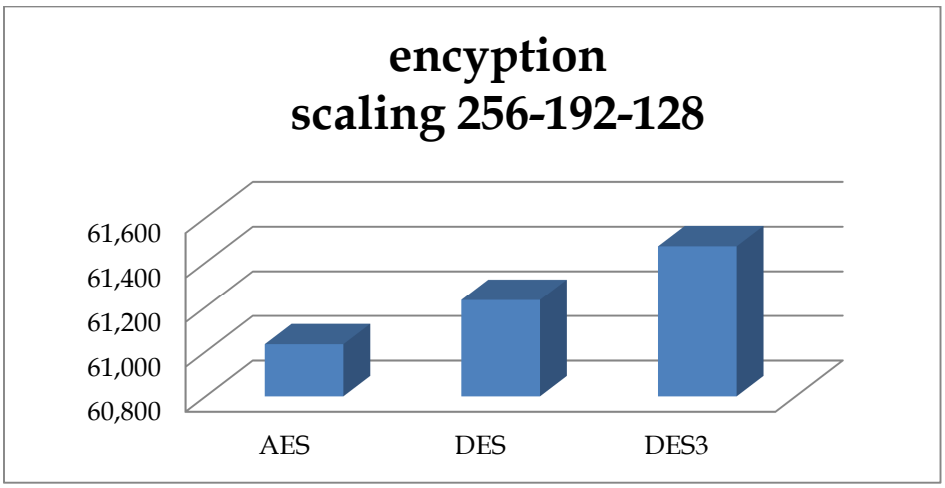
ΜΕΣΗ ΤΙΜΗ ΧΡΟΝΟΥ encryption ΑΝΑ frame ΓΙΑ ΔΙΑΦΟΡΕΤΙΚΟΥΣ ΑΛΓΟΡΙΘΜΟΥΣ ΚΡΥΠΤΟΓΡΑΦΗΣΗΣ (msec)			
	AES	DES	DES3
128bit key	67,190	67,234	67,473
192bit key	67,255	67,245	67,534
256bit key	67,375	67,344	67,611

Πίνακας 4.1 Κρυπτογράφηση με την χρήση AES – DES- DES3 encryption σε όλα τα layers



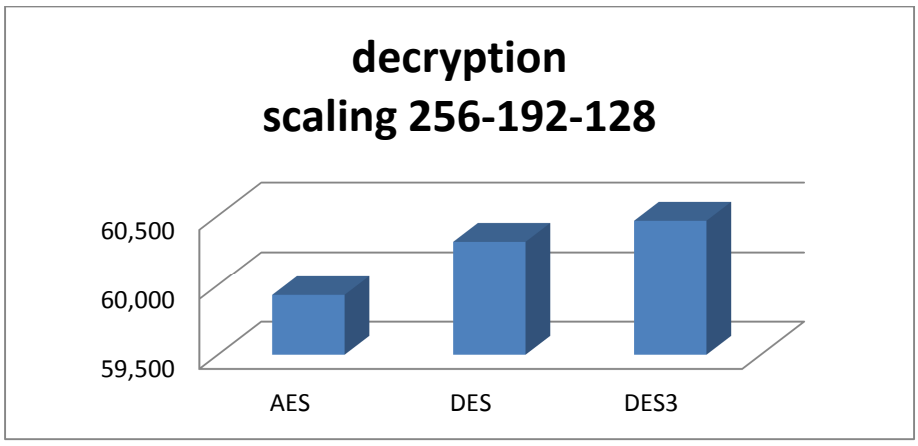
ΜΕΣΗ ΤΙΜΗ ΧΡΟΝΟΥ decryption ΑΝΑ frame ΓΙΑ ΔΙΑΦΟΡΕΤΙΚΟΥΣ ΑΛΓΟΡΙΘΜΟΥΣ ΚΡΥΠΤΟΓΡΑΦΗΣΗΣ (msec)			
	AES	DES	DES3
128bit key	66,043	66,209	66,244
192bit key	66,058	66,220	66,289
256bit key	66,331	66,463	66,487

Πίνακας 4.2 Αποκρυπτογράφηση με την χρήση AES – DES- DES3 decryption σε όλα τα layers



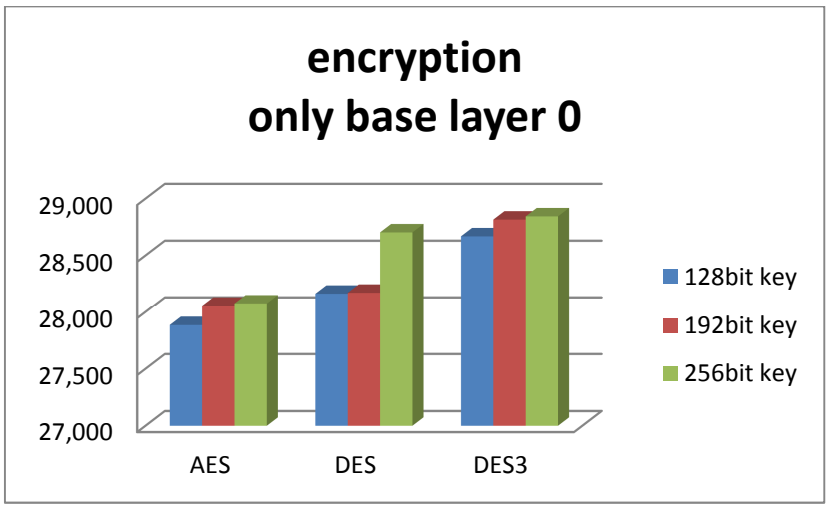
ΜΕΣΗ ΤΙΜΗ ΧΡΟΝΟΥ encoding ΓΙΑ ΔΙΑΦΟΡΕΤΙΚΟΥΣ ΑΛΓΟΡΙΘΜΟΥΣ ΚΡΥΠΤΟΓΡΑΦΗΣΗΣ σε msec	
AES	61,033
DES	61,232
DES3	61,473

Πίνακας 4.3 Κρυπτογράφηση με την χρήση AES – DES- DES3 encryption σε όλα τα layers με κλιμακωτή χρήση κλειδιών ανά layer



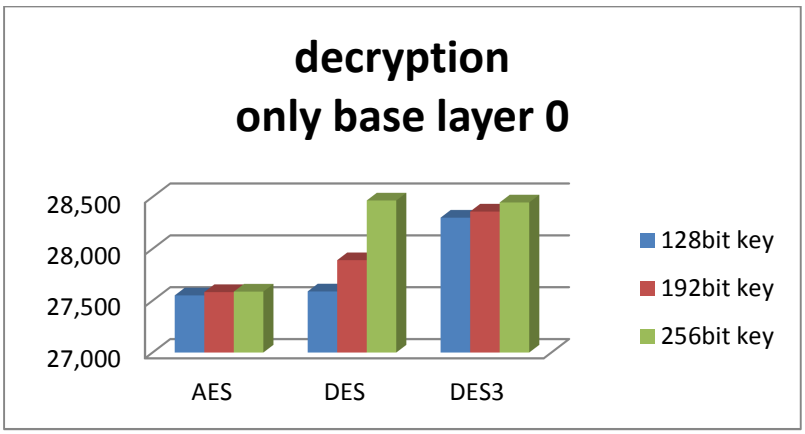
ΜΕΣΗ ΤΙΜΗ ΧΡΟΝΟΥ encoding ΓΙΑ ΔΙΑΦΟΡΕΤΙΚΟΥΣ ΑΛΓΟΡΙΘΜΟΥΣ ΑΠΟΚΡΥΠΤΟΓΡΑΦΗΣΗΣ σε msec	
AES	59,928
DES	60,313
DES3	60,464

Πίνακας 4.4 Αποκρυπτογράφηση με την χρήση AES – DES- DES3 decryption σε όλα τα layers με κλιμακωτή χρήση κλειδιών ανά layer



ΜΕΣΗ ΤΙΜΗ ΧΡΟΝΟΥ encryption ΓΙΑ ΔΙΑΦΟΡΕΤΙΚΟΥΣ ΑΛΓΟΡΙΘΜΟΥΣ ΚΡΥΠΤΟΓΡΑΦΗΣΗΣ ΣΤΟ BASE LAYER 0 (msec)			
	AES	DES	DES3
128bit key	27,884	28,162	28,668
192bit key	28,056	28,170	28,816
256bit key	28,076	28,703	28,844

Πίνακας 4.5 Κκρυπτογράφηση με την χρήση AES – DES- DES3 encryption μόνο στο base layer.



ΜΕΣΗ ΤΙΜΗ ΧΡΟΝΟΥ decryption ΓΙΑ ΔΙΑΦΟΡΕΤΙΚΟΥΣ ΑΛΓΟΡΙΘΜΟΥΣ ΚΡΥΠΤΟΓΡΑΦΗΣΗΣ σε msec

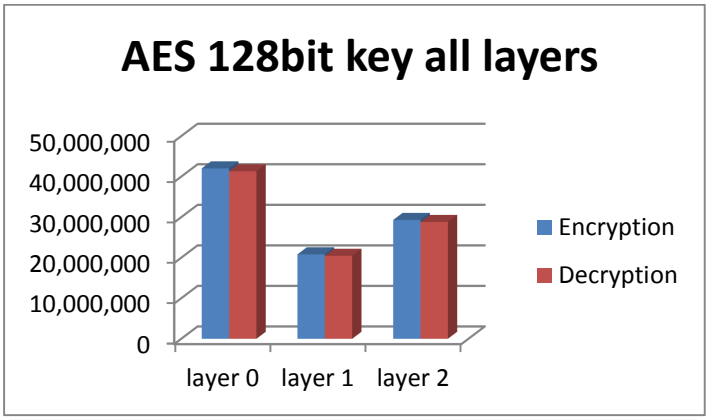
	AES	DES	DES3
128bit key	27,551	27,589	28,300
192bit key	27,583	27,891	28,360
256bit key	27,587	28,466	28,447

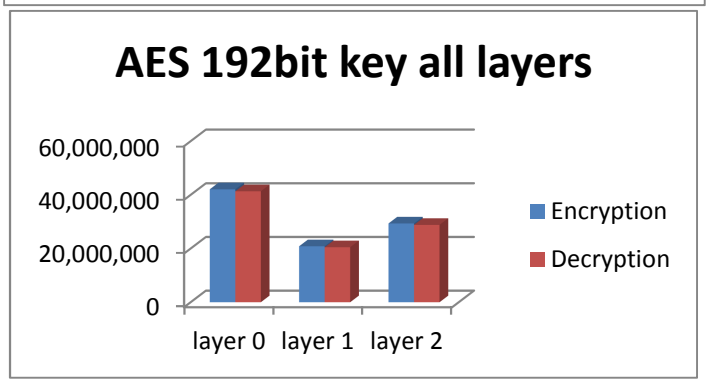
Πίνακας 4.6 Αποκρυπτογράφηση με την χρήση AES – DES- DES3 decryption μόνο στο base layer.

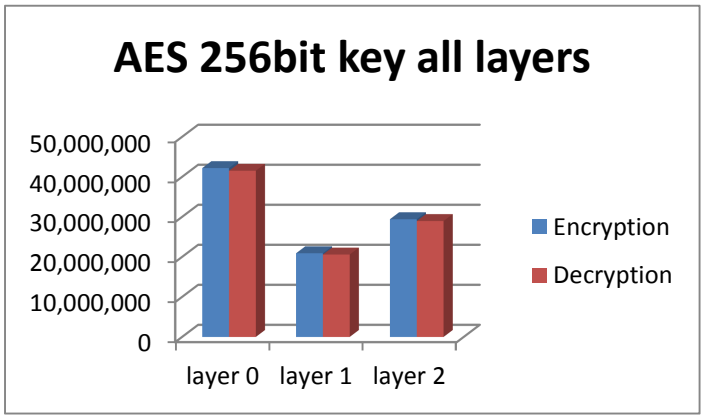
Είναι σαφές ότι η κρυπτογράφηση του base layer 0 δίνει σαφώς μικρότερους χρόνους καθυστέρησης. Συγκρίνοντας τις κρυπτογραφήσεις με βάσει το μέγεθος του κλειδιού παρατηρούμε ότι το μεγαλύτερο κλειδί εμπεριέχει μια πιο πολύπλοκη διαδικασία και αυτό έχει αντίκτυπο στους χρόνους. Οι διαφορές όμως με τους χρόνους με μικρότερα κλειδιά δεν μπορούν να χαρακτηριστούν απαγορευτικές για την χρήση κλειδιών 256bit. Αντίστοιχα ο αλγόριθμος AES όπως αναμενόταν φαίνεται σε κάθε περίπτωση να δίνει καλύτερα αποτελέσματα.

Η στατιστική μελέτη των αποτελεσμάτων δείχνει ότι για διάστημα εμπιστοσύνης 95% οι χρόνοι κρυπτογράφησης και αποκρυπτογράφησης για κάθε περίπτωση κυμαίνονται μεταξύ 0.4msec και 0.3msec. Αυτό σημαίνει ότι παρά το γεγονός ότι κατά τις μετρήσεις σε ένα τυπικό υπολογιστή όπου παράλληλα με τον κώδικα C που εκτελέστηκε έτρεχαν και άλλες διεργασίες στον υπολογιστή, οι μετρήσεις είναι πολύ κοντά μεταξύ τους και φαίνεται ότι δεν έχουν μεγάλες αποκλίσεις.

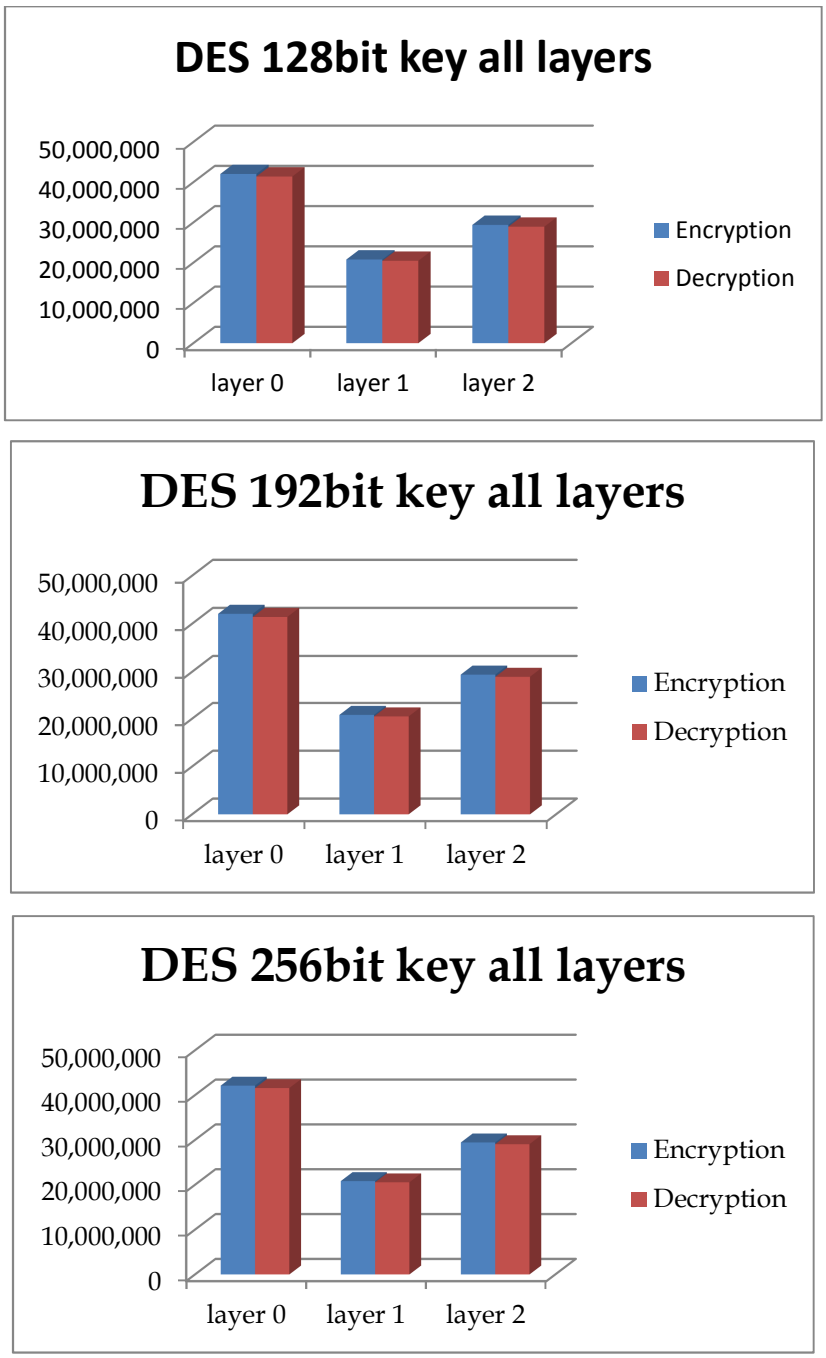
Μελετώντας την διαδικασία κρυπτογράφησης ανά layer βλέπουμε στα ακόλουθα διαγράμματα ότι σε κάθε περίπτωση το layer 0 χρειάζεται τον μεγαλύτερο χρόνο κρυπτογράφησης. Το layer 1 θέλει τον λιγότερο χρόνο από το layer 2 ακόμα και στην περίπτωση που το layer 1 έχει μεγαλύτερο κλειδί κρυπτογράφησης 192bit έναντι 128bit. Αυτό συμβαίνει γιατί το layer 2 έχει κατά κανόνα πολύ μεγαλύτερο μέγεθος από το layer 1.



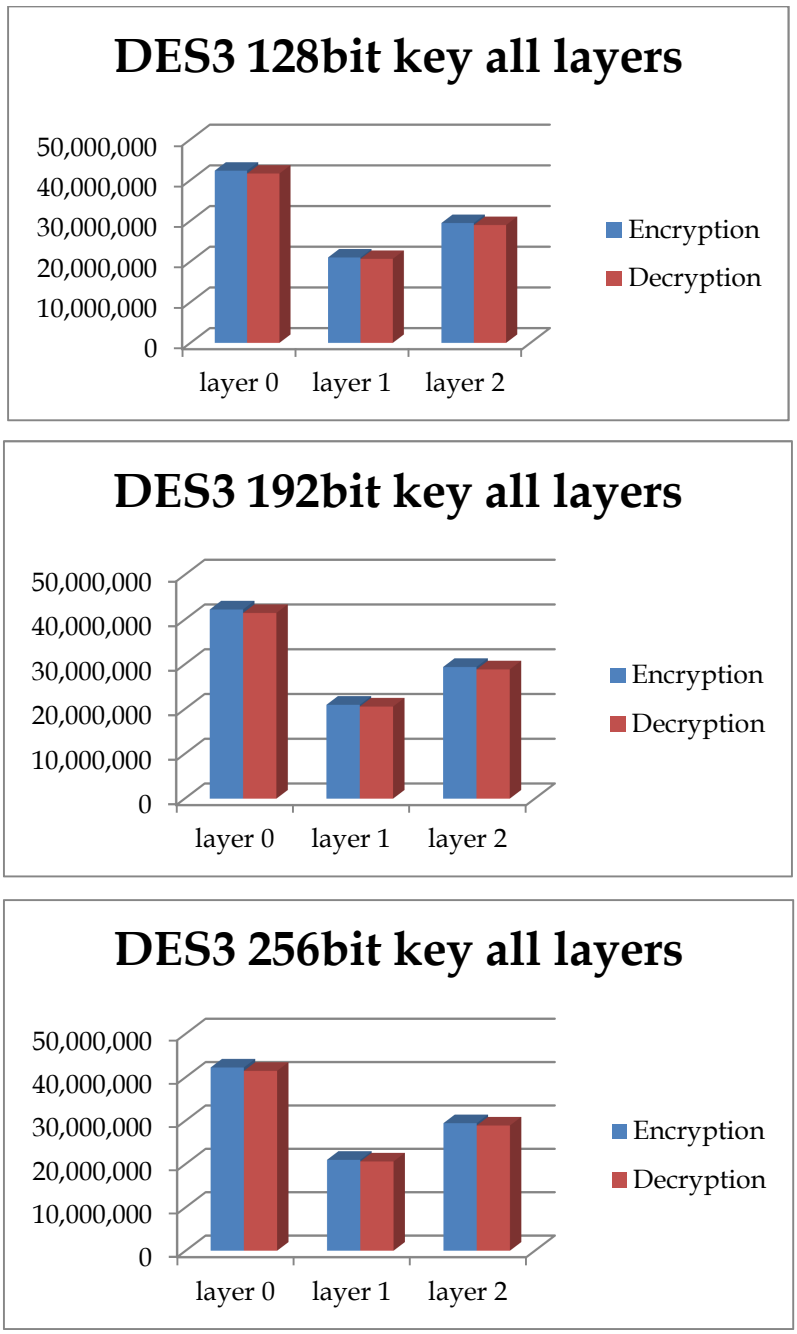




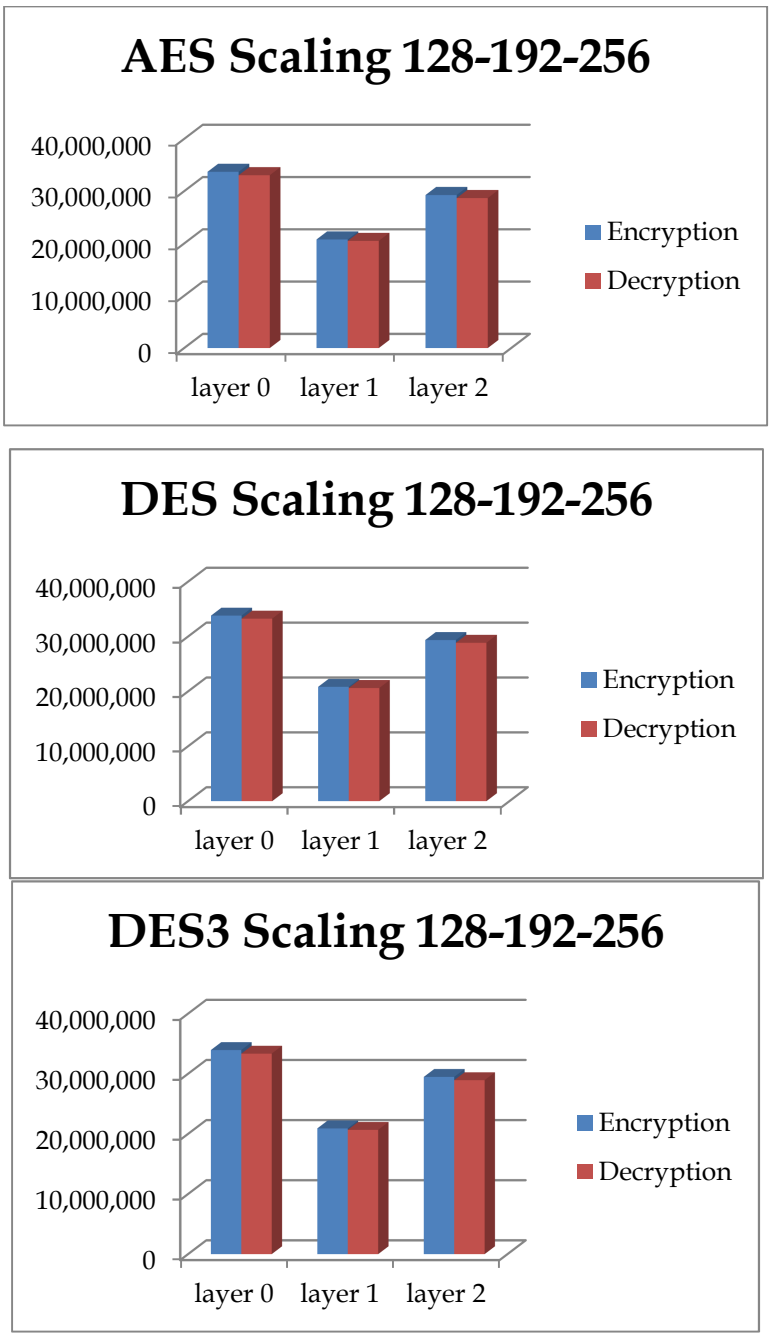
Πίνακας 4.7: Συνολικοί χρόνοι κρυπτογράφησης και αποκρυπτογράφησης ανά layer με την χρήση AES (μsec)



Πίνακας 4.8: Συνολικοί χρόνοι κρυπτογράφησης και αποκρυπτογράφησης ανά layer με την χρήση DES (μsec)



Πίνακας 4.9: Συνολικοί χρόνοι κρυπτογράφησης και αποκρυπτογράφησης ανά layer με την χρήση 3DES (μsec)



Πίνακας 4.10: Συνολικοί χρόνοι κρυπτογράφησης και αποκρυπτογράφησης ανά layer για κλιμακωτή χρήση κλειδιών ανά layer (μsec).

Κεφάλαιο 5

Συμπεράσματα

Σκοπός της μεταπτυχιακής διατριβής ήταν η βασική διερεύνηση ενός συστήματος CAS για την κωδικοποίηση ενός video streaming σε περιβάλλον IPTV. Βασικό σημείο είναι το μοντέλο για τον σημείο εγκατάστασης του όπως και ο τρόπος κρυπτογράφησης. Το προτεινόμενο σύστημα CAS για την εγκατάσταση του σε ένα server για IPTV είναι ένα σύστημα που εγκαθίσταται αμέσως μετά τον streamer με λειτουργία πολλαπλών IP καναλιών και layers. Η πρόσβαση σε περισσότερα IP κανάλια σημαίνει και ανώτερης- καλύτερης ποιότητας video (Layered multicast). Χρειάζεται όμως διερεύνηση αν η χρήση ενός τέτοιου συστήματος μπορεί να συνδυαστεί με την χρήση διαδεδομένων αλγορίθμων κρυπτογράφησης όπως οι AES-CBC, DES-CBC και 3DES-CBC.

Τα αποτελέσματα έδειξαν ότι κάτι τέτοιο είναι εφικτό αφού οι χρόνοι που προέκυψαν είναι αποδεκτοί για την συνολική καθυστέρηση που δημιουργείται. Ακόμα βλέποντας τους χρόνος κρυπτογραφίας ανά layer διαπιστώνουμε ότι το base layer χρειάζεται το μεγαλύτερο χρόνο κρυπτογράφησης και ακολουθεί το enhancement layer 2 που βελτιώνει αισθητά την ποιότητα του video στον τελικό χρήστη. Συγκρίνοντας τους χρόνους καθυστέρησης μεταξύ των αλγορίθμων διαπιστώνεται ότι ο AES είναι ο πιο γρήγορος για αυτή τη διαδικασία. Ανεβάζοντας το επίπεδο ασφαλείας με την χρήση μεγαλυτέρων bit κλειδιών δεν παρατηρήθηκε ιδιαίτερα μεγαλύτερη καθυστέρηση κάτι που δίνει την δυνατότητα για χρήση κλειδιού 256bit χωρίς ιδιαίτερα προβλήματα.

Βιβλιογραφία:

[01] Marie-Jos´e Montpetit MIT Research Lab for Electronics, Cambridge, MA, USA Email: mariejo@mit.edu Thomas Mirlacher IRIT, University of Toulouse, FR Email: thomas.mirlacher@irit.fr Michael Ketcham Keller, TX, USA Email: mgketcham@gmail.com **IPTV: An End to End Perspective** JOURNAL OF COMMUNICATIONS, VOL. 5, NO. 5, MAY 2010

[02] Shiguo Lian, Member IEEE, Zhongxuan Liu, Member, **Secure Media Content Distribution Based on the Improved Set-Top Box in IPTV** EEE Transactions on Consumer Electronics, Vol. 54, No. 2, MAY 2008

[03] Mark Jeffrey, Microsoft Sungjin Park and Kwangkee Lee, Samsung Glenn Adams, XFSIStuart Savage, Zetacast **Content Security for IPTV** IEEE Communications Magazine November 2008

[04] Daniel Díaz-Sánchez, Member, IEEE, Andrés Marín, Member, IEEE, Florina Almenárez, Member, IEEE, and Alberto Cortés **Sharing Conditional Access Modules through the Home Network for Pay TV Access** IEEE Transactions on Consumer Electronics, Vol. 55, No. 1, FEBRUARY 2009

[05] Seong Oun Hwanγ, **Content and Service Protection for IPTV** IEEE TRANSACTIONS ON BROADCASTING, VOL. 55, NO. 2, JUNE 2009

[06] Jinyoung Moon, Jungtae Kim, Jongyoul P ark, Euihyun Paik, and Kwangroh Park, **A Dynamic Conditi onal Access SystemBased on Cryptographic Software for the IPTV Set-top Box**, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea 978-1-4244-2559-4/09/$25.00 ©2009 IEEE

[07] Ali C. Begen, Tankut Akgul and Mark Baugher, **Watching Video over the Web Streaming Protocols**, Cisco

[08] Achie_ ving IPTV Service Portability Davide Proserpio, Fabio Sanvido, Daniel Diaz-Sanchez and Andrıs Marin, Ingenierva Telemαtica, Universidad Carlos III de Madrid, Avda. de la Universidad 30, 28911, Leganıs, Madrid 2011 **IPTV Meets Delegation** IEEE International Conference on Consumer Electronics (ICCE)

[09] Manhyun Chung, Younghoon Lee, Taeshik Shon, Jongsub Moon **A security model for IPTV with one-time password and Conditional Access System for smart mobile platform** Springer Science+Business Media, LLC 2011

[10] Axis Communications, **H.264 video compression standard New possibilities within video surveillance**

[11] Stephen Northcutt, Lenny Zeltser, Scott Winters, Karen Kent, Ronald W. Ritchery Second Edition, **Inside Network Perimeter Security**, Sans Institute March 2005

[12] James F. Kurose, Keith W. Ross **Computer Networking A top down approach** fifth edition Pearson

[13] Kwang-deok Seo, Jin-soo Kim, Soon-heung Jung, and Jeong-ju Yoo **A Practical RTP Packetization Scheme for SVC Video Transport over IP Networks** ETRI Journal, Volume 32, Number 2, April

[14] Touradj Ebrahimi and Frédéric Dufaux, **Information Processing for Video Surveillance** EMITALL Surveillance SA Montreux, Switzerland

Παράρτημα Α

Στοιχεία εξομοίωσης του CAS

Α.1 Κώδικας μέτρησης του χρόνου κρυπτογράφησης και αποκρυπτογράφησης με σταθερό bit κλειδιού ανά layer.

```
/* Ορισμός μεταβλητών*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <time.h>

struct timeval t0,t1;

struct packetsData{
	int packetid;
	int framenum;
	int layerid;
	int length;
};
```

```
int main(void)
{
 /*ορισμός μεταβλητών για αρχεία*/
  static const char filename[] = "deadline1.txt";
  static const char filenameout[]="256rc5again.txt";
  FILE *file = fopen(filename, "r");
  FILE *file1=fopen(filenameout, "w");
  float elapTicks;

 /*έλεγχος ορθότητας αρχείου*/
  if ( file != NULL )
  {
        char line[BUFSIZ];
     int j=0;
        int i=0;
        int bit=32;
     struct packetsData packets[14100];
        packets[0].packetid=0;
        packets[0].framenum=0;
        packets[0].layerid=0;
        packets[0].length=0;


  /*αναγνωση αρχείου και αποθήκευση σε πίνακα*/
   for (i=1;i<=14071;i++)
   {
        fgets(line, sizeof line, file);
        sscanf(line, "%d %d %d %d", &packets[i].packetid, &packets[i].framenum,
&packets[i].layerid, &packets[i].length);
   }
/*εφαρμογή της διαδικασίας CAS*/
   for (i=1;i<=14071;i++)
   {
        j=i-1;

     if ((packets[i].framenum != packets[j].framenum) && ((packets[i].framenum % 12)
== 0)) /*Αλλαγή κλειδιού ανα 12 frames – χρονομέτρηση διαδικασίας*/
        {
                gettimeofday(&t0, 0);
                randomfile(packets[i].length);
                keygeneration(bit);
                system ("openssl enc -e -rc5 -salt -in packet.txt -out packet.enc -k
key.txt");
                gettimeofday(&t1, 0);
          long elapsed1 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-t0.tv_usec;
                gettimeofday(&t0, 0);
                system ("openssl enc -d -rc5 -salt -in packet.enc -out packetout.txt -
k key.txt");
                gettimeofday(&t1, 0);
                long elapsed2 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-
t0.tv_usec;
```

```
                        fprintf(file1, "%d\t%d\t%d\t%d\t%ld\t%ld\n",
packets[i].packetid, packets[i].framenum, packets[i].layerid, packets[i].length, elapsed1,
elapsed2);
        }
        else
        {
                        randomfile(packets[i].length);
                        gettimeofday(&t0, 0);
                        system ("openssl enc -e -rc5 -salt -in packet.txt -out packet.enc -k
key.txt");
                        gettimeofday(&t1, 0);
                long elapsed1 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-t0.tv_usec;
                        gettimeofday(&t0, 0);
                        system ("openssl enc -d -rc5 -salt -in packet.enc -out packetout.txt -
k key.txt");
                        gettimeofday(&t1, 0);
                        long elapsed2 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-
t0.tv_usec;
                        fprintf(file1, "%d\t%d\t%d\t%d\t%ld\t%ld\n",
packets[i].packetid, packets[i].framenum, packets[i].layerid, packets[i].length, elapsed1,
elapsed2);
        }



  }
    system("pause");
  }
  else
  {
    perror(filename);
  }
fclose(file);
fclose(file1);
  return 0;
}



randomfile(int bytes) /* δημιουργία τυχαίου αρχείου προς κωδικοποίηση*/
{
        FILE *cfPtra;
        char
*A[]={"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V",
"W","X","Y","Z"};
        int rep=0;
        int i=0;
        if ( ( cfPtra = fopen ( "packet.txt", "w") )==NULL)
                printf ("File couldnot be opened. \n");
                else {
                for (rep=0; rep < bytes; rep++)
```

```
                {
                 i=rand() %26;
                 fprintf (cfPtra, "%s", A[i]);
                }
        fclose (cfPtra);
        }
return 0;
}
keygeneration(int bit) /*δημιουργία κλειδιού προς κωδικοποίηση*/
{
        FILE *cfPtrb;
        int A[02]={0,1};
        int rep=0;
        int i=0;
        if ( ( cfPtrb = fopen ( "key.txt", "w") )==NULL)
                printf ("File couldnot be opened. \n");
                else {
                for (rep=0; rep < bit; rep++)
                {
                 i=1+rand() %1;
                 fprintf (cfPtrb, "%d", A[i]);
                }
        fclose (cfPtrb);
        }
return 0;
}
```

A.2 Κώδικας μέτρησης του χρόνου κρυπτογράφησης και αποκρυπτογράφησης με κλιμάκωση ανά layer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <time.h>

struct timeval t0,t1;

struct packetsData{
        int packetid;
        int framenum;
        int layerid;
        int length;
};
int keygeneration128(void);
int keygeneration192(void);
```

```
int keygeneration256(void);

int main(void)
{


  static const char filename[] = "deadline1.txt";
  static const char filenameout[]="aes-2again.txt";
  FILE *file = fopen(filename, "r");
  FILE *file1=fopen(filenameout, "w");
  float elapTicks;


  if ( file != NULL )
  {
        char line[BUFSIZ];
        int i=0;
        int j=0;
        int bit=256;
     struct packetsData packets[14100];
        packets[0].packetid=0;
        packets[0].framenum=0;
        packets[0].layerid=0;
        packets[0].length=0;



   for (i=1;i<=14071;i++)
   {
        fgets(line, sizeof line, file);
        sscanf(line, "%d %d %d %d", &packets[i].packetid, &packets[i].framenum,
&packets[i].layerid, &packets[i].length);
   }
   for (i=1;i<=14071;i++)
   {
        j=i-1;

      if ((packets[i].framenum != packets[j].framenum) && ((packets[i].framenum % 12)
== 0))
        {



                        gettimeofday(&t0, 0);
                        randomfile(packets[i].length);
                        keygeneration128;
                        keygeneration256;
                        keygeneration192;

                        system ("openssl enc -e -aes-256-cbc -salt -in packet.txt -out
packet.enc -k key256.txt");
```

```
                gettimeofday(&t1, 0);
            long elapsed1 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-t0.tv_usec;
                gettimeofday(&t0, 0);
                system ("openssl enc -d -aes-256-cbc -salt -in packet.enc -out
packetout.txt -k key256.txt");
                gettimeofday(&t1, 0);
                long elapsed2 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-
t0.tv_usec;
                fprintf(file1, "%d\t%d\t%d\t%d\t%ld\t%ld\n",
packets[i].packetid, packets[i].framenum, packets[i].layerid, packets[i].length, elapsed1,
elapsed2);
        }
        else if ((packets[i].framenum == packets[j].framenum) && packets[i].layerid==0)
        {
                bit=256;
                randomfile(packets[i].length);
              gettimeofday(&t0, 0);
                system ("openssl enc -e -aes-256-cbc -salt -in packet.txt -out
packet.enc -k key256.txt");
                gettimeofday(&t1, 0);
            long elapsed1 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-t0.tv_usec;
                gettimeofday(&t0, 0);
                system ("openssl enc -d -aes-256-cbc -salt -in packet.enc -out
packetout.txt -k key256.txt");
                gettimeofday(&t1, 0);
                long elapsed2 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-
t0.tv_usec;
                fprintf(file1, "%d\t%d\t%d\t%d\t%ld\t%ld\n",
packets[i].packetid, packets[i].framenum, packets[i].layerid, packets[i].length, elapsed1,
elapsed2);
        }
        else if ((packets[i].framenum == packets[j].framenum) && packets[i].layerid==1)
        {
                bit=192;
                randomfile(packets[i].length);
             gettimeofday(&t0, 0);
                system ("openssl enc -e -aes-192-cbc -salt -in packet.txt -out
packet.enc -k key192.txt");
                gettimeofday(&t1, 0);
            long elapsed1 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-t0.tv_usec;
                gettimeofday(&t0, 0);
                system ("openssl enc -d -aes-192-cbc -salt -in packet.enc -out
packetout.txt -k key192.txt");
                gettimeofday(&t1, 0);
                long elapsed2 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-
t0.tv_usec;
                fprintf(file1, "%d\t%d\t%d\t%d\t%ld\t%ld\n",
packets[i].packetid, packets[i].framenum, packets[i].layerid, packets[i].length, elapsed1,
elapsed2);
        }
        else if ((packets[i].framenum == packets[j].framenum) && packets[i].layerid==2)
```

```
        {
                        bit=128;
                        randomfile(packets[i].length);
                    gettimeofday(&t0, 0);
                        system ("openssl enc -e -aes-128-cbc -salt -in packet.txt -out
packet.enc -k key128.txt");
                        gettimeofday(&t1, 0);
                long elapsed1 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-t0.tv_usec;
                        gettimeofday(&t0, 0);
                        system ("openssl enc -d -aes-128-cbc -salt -in packet.enc -out
packetout.txt -k key128.txt");
                        gettimeofday(&t1, 0);
                        long elapsed2 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-
t0.tv_usec;
                        fprintf(file1, "%d\t%d\t%d\t%d\t%ld\t%ld\n",
packets[i].packetid, packets[i].framenum, packets[i].layerid, packets[i].length, elapsed1,
elapsed2);
        }



  }
    system("pause");
  }
  else
  {
    perror(filename);
  }
fclose(file);
fclose(file1);
  return 0;
}



randomfile(int bytes)
{
        FILE *cfPtra;
        char
*A[]={"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V",
"W","X","Y","Z"};
        int rep=0;
        int i=0;
        if ( ( cfPtra = fopen ( "packet.txt", "w") )==NULL)
                printf ("File couldnot be opened. \n");
                else {
                for (rep=0; rep < bytes; rep++)
                {
                 i=rand() %26;
                 fprintf (cfPtra, "%s", A[i]);
```

```
                }
        fclose (cfPtra);
        }
return 0;
}

keygeneration128()
{
        int bit=16;
        FILE *cfPtrb;
        int A[02]={0,1};
        int rep=0;
        int i=0;
        if ( ( cfPtrb = fopen ( "key128.txt", "w") )==NULL)
                printf ("File couldnot be opened. \n");
                else {
                for (rep=0; rep < bit; rep++)
                {
                 i=rand() %1;
                 fprintf (cfPtrb, "%d", A[i]);
                }
        fclose (cfPtrb);
        }
return 0;
}

keygeneration256()
{
        int bit=32;
        FILE *cfPtrb;
        int A[02]={0,1};
        int rep=0;
        int i=0;
        if ( ( cfPtrb = fopen ( "key256.txt", "w") )==NULL)
                printf ("File couldnot be opened. \n");
                else {
                for (rep=0; rep < bit; rep++)
                {
                 i=rand() %1;
                 fprintf (cfPtrb, "%d", A[i]);
                }
        fclose (cfPtrb);
        }
return 0;
}

keygeneration192()
{
        int bit=24;
        FILE *cfPtrb;
        int A[02]={0,1};
```

```
        int rep=0;
        int i=0;
        if ( ( cfPtrb = fopen ( "key192.txt", "w") )==NULL)
                printf ("File couldnot be opened. \n");
                else {
                for (rep=0; rep < bit; rep++)
                {
                 i=rand() %1;
                 fprintf (cfPtrb, "%d", A[i]);
                }
        fclose (cfPtrb);
        }
return 0;
}
```

A.3 Κώδικας μέτρησης του χρόνου κρυπτογράφησης και αποκρυπτογράφησης μόνο στο layer 0.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>
#include <time.h>

struct timeval t0,t1;

struct packetsData{
        int packetid;
        int framenum;
        int layerid;
        int length;
};


int main(void)
{


  static const char filename[] = "deadline1.txt";
  static const char filenameout[]="aes256-3again.txt";
  FILE *file = fopen(filename, "r");
  FILE *file1=fopen(filenameout, "w");
  float elapTicks;


  if ( file != NULL )
  {
        char line[BUFSIZ];
```

```
        int i=0;
        int j=0;
        int bit=32;
     struct packetsData packets[14100];
        packets[0].packetid=0;
        packets[0].framenum=0;
        packets[0].layerid=0;
        packets[0].length=0;



  for (i=1;i<=14071;i++)
  {
        fgets(line, sizeof line, file);
        sscanf(line, "%d %d %d %d", &packets[i].packetid, &packets[i].framenum,
&packets[i].layerid, &packets[i].length);
  }
  for (i=1;i<=14071;i++)
  {
        j=i-1;

    if ((packets[i].framenum != packets[j].framenum) && ((packets[i].framenum % 12)
== 0))
        {
                        gettimeofday(&t0, 0);
                        randomfile(packets[i].length);
                        keygeneration(bit);
                        system ("openssl enc -e -aes-256-cbc -salt -in packet.txt -out
packet.enc -k key256.txt");
                        gettimeofday(&t1, 0);
                long elapsed1 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-t0.tv_usec;
                        gettimeofday(&t0, 0);
                        system ("openssl enc -d -aes-256-cbc -salt -in packet.enc -out
packetout.txt -k key256.txt");
                        gettimeofday(&t1, 0);
                        long elapsed2 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-
t0.tv_usec;
                        fprintf(file1, "%d\t%d\t%d\t%d\t%ld\t%ld\n",
packets[i].packetid, packets[i].framenum, packets[i].layerid, packets[i].length, elapsed1,
elapsed2);
        }

        else if ((packets[i].framenum == packets[j].framenum) && packets[i].layerid==0)
        {

                        randomfile(packets[i].length);
                    gettimeofday(&t0, 0);
                        system ("openssl enc -e -aes-256-cbc -salt -in packet.txt -out
packet.enc -k key256.txt");
                        gettimeofday(&t1, 0);
                long elapsed1 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-t0.tv_usec;
```

```
                        gettimeofday(&t0, 0);
                        system ("openssl enc -d -aes-256-cbc -salt -in packet.enc -out
packetout.txt -k key256.txt");
                        gettimeofday(&t1, 0);
                        long elapsed2 = (t1.tv_sec-t0.tv_sec)*1000000 + t1.tv_usec-
t0.tv_usec;
                        fprintf(file1, "%d\t%d\t%d\t%d\t%ld\t%ld\n",
packets[i].packetid, packets[i].framenum, packets[i].layerid, packets[i].length, elapsed1,
elapsed2);
        }




  }
    system("pause");
  }
  else
  {
    perror(filename);
  }
fclose(file);
fclose(file1);
  return 0;
}



randomfile(int bytes)
{
        FILE *cfPtra;
        char
*A[]={"A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q","R","S","T","U","V",
"W","X","Y","Z"};
        int rep=0;
        int i=0;
        if ( ( cfPtra = fopen ( "packet.txt", "w") )==NULL)
                printf ("File couldnot be opened. \n");
                else {
                for (rep=0; rep < bytes; rep++)
                {
                 i=rand() %26;
                 fprintf (cfPtra, "%s", A[i]);
                }
        fclose (cfPtra);
        }
return 0;
}
keygeneration(int bit)
{
        FILE *cfPtrb;
```

```
        int A[02]={0,1};
        int rep=0;
        int i=0;
        if ( ( cfPtrb = fopen ( "key256.txt", "w") )==NULL)
                printf ("File couldnot be opened. \n");
                else {
                for (rep=0; rep < bit; rep++)
                {
                 i=rand() %1;
                 fprintf (cfPtrb, "%d", A[i]);
                }
        fclose (cfPtrb);
        }
return 0;
}
```

A.4 Επεξεργασία αποτελεσμάτων, άθροιση χρόνων κωδικοποίησης ανά frame.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct packetsData{
        int packetid;
        int framenum;
        int layerid;
        int length;
        int time_enc;
        int time_dec;
};


int main(void)
{


  static const char filename[] = "aes256-3again.txt";
  static const char filenameout[]="aes256-3again-res.txt";
  FILE *file = fopen(filename, "r");
  FILE *file1= fopen(filenameout, "w");
     char line[BUFSIZ];
        int i=1;

        int j=0;
        int Sumtime_enc=0;
        int Sumtime_dec=0;
```

```
    struct packetsData packets[14100];
        packets[0].packetid=0;
        packets[0].framenum=0;
        packets[0].layerid=0;
        packets[0].length=0;
        packets[0].time_enc=0;
        packets[0].time_dec=0;

  if (file == NULL) {
    fprintf(stderr, "Could not open file\n");
    exit(8);
  }
  for (i=1;i<=14071;i++)
  {
        fgets(line, sizeof line, file);
        sscanf(line, "%d %d %d %d %d %d", &packets[i].packetid,
&packets[i].framenum, &packets[i].layerid, &packets[i].length, &packets[i].time_enc,
&packets[i].time_dec);
  }
  for (i=1;i<=14071;i++)
  {
        j=i-1;


    if (packets[i].framenum==packets[j].framenum)
        {
                printf("found\n");
                Sumtime_enc=Sumtime_enc+packets[i].time_enc;
                Sumtime_dec=Sumtime_dec+packets[i].time_dec;
        }
        else
        {
                fprintf(file1, "%d %d %d\n", packets[j].framenum, Sumtime_enc,
Sumtime_dec);
                Sumtime_enc=packets[i].time_enc;
                Sumtime_dec=packets[i].time_dec;
}

  }
  fclose(file);
  fclose(file1);


  return 0;
}
```

A.5 Επεξεργασία αποτελεσμάτων, άθροιση χρόνων κωδικοποίησης ανά layer.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>



struct packetsData{
        int packetid;
        int framenum;
        int layerid;
        int length;
        int time;
        int timed;
};


int main(void)
{


  static const char filename[] = "des3256-3again.txt";
  static const char filenameout[]="des3256-3againprolayer.txt";
  FILE *file = fopen(filename, "r");
  FILE *file1= fopen(filenameout, "w");
    char line[BUFSIZ];
        int i=1;

        int j=0;
        int Sumtime1=0;
        int Sumtime2=0;
        int Sumtime3=0;

    int Sumtime1d=0;
        int Sumtime2d=0;
        int Sumtime3d=0;

    struct packetsData packets[14100];
        packets[0].packetid=0;
        packets[0].framenum=0;
        packets[0].layerid=0;
        packets[0].length=0;
        packets[0].time=0;
    packets[0].timed=0;

  if (file == NULL) {
```

```
      fprintf(stderr, "Could not open file\n");
      exit(8);
   }
   for (i=1;i<=14071;i++)
   {
         fgets(line, sizeof line, file);
         sscanf(line, "%d %d %d %d %d %d", &packets[i].packetid,
&packets[i].framenum, &packets[i].layerid, &packets[i].length, &packets[i].time,
&packets[i].timed);
   }
   for (i=1;i<=14071;i++)
   {
         j=i-1;


      if (packets[i].layerid==0)
         {
               Sumtime1=Sumtime1+packets[i].time;
               Sumtime1d=Sumtime1d+packets[i].timed;
         }
         else if (packets[i].layerid==1)
         {

               Sumtime2=Sumtime2+packets[i].time;
               Sumtime2d=Sumtime2d+packets[i].timed;
         }
         else if (packets[i].layerid==2)
         {
               Sumtime3=Sumtime3+packets[i].time;
               Sumtime3d=Sumtime3d+packets[i].timed;
         }
   }
         fprintf(file1, " layerid 0 %d %d\n layerid 1 %d %d\n layerid 2 %d %d\n",
Sumtime1, Sumtime1d, Sumtime2, Sumtime2d, Sumtime3, Sumtime3d);


   fclose(file);
   fclose(file1);


   return 0;
}
```

Α.6 Χαρακτηριστικά Η/Υ που χρησιμοποιήθηκε για τις μετρήσεις

Operating System – OS type Version	Ubuntu 12.10 – Linux 4.7 (i686-linux-gnu)
CPU- Model Name- Frequency – L2 Cache	Celeron Dual Core T3100 @1.9Ghz 1024KB
Memory	1978MB
Storage	FUJITSU MJA2250B - DVD A DS8A4S
Graphics Card	Intel Corporation Mobile 4 series Chipset integrated graphics controller
Sound Card	Intel Corporation 82801I(ICH9 Family)