ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«SECURITY TOKEN DEVICE»

Του φοιτητή:
Τσιούπρου Δημήτριου
Αρ. Μητρώου: 515302

Επιβλέπων
Ονοματεπώνυμο: Γιακουμής
Άγγελος
Βαθμίδα: Αναπληρωτής
Καθηγητής

Ημερομηνία 10/09/2025

Τίτλος Δ.Ε.: Security Token Device
Κωδικός Δ.Ε.: 25273
Ονοματεπώνυμο φοιτητή: Δημήτριος Τσιούπρος
Ονοματεπώνυμο εισηγητή: Γιακουμής Άγγελος
Ημερομηνία ανάληψης Δ.Ε.: 20/06/2025
Ημερομηνία περάτωσης Δ.Ε.: 10/09/2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Πρόλογος

Καθώς η εποχή που διανύουμε είναι άμεσα συνδεδεμένη με την ψηφιακή πληροφορία και οι απειλές στον κυβερνοχώρο αυξάνονται συνεχώς, η ανάγκη πιο ασφαλούς και πιο αξιόπιστης ταυτοποίησης των χρηστών αποτελεί πολύ σημαντικό παράγοντα για την προστασία των ευαίσθητων δεδομένων και συστημάτων. Οι συσκευές Security Token αποτελούν μια εξαιρετική λύση απέναντι στις ψηφιακές απειλές και μπορούν να προσφέρουν υψηλή ασφάλεια στην ταυτοποίηση χρηστών, αφού προσφέρουν κάποιον επιπλέον παράγοντα ταυτοποίησης ενισχύοντας την είσοδο με έναν απλό κωδικό πρόσβασης.

Στην παρούσα εργασία γίνεται μελέτη των Security Tokens και αναλύονται τόσο η λειτουργία και οι τεχνολογίες που υποστηρίζουν όσο και οι απειλές που καλούνται να αντιμετωπίσουν κατά τη χρήση τους. Σε συνδυασμό με τη θεωρητική ανάλυση, σχεδιάζεται και υλοποιείται ένα συγκεκριμένο είδος Security Token με σκοπό να παρουσιαστούν τόσο οι δυνατότητες που μπορεί να προσφέρει όσο και οι περιορισμοί που το συνοδεύουν.

Στόχος της παρούσας εργασίας είναι η κατανόηση του κρίσιμου ρόλου που παίζουν τα Security Tokens στον ψηφιακό κόσμο και στην ασφάλεια της ψηφιακής πληροφορίας ενώ ταυτόχρονα, παρουσιάζεται ένας τρόπος σχεδίασης, ανάπτυξης και αξιολόγησης μιας τέτοιας συσκευής.

Περίληψη

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάλυση και υλοποίηση συσκευών Security Token, συσκευές οι οποίες αποτελούν σημαντικό εργαλείο έναντι των ψηφιακών απειλών που συναντώνται κατά την ταυτοποίηση των χρηστών στα διάφορα ψηφιακά συστήματα. Πιο συγκεκριμένα, μετά την ολοκλήρωση της θεωρητικής μελέτη τους, σχεδιάζεται και κατασκευάζεται ένα Challenge-Response Security Token με τη βοήθεια του μικροελεγκτή PIC18F1320. Το Token μπορεί να χρησιμοποιηθεί ως επιπλέον παράγοντας ταυτοποίησης χρήστη προσφέροντας υψηλό επίπεδο ασφάλειας σε σχέση με άλλα είδη Token. Το περιεχόμενο της εργασίας κατανέμεται ως εξής:

Στο 1ο Κεφάλαιο παρουσιάζονται οι έννοιες της ταυτοποίησης και της αυθεντικοποίησης και γίνεται αναφορά στις διάφορες μορφές ταυτοποίησης που συναντώνται στα ψηφιακά συστήματα.

Στο 2ο Κεφάλαιο αναλύονται οι διάφοροι τύποι Security Token ενώ γίνεται ξεχωριστή αναφορά στα Challenge-Response Security Tokens.

Στο 3ο Κεφάλαιο γίνεται αναφορά σε θέματα ασφαλείας όπου επεξηγούνται διαφορετικοί τύποι κυβερνοεπιθέσεων καθώς και οι τρόποι που αντιμετωπίζονται με τη χρήση των Tokens.

Στο 4ο Κεφαλαίο μελετάται ο σχεδιασμός της συσκευής τόσο σε επίπεδο hardware όσο και σε software.

Στο 5ο Κεφάλαιο παρουσιάζεται ο τρόπος υλοποίησης του Token καθώς και η αξιολόγηση του αποτελέσματος.

Τέλος, αναφέρονται τα συμπεράσματα της εργασίας και σημειώνονται προτάσεις βελτίωσης της κατασκευής.

SECURITY TOKEN DEVICE

Tsioupros Dimitrios

Abstract

This thesis deals with the analysis and implementation of Security Token devices, which are an important tool against digital threats encountered during user authentication in various digital systems. More specifically, after completing the theoretical study, a Challenge-Response Security Token is designed and constructed with the help of the PIC18F1320 microcontroller. The Token can be used as an additional user authentication factor, offering a high-level security compared to other Token types. The content of the thesis is distributed as follows:

Chapter 1 presents the concepts of identification and authentication and refers to the various forms of identification found in digital systems.

Chapter 2 analyzes the various types of security tokens, with a separate reference to challenge-response security tokens.

Chapter 3 refers to security issues, explaining different types of cyber-attacks and how they can be addressed using tokens.

Chapter 4 examines the design of the device in terms of both hardware and software.

Chapter 5 presents the implementation of the token and the evaluation of the results.

Finally, the conclusions of the thesis are presented and suggestions for improving the design are noted.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1ο: Ταυτοποίηση και αυθεντικοποίηση

1.1 Έλεγχος πρόσβασης

Για να αντιληφθεί κάποιος την αναγκαιότητα της χρήσης των security tokens θα πρέπει πρώτα να γίνει κατανοητό τι είναι ο έλεγχος πρόσβασης και πώς αυτός επιτελείται.

Ο έλεγχος πρόσβασης είναι ένα βασικό στοιχείο στον κλάδο της ασφάλειας συστημάτων, καθώς μέσω αυτού επιτρέπεται ή απορρίπτεται η πρόσβαση ενός χρήστη σε κάποιο σύστημα. Πιο συγκεκριμένα μέσου του ελέγχου πρόσβασης ορίζεται ποιος μπορεί να έχει πρόσβαση σε οποιοδήποτε δεδομένο ενός ψηφιακού χώρου και κάτω από ποιες συνθήκες. Με τον τρόπο μη εξουσιοδοτημένοι χρήστες δεν έχουν πρόσβαση σε απόρρητα για τους ίδιους δεδομένα και μειώνεται ο κίνδυνος υποκλοπής.

Με απλά λόγια, ο έλεγχος πρόσβασης «αναγνωρίζει» έναν χρήστη από τα διαπιστευτήριά του (κωδικοί πρόσβασης, pin, βιομετρικές σαρώσεις κ.α.) και έπειτα δίνει εξουσιοδότηση για το κατάλληλο επίπεδο πρόσβασης. Ο χρήστης μπορεί να συνεχίσει όπως επιθυμεί, λαμβάνοντας υπόψη τα δικαιώματα πρόσβασης τα οποία έχει.

Υπάρχουν 4 βασικά είδη ελέγχου πρόσβασης τα οποία είναι:

- Διακριτικός έλεγχος πρόσβασης (DAC)

Το είδος αυτό βασίζεται σε λίστες ελέγχου πρόσβασης (ACLs) και ο ιδιοκτήτης του δεδομένου ή του πόρου καθορίζει τόσο το ποιος έχει πρόσβαση αλλά και ποια δικαιώματα πρόσβασης θα έχει.

Αποτελεί ένα ευέλικτο μοντέλο παρ' όλα αυτά είναι αρκετά επιρρεπές σε λάθη σχετικά με την ασφάλεια.

- Υποχρεωτικός έλεγχος πρόσβασης (MAC)

Το είδος αυτό σχετίζεται με επίπεδα διαβάθμισης του χρήστη αλλά και του πόρου. Η πρόσβαση είναι αποτέλεσμα της πολιτικής ασφάλειας του συστήματος και όχι κάτι που καθορίζει ο χρήστης.

Πρόκειται για ένα μοντέλο το οποίο δεν είναι αρκετά ευέλικτο, παρουσιάζει όμως πολύ υψηλή ασφάλεια και γι' αυτό χρησιμοποιείται σε κυβερνητικά και στρατιωτικά συστήματα.

- Έλεγχος πρόσβασης βάσει ρόλων (RBAC)

Σε αυτό το είδος σε κάθε χρήστη ανατίθεται κάποιος «ρόλος». Ο ρόλος αυτός έχει συγκεκριμένα δικαιώματα και είναι αυτός ο οποίος θα κρίνει το επίπεδο πρόσβασης που θα έχει ο ίδιος ο χρήστης.

Το μοντέλο αυτό είναι αρκετά εύκολο στη διαχείρισή του όμως η παράλειψη ενημέρωσης και ανανέωσης των ρόλων μπορεί να οδηγήσει σε υψηλή πρόσβαση από μη κατάλληλα εξουσιοδοτημένους χρήστες.

- Έλεγχος πρόσβασης βάσει χαρακτηριστικών (ABAC)

Σε αυτό το είδος η πρόσβαση επηρεάζεται από περισσότερα από ένα χαρακτηριστικά όπως είναι ο ίδιος ο χρήστης, η ενέργεια που συμβαίνει και το περιβάλλον στο οποίο βρίσκεται. Δηλαδή θα πρέπει να ελεγχθεί ένας αριθμός παραγόντων ώστε να κριθεί η ολοκλήρωση της πρόσβασης.

Πίνακας 1.1: Μοντέλα ελέγχου πρόσβασης

Μοντέλο	**Βάση μοντέλου**	**Λήψη απόφασης**
DAC	Χρήστη (ιδιοκτήτη)	Ο κάτοχος του πόρου
MAC	Πολιτική ασφαλείας	Το σύστημα/ο διαχειριστής
RBAC	Ρόλους	Ο διαχειριστής ρόλων
ABAC	Χαρακτηριστικά	Πολιτικές και κανόνες

Αν και η υλοποίηση και η διαχείρισή του είναι αρκετά περίπλοκες, αποτελεί μοντέλο με μεγάλη ακρίβεια και ευελιξία.

Δύο επίσης πολύ σημαντικές έννοιες που θα πρέπει να παρουσιαστούν είναι η ταυτοποίηση και η αυθεντικοποίηση. Αποτελούν βασικά στοιχεία της ασφάλειας των πληροφοριών αλλά και της διαχείρισης προσβασιμότητας στα διάφορα συστήματα.

Η ταυτοποίηση σχετίζεται με την ταυτότητα ενός χρήστη σε ένα σύστημα. Αποτελεί δηλαδή τη διαδικασία μέσω της οποίας το σύστημα προσπαθεί να αναγνωρίσει ποιος είναι αυτός που θέλει να αποκτήσει πρόσβαση. Ένα παράδειγμα εργαλείου ταυτοποίησης είναι το όνομα χρήστη (username).

Η αυθεντικοποίηση από την άλλη αποτελεί τη διαδικασία επαλήθευσης της ταυτοποίησης., εάν δηλαδή ο εκάστοτε χρήστης είναι πραγματικά αυτός που ισχυρίζεται ότι είναι. Παραδείγματα εργαλείων αυθεντικοποίησης αποτελούν τα pin και τα βιομετρικά στοιχεία όπως το δακτυλικό αποτύπωμα.

Είναι σημαντικό να αναφερθεί πως η ταυτοποίηση είναι απαραίτητη αλλά όχι αρκετή. Η διαδικασία της αυθεντικοποίησης είναι αυτή που θα εγγυηθεί για την επαλήθευση της ταυτότητας κάποιου χρήστη και που θα μπορέσει με ασφάλεια να δώσει την κατάλληλη πρόσβαση. Και οι δύο όμως αποτελούν σημαντικά στοιχεία στους ψηφιακούς χώρους καθώς αποτρέπουν τους μη εξουσιοδοτημένους χρήστες να έχουν πρόσβαση σε πόρους και πληροφορίες οι οποίες δεν σχετίζονται με τους ίδιους, επιτυγχάνεται λοιπόν σωστά ο έλεγχος πρόσβασης.

1.2 Μορφές ταυτοποίησης

Η ταυτοποίηση μπορεί να διακριθεί σε 3 βασικές μορφές, οι οποίες είναι:

- Αυτό που ξέρεις
- Αυτό που έχεις
- Αυτό που είσαι

Νεότερες μορφές ταυτοποίησης είναι:

- Αυτό που κάνεις
- Εκεί που βρίσκεσαι

Αυτό που ξέρεις (something you know)

Όπως είναι κατανοητό πρόκειται για πληροφορίες που μόνο ο χρήστης μπορεί να γνωρίζει. Τέτοιες πληροφορίες είναι οι κωδικοί πρόσβασης, τα PIN καθώς και απαντήσεις σε μυστικές ερωτήσεις.

Αυτή η μορφή ταυτοποίησης μπορεί να υλοποιηθεί πολύ εύκολα όμως στην πράξη δεν είναι η πλέον κατάλληλη καθώς είναι αρκετά εύκολο και σύνηθες, να ξεχάσει ο χρήστης τις πληροφορίες αυτές ή οι πληροφορίες να υποκλαπούν.

Αυτό που έχεις (something you have)

Στην περίπτωση αυτή η ταυτοποίηση σχετίζεται με κάποιο αντικείμενο που βρίσκεται στην κατοχή του χρήστη. Αυτό μπορεί να είναι κάποιο κινητό τηλέφωνο στο οποίο δέχεται OTP ή push notifications, μια κάρτα ασφαλείας ή κάποιο USB Token.

Πρόκειται για πιο ασφαλή τρόπο ταυτοποίησης όμως ο κίνδυνος κλοπής ή απώλειας είναι αυξημένος.

Αυτό που είσαι (something you are)

Όπως γίνεται αντιληπτό, πρόκειται για τα βιομετρικά χαρακτηριστικά ενός χρήστη, δηλαδή το δακτυλικό του αποτύπωμα, η σάρωση του προσώπου του ή του ματιού του αλλά και τη αναγνώριση φωνής.

Αν και είναι αρκετά διαδεδομένα στις μέρες μας, κυρίως στα κινητά τηλέφωνα, η υλοποίηση αυτών των εργαλείων ταυτοποίησης εμφανίζει πολύ μεγάλο κόστος ενώ ταυτόχρονα δημιουργούνται θέματα ιδιωτικότητας. Συγκριτικά με τις προηγούμενες μορφές είναι η πλέον ασφαλέστερη και η λιγότερο εύκολο να παραβιαστεί.

Σχετικά με τις νεότερες μορφές, χρησιμοποιούνται συνήθως ως δευτερεύοντα στοιχεία ασφαλείας ή συνδυαστικά με κάποιο άλλο μέσο ταυτοποίησης.

Αυτό που κάνεις

Πρόκειται για μορφές όπου ελέγχεται η συμπεριφορά του χρήστη όπως για παράδειγμα ο τρόπος που πληκτρολογεί ή ο τρόπος που χειρίζεται τις συσκευές (π.χ. ποντίκι ή κινητό).

Εκεί που βρίσκεσαι

Σχετίζεται με την τοποθεσία του χρήστη. Του δίνεται δηλαδή πρόσβαση ανάλογα με τη γεωγραφική του θέση ή την IP την οποία χρησιμοποιεί. Η συγκεκριμένη μορφή ταυτοποίησης εφαρμόζεται κυρίως σε επιχειρήσεις ή σε στρατιωτικές υπηρεσίες.

Καθώς όλο και περισσότερες συσκευές τις οποίες χρησιμοποιούμε καθημερινά είναι συσκευές ευάλωτες σε μη εξουσιοδοτημένη πρόσβαση, είναι επιτακτική ανάγκη η ασφάλεια πρόσβασης να είναι υψηλότερη. Για το λόγο αυτό συνηθίζεται να χρησιμοποιούνται μορφές πολυεπίπεδης ταυτοποίησης, δηλαδή να γίνεται συνδυασμός των παραπάνω μορφών ταυτοποίησης.

1.3 Factor Authentications

Ένας άλλος επομένως διαχωρισμός που μπορούμε να κάνουμε για τις μορφές ταυτοποίησης είναι ο εξής:

- Single-Factor Authentication
- Two-Factor Authentication
- Multi-Factor Authentication

Single-Factor Authentication

Αυτή η μορφή ταυτοποίησης αποτελεί την απλούστερη μορφή και περιλαμβάνει όλες τις προαναφερθείσες περιπτώσεις που παρουσιάστηκαν. Σε αυτή την μορφή η επιβεβαίωση του χρήστη και κατά συνέπεια το δικαίωμα πρόσβασης καθορίζεται από έναν μόνο παράγοντα. Όπως έχει ήδη αναφερθεί πρόκειται για ταυτοποίηση που υλοποιείται αρκετά εύκολα και ταυτόχρονα η εξουσιοδότηση στους χρήστες γίνεται εύκολα και γρήγορα. Ταυτόχρονα όμως η παραβίασή του καθίσταται εξίσου εύκολη κι έτσι δεν μπορεί να χρησιμοποιηθεί σε συστήματα που απαιτούν υψηλή ασφάλεια.

Σχήμα 1.1: Single Factor Authentication-1FA

Two-Factor Authentication

Το πρόβλημα της όχι και τόσο υψηλής ασφάλειας που παρουσιάζεται στην Single-Factor ταυτοποίηση λύνεται εύκολα με την Two-Factor.

Η μορφή αυτή απαιτεί το συνδυασμό δύο διαφορετικών παραγόντων οι οποίοι προέρχονται από διαφορετικές κατηγορίες ταυτοποίησης, για παράδειγμα αυτό που ξέρεις και αυτό που είσαι.

Με αυτό τον τρόπο αποτρέπονται πολλές μορφές επιθέσεων, παράλληλα όμως υπάρχει ο κίνδυνος απώλειας κάποιας συσκευής κάτι το οποίο θα μπλοκάρει πλήρως την πρόσβαση. Πρόκειται λοιπόν για μια μορφή σίγουρα ασφαλέστερη από την Single-Factor οπωσδήποτε όμως είναι πιο αργή και πιο περίπλοκη η υλοποίησή της.

Σχήμα 1.2: Two Factor Authentication-2FA

Multi-Factor Authentication

Σε αυτή τη μορφή χρησιμοποιούνται τουλάχιστον 2 παράγοντες από 2 διαφορετικές κατηγορίες, επομένως είναι κατανοητό πως η Two-Factor Authentication αποτελεί τμήμα της Multi-Factor.

Το πλεονέκτημα που εμφανίζει έναντι των 2 προηγούμενων μορφών είναι ότι η ασφάλεια που παρέχεται είναι πολύ υψηλότερη και γι' αυτό το λόγο χρησιμοποιείται ευρέως σε συστήματα όπως τα τραπεζικά αλλά και τα κυβερνητικά.

Από την άλλη, η χρήση πολλών και διαφορετικών παραγόντων μπορεί να καταστήσει τη διαδικασία αρκετά δύσκολη για το χρήστη και ταυτόχρονα το κόστος για την αρκετά περίπλοκη αυτή μορφή είναι αυξημένο.

Σχήμα 1.3: Multi Factor Authentication-MFA

Κεφάλαιο 2ο: Παρουσίαση των security token συσκευών

2.1 Εισαγωγή

Τα security tokens είναι συσκευές/μέσα τα οποία προσφέρουν έναν επιπλέον παράγοντα ταυτοποίησης (αποτελούν δηλαδή μέρος ενός 2FA ή MFA). Μέσω της δημιουργίας δυναμικών διαπιστευτηρίων μειώνουν την πιθανότητα κάποιος μη εξουσιοδοτημένος χρήστης να αποκτήσει πρόσβαση.

2.2 Τύποι security tokens

2.2.1 Hardware tokens

Τα Hardware tokens είναι φυσικές συσκευές από τις οποίες παράγονται μοναδικοί κωδικοί.

- TOTP (Time-Based One-Time Password Tokens)

Τα TOTP παράγουν έναν κωδικό ο οποίος ανανεώνεται ανά συγκεκριμένα χρονικά διαστήματα μερικών δευτερολέπτων ενώ ταυτόχρονα η συσκευή token βρίσκεται σε συγχρονισμό με το ρολόι του server. Για να μπορέσει ένας χρήστης να το χρησιμοποιήσει δεν απαιτείται σύνδεση στο διαδίκτυο. Πρόκειται για αρκετά αξιόπιστες συσκευές οι οποίες δεν μπορούν να παραβιαστούν εύκολα. Παρ' όλα αυτά θα πρέπει να υπάρχει συνεχώς συγχρονισμός των ρολογιών του token και του server ώστε να μην υπάρξει αποτυχία αυθεντικοποίησης και θα πρέπει η ίδια η συσκευή να φυλάσσεται καθώς σε περίπτωση απώλειάς της θα πρέπει να εκδοθεί νέα.

Παραδείγματα τέτοιων συσκευών είναι το RSA SecurID, το FortiToken και το SafeNet Token.

Σχήμα 2.4: TOTP Tokens

- HOTP (Event-Based One-Time Password Tokens)

Στα HOTP ο κωδικός παράγεται κάθε φορά που ο χρήστης πατάει ένα κουμπί. Στην συσκευή εμπεριέχεται ένας μετρητής ο οποίος κάθε φορά που η συσκευή χρησιμοποιείται, αυξάνει και δίνεται έτσι η εντολή για την παραγωγή του κωδικού. Αποτελεί ένα απλό σύστημα όπου ο συγχρονισμός δεν επηρεάζει τη λειτουργία. Όμως, ο χρήστης θα πρέπει να είναι αρκετά προσεκτικός καθώς το επαναλαμβανόμενο πάτημα του κουμπιού χωρίς τη χρήση του κωδικού που παράγεται μπορεί να προκαλέσει ασυμφωνία με τον sever.

Παραδείγματα τέτοιων συσκευών είναι το YubiKey και το DIGIPASS.

Σχήμα 2.5: HOTP Tokens

- USB Tokens

Τα USB tokens συνδέονται σε θύρες USB και η λειτουργία τους εξαρτάται από ένα απλό πάτημα ή την παρουσία απλώς του χρήστη. Τέτοιες συσκευές συχνά βασίζονται σε πρότυπα όπως το U2F (Universal 2nd factor). Αποτελούν ιδανική λύση στην αντιμετώπιση του Phishing ενώ προσφέρουν πολύ υψηλή ασφάλεια. Καθώς όμως απαιτούν σύνδεση σε θύρα USB θα πρέπει να υπάρχει φυσική πρόσβαση στον υπολογιστή ενώ υπάρχει και περιορισμός στη συμβατότητα με παλαιότερες πλατφόρμες.

Παραδείγματα USB tokens αποτελούν τα YubiKey και SoloKey.

Σχήμα 2.6: USB Tokens

- Smart Cards

Οι smart cards περιλαμβάνουν τσιπάκι το οποίο αποθηκεύει ψηφιακά πιστοποιητικά ενώ για τη χρήση τους απαιτείται smart card reader. Ο τύπος αυτός προσφέρει πολύ υψηλή ασφάλεια και μπορεί να χρησιμοποιηθεί ακόμη και για υπογραφές. Για το λόγο αυτό χρησιμοποιούνται πολύ συχνά από κυβερνητικές υπηρεσίες αλλά και επιχειρήσεις. Από την άλλη πλευρά, το κόστος τους είναι εξίσου υψηλό και πολλές φορές η διαχείρισή τους μπορεί να αποβεί αρκετά περίπλοκη.

Σχήμα 2.7: Smart Card Token

- Bluetooth/ NFC Tokens

Η χρήση των Bluetooth/ NFC Tokens προϋποθέτει σύνδεση με κάποιο smartphone ή Η/Υ. Η σύνεση πραγματοποιείται-όπως προκύπτει και από το όνομά τους-μέσω Bluetooth ή NFC και αποτελούν την πιο κατάλληλη επιλογή για Mobile εφαρμογές. Τα συγκεκριμένα token είναι πολύ γρήγορα στη χρήση τους και δίνεται και η δυνατότητα της φορητότητας. Βασική προϋπόθεση για τη χρήση τους είναι η υποστήριξη Bluetooth και NFC ενώ δεν είναι απίθανο να υπάρξουν δυσλειτουργίες σε ορισμένα περιβάλλοντα.

Παράδειγμα τέτοιου τύπου είναι το Yubikey 5 NFC.

Σχήμα 2.8: Bluetooth και NFC Tokens

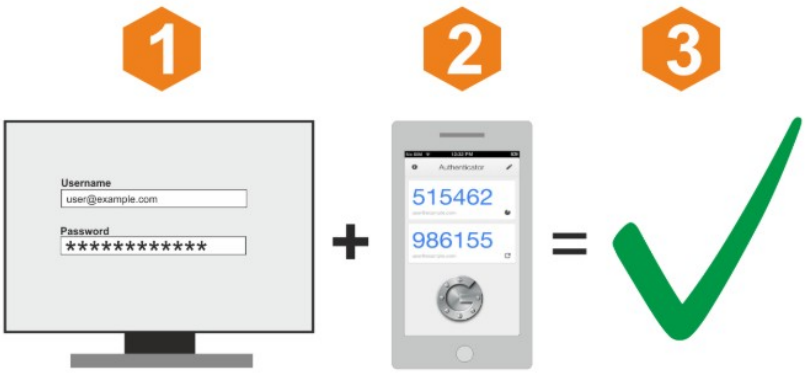
2.2.2 Software tokens

Τα Software Tokens είναι εφαρμογές ή υπηρεσίες οι οποίες λειτουργούν με τον ίδιο ακριβώς τρόπο όπως τα hardware tokens με τη διαφορά ότι δεν απαιτείται επιπλέον συσκευή αλλά λειτουργούν μέσω κινητού τηλεφώνου ή υπολογιστή.

- Authenticator Apps (TOTP)

Αποτελούν τον πιο συνηθισμένο τρόπο 2FA. Όπως και τα hardware tokens, μέσω των Authenticator Apps δημιουργείται ένας κωδικός στον χρήστη. Οι εφαρμογές αυτές είναι δωρεάν και η διαδικασία της εγκατάστασής τους πολύ εύκολη ενώ πολλές φορές δίνουν τη δυνατότητα δημιουργίας backup και sync, το οποίο είναι πολύ σημαντικό σε περίπτωση που χαθεί η το κινητό τηλέφωνο. Παρ' όλο που όπως αναφέρθηκε νωρίτερα πρόκειται για μια αντιστοιχία των TOTP στα hardware tokens, τα Authenticator Apps προσφέρουν πιο χαμηλή ασφάλεια.

Παραδείγματα τέτοιων εφαρμογών είναι οι Google Authenticator, Microsoft Authenticator, Authy και FreeOTP.



Σχήμα 2.9: Google Authenticator

- Push Notification Authentication

Η διαδικασία είναι πάρα πολύ απλή καθώς ο χρήστης λαμβάνει μια ειδοποίηση push στο κινητό του τηλέφωνο, την οποία είτε αποδέχεται είτε απορρίπτει. Με αυτό τον τρόπο αποφεύγονται λάθη κατά την πληκτρολόγηση του χρήστη όπως στην περίπτωση ενός OTP κωδικού, όμως η λειτουργία του στηρίζεται και στην δυνατότητα σύνδεσης στο διαδίκτυο αλλά και στην εύρυθμη λειτουργία του συστήματος push.

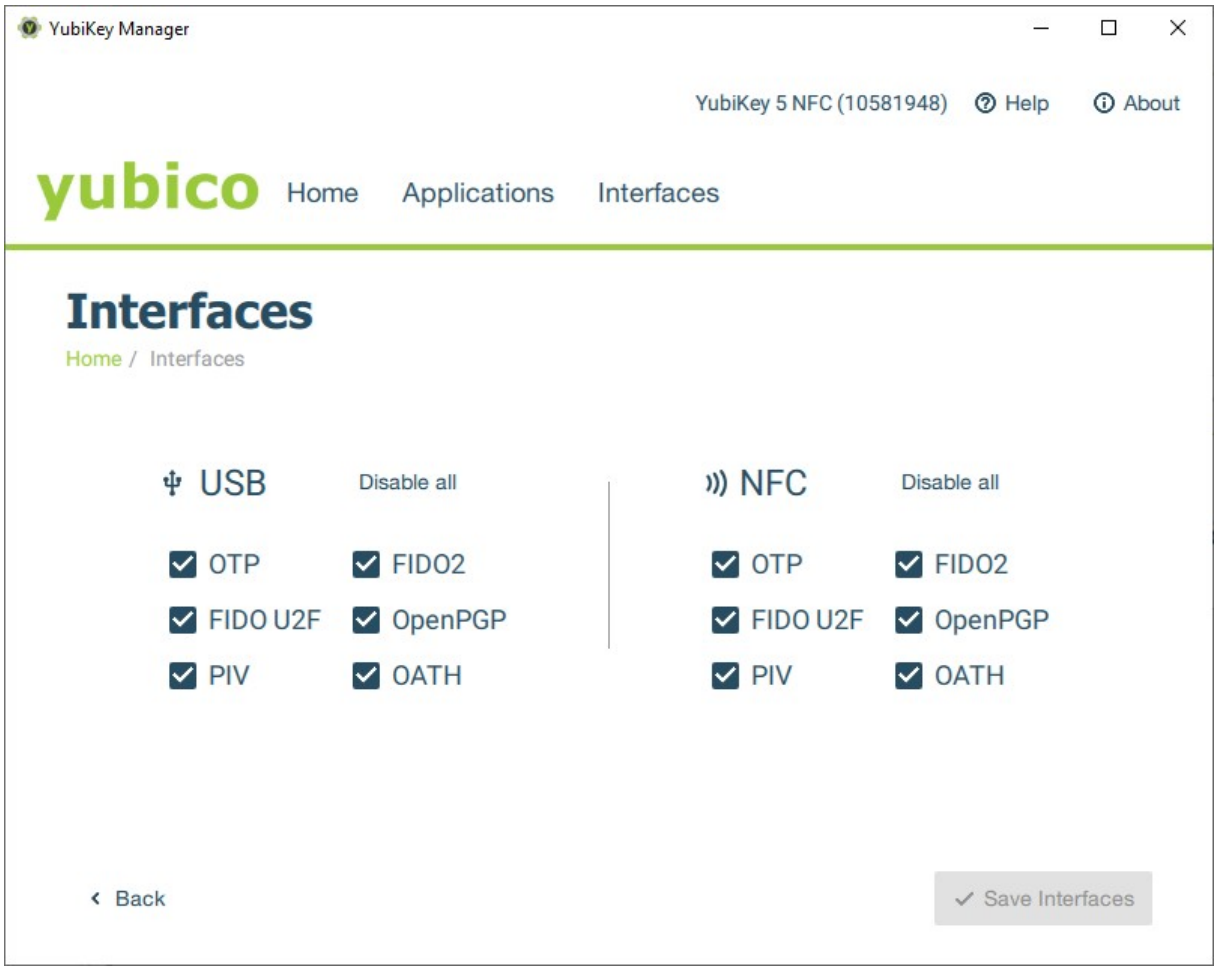
Παραδείγματα Push Notification Authentication tokens είναι τα Duo Mobile, Okta Verify και Microsoft Authenticator.

Σχήμα 2.10: Duo Mobile Authentication App

- Desktop Tokens

Τα συγκεκριμένα tokens χρησιμοποιούνται κυρίως από εταιρίες. Για να λειτουργήσουν γίνεται εγκατάσταση σε κάποιον Η/Υ είτε σε μορφή λογισμικού είτε σε μορφή επέκτασης browser.

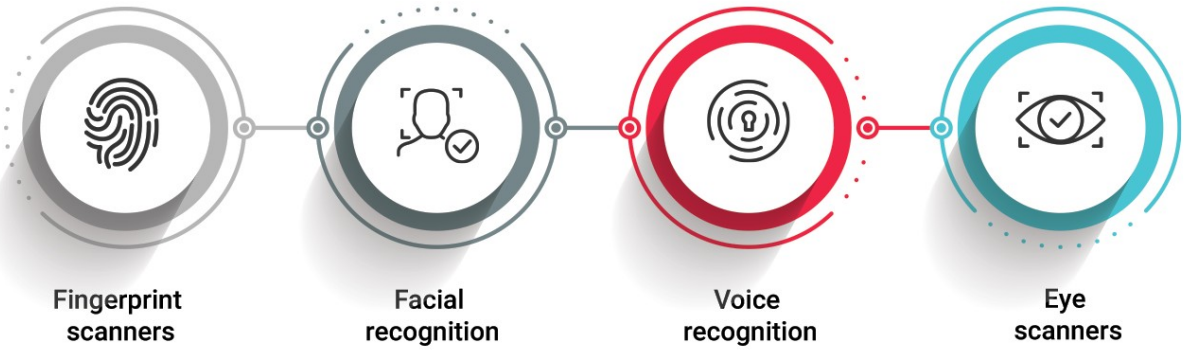
Παραδείγματα τέτοιων token είναι το SafeNet Authenticator και το YubiKey Manager.



Σχήμα 2.11: YubiKey Manager App

2.2.3 Biometric tokens

Πρόκειται για τα πιο συνηθισμένα token στα κινητά τηλέφωνα και στα laptops. Ο έλεγχος γίνεται μέσω βιομετρικών στοιχείων όπως είναι το δακτυλικό αποτύπωμα η αναγνώριση προσώπου. Η διαδικασία είναι πολύ απλή και ταυτόχρονα γρήγορη. Όμως, παρουσιάζονται και προβλήματα καθώς η λειτουργία τους στηρίζεται στην απόλυτη διατήρηση των βιομετρικών στοιχείων που χρησιμοποιούνται και οποιαδήποτε ακραία αλλαγή μπορεί να απορρίψει την πρόσβαση. Επίσης, τίθεται το θέμα της παραβίασης του βιομετρικού στοιχείου καθώς σε μια τέτοια περίπτωση είναι αδύνατη η αντικατάστασή του.



Σχήμα 2.12: Biometric Token

2.2.4 Web-based/ Cloud tokens

Πρόκειται για tokens που χρησιμοποιούνται από πλατφόρμες cloud όπως είναι το Google Workspace, το Microsoft 365, το AWS κ.α. Η λειτουργία τους βασίζεται στη γλώσσα σήμανσης δήλωσης ασφαλείας (SAML), στο πρωτόκολλο OAuth και στο πρωτόκολλο OpenID Connect. Η διαχείριση και η ενσωμάτωσή τους είναι πολύ εύκολη και μπορεί να συνδυαστεί και με άλλους τύπους token. Το αρνητικό των συγκεκριμένων token είναι ότι υπάρχει πλήρης εξάρτηση από τον Cloud Provider και πως το ρίσκο είναι πολύ μεγάλο στην περίπτωση παραβίασης του λογαριασμού.

Παραδείγματα Web-Based/ Cloud Tokens είναι τα SSO tokens, API tokens και Session tokens.

Στη συνέχεια παρατίθεται ένας πίνακας που προσδιορίζει τα προτεινόμενα token ανάλογα με το περιβάλλον χρήσης.

Πίνακας 2.2: Παραδείγματα Token ανάλογα με το περιβάλλον χρήσης

Περιβάλλον	**Προτεινόμενος τύπος token**
Τράπεζες / Χρηματοοικονομικά	Hardware tokens και Biometric
Επιχειρήσεις / Οργανισμοί	Software tokens και Push MFA
Εκπαίδευση / Πανεπιστήμια	Authenticator apps και SMS backup
Κυβερνητικός τομέας	Smart cards και Biometric
Remote εργασία	USB tokens (YubiKey) ή Duo push

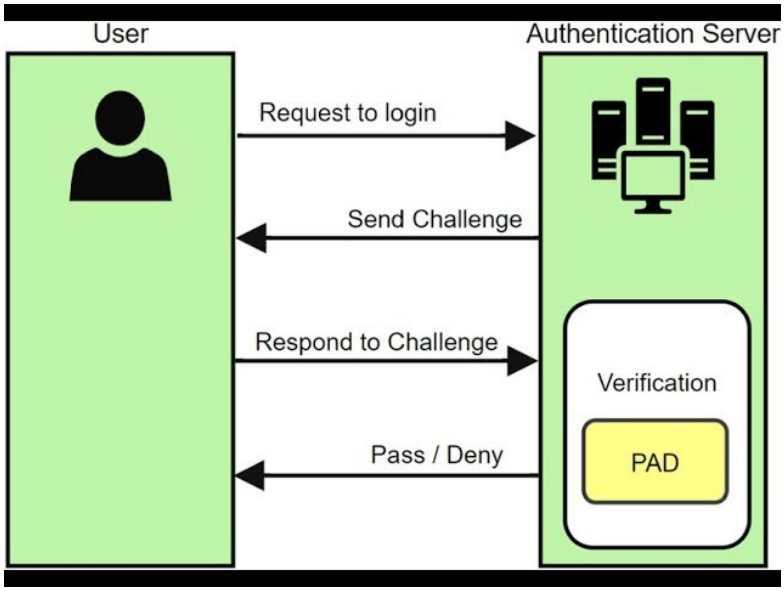
Η χρήση παρ' όλα αυτά των διαφόρων τύπων security token ποικίλει ανάλογα με το περιβάλλον στο οποίο χρησιμοποιείται. Έτσι, την τελευταία 5ετία φαίνεται ότι οι γενικοί χρήστες έχουν μειώσει τη χρήση των hardware tokens με αντίθεση με τα υψηλής ασφαλείας περιβάλλοντα τα οποία εξακολουθούν να τα χρησιμοποιούν και μάλιστα όλο και περισσότερο. Οι επιχειρήσεις αλλά και στις περιπτώσεις εξ' αποστάσεως εργασίας φαίνεται να υπάρχει η τάση μεικτού μοντέλου που περιλαμβάνει software tokens σε συνδυασμό με USB tokens.

2.3 Challenge Response Security Token

Το Challenge Response security token δεν αποτελεί ξεχωριστή κατηγορία. Καθώς όμως αποτελεί το βασικό κομμάτι αυτής της εργασίας, η ανάλυσή του θα γίνει μεμονωμένα.

Η διαφορά με ένα TOTP token είναι ότι δεν παράγεται απλώς ένας κωδικός. Το token λαμβάνει από το σύστημα κατά την ταυτοποίηση μια «πρόκληση». Ο χρήστης μέσω του token παράγει μια «απάντηση» η οποία συνδέεται με την «πρόκληση» και με ένα μυστικό κλειδί που έχει αποθηκευτεί μόνο στο token. Πιο συγκεκριμένα:

- Ο server στέλνει την «πρόκληση» η οποία συνήθως αποτελεί μια τυχαία τιμή.
- Το token αφού λάβει την «πρόκληση» χρησιμοποιεί έναν αλγόριθμο κρυπτογράφησης και σε συνδυασμό με το μυστικό κλειδί παράγει την απάντηση, η οποία είναι μοναδική για την συγκεκριμένη «πρόκληση».
- Η απάντηση του χρήστη αποστέλλεται πίσω στον server.
- Τέλος, ο server χρησιμοποιώντας το μυστικό κλειδί και την «απάντηση» που έλαβε, συγκρίνει την απάντηση και εάν το αποτέλεσμα ταιριάζει τότε η ταυτοποίηση είναι επιτυχής.



Σχήμα 2.13: Διάγραμμα λειτουργίας ενός Challenge-Response Token

Υπάρχουν 3 τύποι challenge-response token:

- Hardware-based

Ο χρήστης πληκτρολογεί την πρόκληση στη συσκευή και αυτή εμφανίζει στην οθόνη την απόκριση.

Παραδείγματα τέτοιων συσκευών είναι ειδικές εκδόσεις RSA SecurID, YubiKey (FIDO2, HMAC-SHA 1 challenge).

- Software-based

Ο χρήστης λαμβάνει την πρόκληση μέσω εφαρμογής και μέσω ενός αλγορίθμου και του μυστικού παράγεται η απόκριση.

Παραδείγματα τέτοιων συσκευών αποτελούν εφαρμογές custom mobile authentication και desktop tokens.

- FIDO U2F/ FIDO2

Η πρόκληση σχετίζεται με το domain και με τον αριθμό session nonce(number used once) της συνεδρίας. Ο αριθμός αυτός είναι τυχαίος και δημιουργείται ξεχωριστός για κάθε συνεδρία. Η συσκευή υπογράφει την πρόκληση μέσω του μυστικού.

Παραδείγματα τέτοιων token είναι τα YubiKey και Google Titan Key.

Ως συσκευή security token εμφανίζει τα εξής πλεονεκτήματα:

- Προσφέρει πολύ υψηλό επίπεδο ασφάλειας . Όπως θα παρουσιαστεί αργότερα, μπορεί να προστατεύσει τα δεδομένα από πολλά είδη ηλεκτρονικών επιθέσεων.
- Το μυστικό δεν μεταδίδεται ποτέ. Αυτό είναι το πιο σημαντικό κομμάτι του μηχανισμού, καθώς χωρίς αυτό δεν μπορεί να λειτουργήσει το σύστημα πρόκλησης-απόκρισης.
- Πλέον οι ηλεκτρονικές επιθέσεις έχουν εξελιχθεί αλλά το token αυτό μπορεί να προστατεύσει ακόμη και από νεότερες μορφές επιθέσεων.

Εμφανίζει όμως και τα εξής μειονεκτήματα:

- Σε πολλές περιπτώσεις απαιτείται εξοπλισμός αρκετά εξειδικευμένος όπως USB Θύρα ή NFC reader. Αυτό έχει ως αποτέλεσμα την αύξηση του κόστους του.
- Είναι πολύ σημαντικό η υλοποίηση του να γίνει σωστά σε ό,τι αφορά τον server-client καθώς μια όχι τόσο καλή υλοποίηση ανοίγει τις πόρτες σε πολλές επιθέσεις.
- Τέλος, θα πρέπει η συσκευή να φυλάσσεται καθώς η ανάκτησή της σε περίπτωση κλοπής αποτελεί μια διαδικασία πολύπλοκη.

Το συμπέρασμα που προκύπτει είναι ότι οι συσκευές αυτές προσφέρουν υψηλή ασφάλεια. Το κόστος όμως για την υλοποίηση τόσο των συσκευών όσο και των υποδομών για να μπορέσουν να λειτουργήσουν είναι εξίσου υψηλό. Για το λόγο αυτό συναντώνται σε υψηλής ασφάλειας συστήματα όπως τα τραπεζικά και εταιρείες.

Σχήμα 2.14: Challenge Response Security Tokens
Hardware-Based (αριστερά), Software-Based (δεξιά)

Κεφάλαιο 3ο: Θέματα ασφαλείας

3.1 Τύποι επιθέσεων

Για να γίνει κατανοητή η ασφάλεια που προσφέρουν τα security token θα πρέπει πρώτα να αναγνωριστούν οι κίνδυνοι που υπάρχουν. Είναι σημαντικό να γνωρίζουμε με ποιον τρόπο ένα token συμβάλλει στην προστασία των δεδομένων και κατά πόσο είναι αποτελεσματικό έναντι των διαφόρων τύπων επιθέσεων που μπορεί να δεχθεί.

Στη συνέχεια παρουσιάζονται οι βασικότεροι τύποι ηλεκτρονικών επιθέσεων που μπορεί να δεχθεί ένα security token.

- Ηλεκτρονικό Ψάρεμα (Phising)

Σε αυτή την περίπτωση η εξαπάτηση του χρήστη γίνεται μέσω υποκλοπής του κωδικού OTP ή του PIN, καθώς του ζητείται να πατήσει κάποιο link και να τα εισάγει σε κάποια ιστοσελίδα που δεν είναι η πραγματική ιστοσελίδα στην οποία ήθελε να εισέλθει. Έτσι, ο μη εξουσιοδοτημένος χρήστης χρησιμοποιεί τα στοιχεία του χρήστη σε πραγματικό χρόνο και έχει πρόσβαση στο σύστημα.

Για να μπορέσει να αντιμετωπιστεί το ηλεκτρονικό ψάρεμα θα πρέπει αρχικά οι χρήστες να είναι σωστά και καλά ενημερωμένοι. Ο τρόπος αυτός επίθεσης είναι πλέον πολύ συνηθισμένος και αρκετοί χρήστες δεν μπορούν να αναγνωρίσουν τις ψεύτικες ιστοσελίδες από τις επίσημες. Επίσης, για την αποφυγή του phising καλό είναι να χρησιμοποιούνται token με challenge-response ή τεχνικές anti-phising όπου μπορεί να ελεγχθεί και να αναγνωριστεί εάν το Link στο οποίο ανακατευθύνεται ο χρήστης είναι ασφαλές.

- Man in the Middle (MitM)

Στην περίπτωση αυτή δεν γίνεται ανακατεύθυνση του χρήστη μέσω κάποιου ξένου λινκ, αλλά ο μη εξουσιοδοτημένος χρήστης υποκλέπτει τον κωδικό token «μπαίνοντας» ανάμεσα στο χρήστη και το σύστημα και χρησιμοποιεί τον κωδικό πριν αυτός λήξει.

Η αντιμετώπιση αυτής της επίθεσης μπορεί να επιτευχθεί χρησιμοποιώντας και εδώ token με challenge response, διαφορετικά μπορεί να χρησιμοποιηθεί πρωτόκολλο TLS ώστε με τη βοήθεια της κρυπτογράφησης να μην μπορεί ο εισβολέας να υποκλέψει τα δεδομένα.

- Replay Attacks

Πρόκειται για μια πιο ιδιαίτερη επίθεση η οποία συμβαίνει μόνο στην περίπτωση που των συγχρονισμένων token όπου δεν ελέγχεται η επανάληψη της εισαγωγής του κωδικού OTP. Έτσι, ο μη εξουσιοδοτημένος χρήστης υποκλέπτει τον κωδικό και τον επαναχρησιμοποιεί με σκοπό να αποκτήσει πρόσβαση.

Με σκοπό την αντιμετώπιση της συγκεκριμένης επίθεσης μπορούν να χρησιμοποιηθούν μοναδικά session IDs, timestamps όπου καταγράφεται τόσο η ημερομηνία όσο και η ώρα που συμβαίνει κάποια ενέργεια αλλά και token μιας χρήσης όπου θα λήγουν έπειτα από μία χρήση και δεν θα μπορεί να επαναληφθεί η χρήση τους.

- Token Cloning-Device Theft

Εδώ το token είτε κλωνοποιείται είτε υποκλέπτεται και ο μη εξουσιοδοτημένος χρήστης το χρησιμοποιεί ως πραγματικός χρήστης. Η δυνατότητα πλήρους πρόσβασης είναι εφικτή σε μια τέτοια περίπτωση όταν δεν χρησιμοποιείται κάποιο Pin ή κάποιο βιομετρικό αναγνωριστικό.

Προς αποφυγή του token cloning ή του device theft πρέπει να χρησιμοποιούνται token με PIN ή να υπάρχει η δυνατότητα να απενεργοποιείται η συσκευή απομακρυσμένα.

- Malware-Keyloggers

Σε αυτή την περίπτωση μπορεί να συμβούν τα εξής: είτε να παρακολουθείται η πληκτρολόγηση κάποιου κωδικού που παράγει το token κι έπειτα να υποκλέπτεται είτε ο ίδιος ο υπολογιστής να αναγκαστεί να καταγράψει τα στοιχεία εισόδου.

Με σκοπό την αντιμετώπιση αυτών των επιθέσεων καλό είναι να χρησιμοποιούνται antivirus προγράμματα ή και τεχνικές όπως το sandboxing ή πλατφόρμες προστασίας endpoints.

- Desynchronization Attack

Η επίθεση αυτή αφορά τα Time-based token, όπου ο μη εξουσιοδοτημένος χρήστης προσπαθεί να δημιουργήσει αποσυγχρονισμό μεταξύ token και server ώστε να αποτύχει ο έλεγχος ταυτότητας.

Για την αντιμετώπισή του θα πρέπει να χρησιμοποιείται μικρό χρονικό παράθυρο ή να χρησιμοποιούνται μηχανισμοί fallback με σκοπό τον επανασυγχρονισμό του token με τον server.

- SIM Swapping

Και αυτή η επίθεση αφορά συγκεκριμένου τύπου token, τα οποία βασίζονται σε SMS. Ο μη εξουσιοδοτημένος χρήστης μαθαίνει τον αριθμό τηλεφώνου του χρήστη, πολύ συχνά μέσω social engineering (κοινωνική μηχανική) και τον μεταφέρει σε διαφορετική SIM, στην οποία θα σταλεί και το αντίστοιχο SMS με τον κωδικό.

Για να αποφευχθεί το SIM Swapping θα πρέπει αρχικά να αποφεύγεται η χρήση token που βασίζονται σε SMS (αποτελεί μια όχι και τόσο ασφαλή 2FA ταυτοποίηση) και να χρησιμοποιούνται εφαρμογές OTP όπως Google Authenticator.

3.2 Προστασία των tokens

Τα security tokens χρησιμοποιούνται όπως με σκοπό την ταυτοποίηση του χρήστη μέσω ενός ασφαλούς περιβάλλοντος καθώς όπως παρουσιάστηκε νωρίτερα υπάρχουν πολλά και διάφορα είδη απειλών. Τα security tokens διακρίνονται σε διάφορους τύπους και κάθε τύπος προστατεύει από

συγκεκριμένες απειλές. Στη συνέχεια, για τους πιο βασικούς τύπους παρουσιάζονται οι απειλές που μπορούν να αντιμετωπίσουν καθώς και οι περιορισμοί που τα συνοδεύουν.

- Hardware Tokens

Οι συσκευές αυτές είναι μικρές σε μέγεθος και φορητές. Η λειτουργία τους βασίζεται στη δημιουργία ενός κωδικού μιας χρήσης (OTP) ο οποίος ανανεώνεται ανά τακτά χρονικά διαστήματα. Η χρήση αυτής της συσκευής μπορεί να αντιμετωπίσει τον κίνδυνο του ηλεκτρονικού ψαρέματος καθώς ακόμη και αν ο κωδικός του χρήστη υποκλαπεί, ο OTP δεν θα έχει ισχύ για μεγάλο χρονικό διάστημα. Από την άλλη πλευρά, είναι συσκευές αρκετά ευάλωτες σε απειλές τύπου MitM όπου απαιτούνται περαιτέρω μέτρα προστασίας για την ασφάλεια του χρήστη.

- Smart Cards

Πρόκειται για κάρτες στις οποίες ενσωματώνεται chip και η χρήση τους προϋποθέτει και τη χρήση κάποιου κωδικού PIN. Η χρήση αυτής της συσκευής προσφέρει ασφάλεια απέναντι στην υποκλοπή των δεδομένων αυθεντικοποίησης καθώς αυτά δεν μεταδίδονται στατικά. Επίσης, για να γίνει η πρόσβαση είναι απαραίτητη η φυσική κάρτα και το PIN. Για να μπορέσει να χρησιμοποιηθεί η κάρτα θα πρέπει να υπάρχει reader. Σε αυτή την περίπτωση ο κίνδυνος που υπάρχει είναι το Malware στη συσκευή αυτή.

- Software Tokens

Είναι εφαρμογές σε κινητά smartphone, μέσω των οποίων δημιουργείται κωδικός OTP η Push Notification. Όπως και τα Hardware Tokens αντίστοιχα, η χρήση OTP προσθέτει ακόμη έναν παράγοντα και υπάρχει μεγαλύτερη ασφάλεια ενώ ταυτόχρονα η συχνή ανανέωση του κωδικού OTP αποτρέπει την υποκλοπή του. Παρ' όλα αυτά δεν παύουν σαν εφαρμογές να εξαρτώνται από το κινητό στο οποίο είναι εγκατεστημένες, το οποίο μπορεί να εκτεθεί σε Malware και να μολυνθεί με αποτέλεσμα την παράκαμψή του σε περίπτωση πρόσβασης. Τέλος, να αναφερθεί πως πρόκειται για tokens αρκετά ευάλωτα στο ηλεκτρονικό ψάρεμα.

- USB Security Key

Αυτές οι συσκευές είναι είτε USB είτε NFC και αποτελούν κομμάτι της Hardware-based αυθεντικοποίησης. Η λειτουργία τους βασίζεται στη χρήση Challenge-Response ενώ ταυτόχρονα ενώνονται με έναν τομέα (domain) στο δίκτυο με αποτέλεσμα να χρησιμοποιείται κεντρική ταυτοποίηση. Έτσι είναι τα πλέον ασφαλέστερα απέναντι σε απειλές ηλεκτρονικού ψαρέματος και MitM. Ο περιορισμός αυτών των συσκευών είναι ότι είναι απαραίτητη η παρουσία του χρήστη και θα πρέπει να υπάρχει συμβατότητα με το σύστημα.

- Biometric Tokens

Τα βιομετρικά χαρακτηριστικά του χρήστη είναι αυτά που χρησιμοποιούνται για ταυτοποίηση σε αυτές τις συσκευές. Κατάλληλα απέναντι σε επιθέσεις Malware και Keyloggers καθώς δεν

πληκτρολογείται κανένας κωδικός οπότε δεν μπορεί να υποκλαπεί. Η παραποίηση ταυτότητας (spoofing) μπορεί να χρησιμοποιηθεί για την πρόσβαση μη εξουσιοδοτημένων χρηστών σε κάποιο σύστημα επομένως αποτελεί σημαντικό μειονέκτημα η μη δυνατότητα αλλαγής του βιομετρικού σε περίπτωση παραβίασης.

- Challenge-Response Tokens

Πρόκειται είτε για συσκευές είτε για λογισμικό που χρησιμοποιεί έναν συνδυασμό πρόκλησης-απόκρισης μέσω ενός αλγορίθμου που έχει κρυπτογραφηθεί και ενός μυστικού που βρίσκεται μέσα του και δεν μεταδίδεται ποτέ. Το συγκριμένο είδος token αντιμετωπίζει τις εξής απειλές:

Ηλεκτρονικό ψάρεμα: Η πρόκληση που δημιουργείται είναι συνδεδεμένη με το domain με αποτέλεσμα να μην μπορεί η απάντηση να χρησιμοποιηθεί αλλού και καθίσταται άχρηστη.

Replay attacks: Η πρόκληση είναι κάθε φορά μοναδική επομένως η χρήση της δεν μπορεί να επαναληφθεί.

Credential theft: Το μυστικό όπως προαναφέρθηκε, δεν μεταδίδεται ποτέ. Επομένως ακόμη κι αν κάποιος καταφέρει να υποκλέψει την απόκριση, δεν θα έχει τη δυνατότητα να την χρησιμοποιήσει ξανά.

MitM: Για να είναι δυνατή η προστασία από τη συγκεκριμένη απειλή θα πρέπει το σύστημα να είναι σωστά υλοποιημένο και έτσι, ο μη εξουσιοδοτημένος χρήστης δεν μπορεί να μεσολαβήσει.

Brute force: Η επίθεση εξαντλητικής αναζήτησης μπορεί να καταπολεμηθεί καθώς οι αποκρίσεις που παράγονται στο challenge-response σύστημα κρυπτογραφούνται ισχυρά. Η μη μετάδοση του μυστικού που είναι αποθηκευμένο στο token καθιστά πολύ δύσκολο να παραχθεί η απόκριση που δημιουργείται στην εκάστοτε περίπτωση.

Από την άλλη πλευρά όμως, η χρήση των challenge-response token περιορίζεται στα εξής:

Κλοπή: Στην περίπτωση του hardware challenge-response token, η κλοπή του ίδιου του token αποτελεί μεγάλο πρόβλημα αλλά εάν το token συνοδεύεται και από το PIN ή το μυστικό, μπορεί να καταλήξει σε κατάχρηση.

Κακόβουλο endpoint: Θα πρέπει για τη χρήση του συγκεκριμένου token ο πελάτης να κρατάει ασφαλή τον υπολογιστή του. Εάν ο υπολογιστής έχει Malware, τότε είναι πιθανό να παρακολουθούνται οι είσοδοί του και η απόκριση να χρησιμοποιηθεί άμεσα .

Διαρροή μυστικού: Όλος ο μηχανισμός challenge-response βασίζεται στο μυστικό και στο γεγονός ότι δεν μεταδίδεται. Εάν για κάποιο λόγο γίνει γνωστό το μυστικό τότε καταρρέουν όλα και το token αχρηστεύεται.

Κεφάλαιο 4ο: Σχεδιασμός της συσκευής

4.1 Εισαγωγή στον σχεδιασμό

Στην παρούσα εργασία θα σχεδιαστεί ένα Challenge-Response Hardware-based Token. Η επιλογή αυτού του είδους token έγινε με γνώμονα την ασφάλεια που προσφέρει σε σχέση με πολλά άλλα είδη. Θεωρείται ίσως μια όχι και τόσο εξελιγμένη μορφή token, δεν παύει όμως να χρησιμοποιείται ακόμη και στις μέρες μας από μεγάλες εταιρείες και επιχειρήσεις.

Στη συνέχεια θα αναλυθεί τόσο η σχεδίαση της συσκευής ως υλικό αλλά και το λογισμικό και ο κώδικας που χρησιμοποιήθηκαν ώστε να μπορέσει η συσκευή να τεθεί σε λειτουργία.

4.2 Hardware συσκευής

Το βασικό στοιχείο του κατασκευαστικού μέρους της συσκευής ήταν η επιλογή του μικρελεγκτή που θα χρησιμοποιούνταν. Χρησιμοποιήθηκε ο PIC18F1320 με βάση τα εξής χαρακτηριστικά:

- Αρχιτεκτονική: 8-bit RISC. Η σειρά PIC18 είναι ισχυρότερη από την σειρά PIC16.
- Ταχύτητα: Η ταχύτητά του φτάνει τα 40MHz με PPL, η οποία είναι αρκετή για LFSR.
- Flash: 8 KB, η οποία αρκεί για τη δημιουργία token generator.
- RAM: 256 bytes, η οποία είναι αρκετή για τις απαιτούμενες μεταβλητές και πίνακα OTP.
- I/O pins: 11 ψηφιακές θύρες οι οποίες καλύπτουν την ανάγκη για σύνδεση οθόνης και πληκτρολογίου.
- EEPROM: 256 Bytes, με τα οποία δίνεται η δυνατότητα για αποθήκευση seed ή serial.

Λαμβάνοντας υπόψη τα παραπάνω, επιλέχθηκε ο συγκεκριμένος PIC, γύρω από τον οποίο σχεδιάστηκε και το υπόλοιπο κύκλωμα. Ο μικροελεγκτής αυτός αποτελεί μια καλή επιλογή για το είδος του token που πρόκειται να υλοποιηθεί, έχει αρκετούς πόρους για μια αξιόπιστη υλοποίηση, είναι κατάλληλος για απλές και offline συσκευές αλλά δεν μπορεί να ανταποκριθεί σε πιο εξελιγμένα συστήματα τα οποία απαιτούν μεγαλύτερη ασφάλεια αλλά ούτε και σε συστήματα τα οποία απαιτούν σύγχρονη συνδεσιμότητα.

Πριν την ανάλυση του σχεδιασμού, επισημαίνεται ότι η συσκευή δημιουργείται με σκοπό την παραγωγή κωδικών, δεν συνδέεται με τον υπολογιστή ή το δίκτυο (disconnected token) και χρησιμοποιείται από το χρήστη χειροκίνητα μέσω της πληκτρολόγησης των κωδικών.

Για το σχεδιασμό του token χρησιμοποιήθηκε το πρόγραμμα σχεδίασης ηλεκτρονικών κυκλωμάτων Altium Designer.

Αρχικά δημιουργήθηκε νέο project. Από το menu επιλέγουμε File→New →Project → PCB Project.

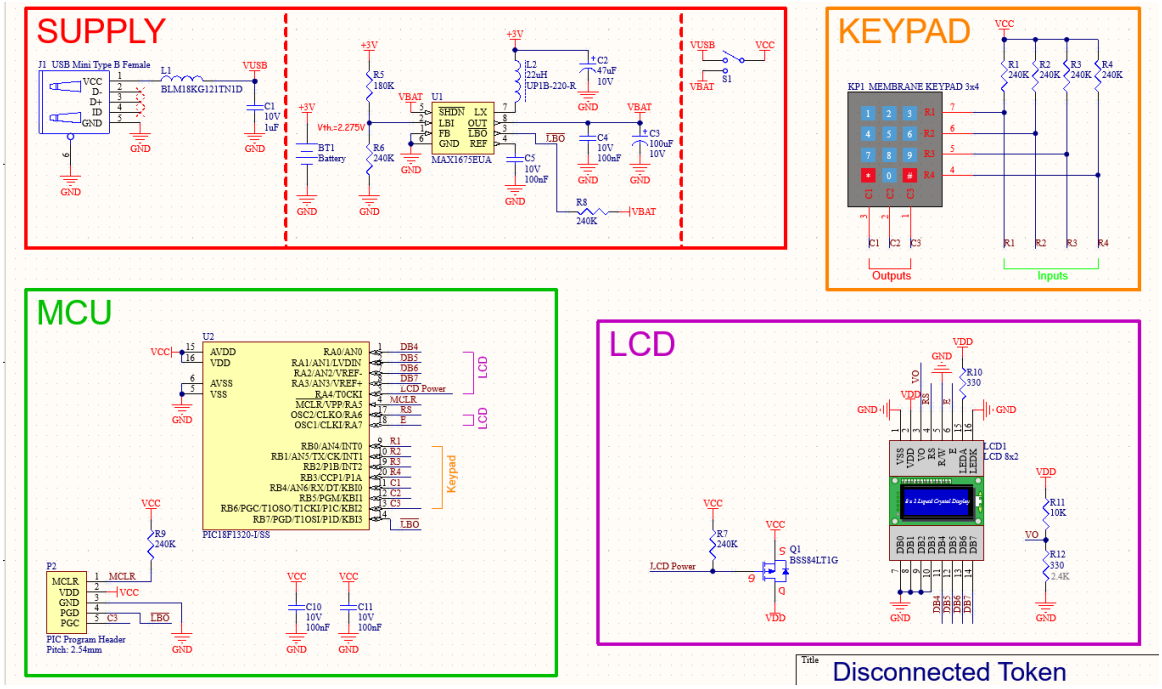
Στη συνέχεια προσθέτουμε Schematic Sheet. Από το Project panel που βρίσκεται αριστερά, επιλέγουμε το project που δημιουργήσαμε. Έπειτα επιλέγουμε Add New to Project → Schematic.

Σε αυτό το σημείο μπορούμε να ξεκινήσουμε την τοποθέτηση των εξαρτημάτων. Από το Components Panel που βρίσκεται δεξιά, κάνουμε αναζήτηση το εξάρτημα που θέλουμε και το σέρνουμε μέσα στο schematic. Σε περίπτωση που το Panel δεν εμφανίζεται, πηγαίνουμε Panels → Components.

Εφόσον τοποθετηθούν όλα τα απαραίτητα εξαρτήματα, πρέπει να γίνουν οι συνδέσεις των καλωδίων ή αλλιώς nets. Από την γραμμή εργαλειών επιλέγουμε Place Wire ή εναλλακτικά πατάμε τα πλήκτρα P → W. Για να ολοκληρωθεί μία σύνδεση, κάνουμε κλικ σε κάποιο Pin και ξανά κλικ στο Pin με το οποίο θέλουμε να ενωθεί.

Αφού ολοκληρωθεί η τοποθέτηση και σύνδεση όλων των απαραίτητων εξαρτημάτων, γίνεται έλεγχος για λάθη. Πηγαίνουμε Project → Validate Project και το Altium κάνει έλεγχο για λάθη μέσω του ERC (Electrical Rule Check).

Το schematic που δημιουργήθηκε για την παρούσα εργασία φαίνεται στο παρακάτω σχήμα.



Σχήμα 4.15: Σχηματικό διάγραμμα Challenge-Response Security Token

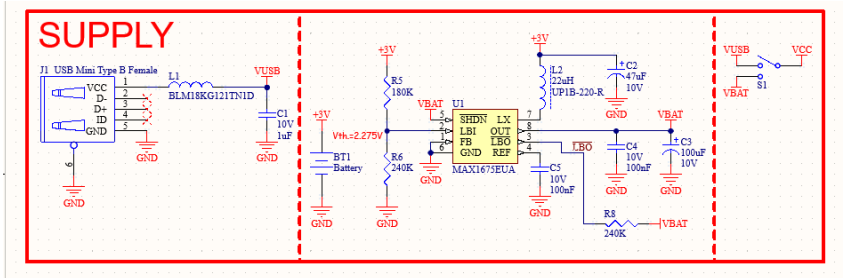
Η λειτουργία του παραπάνω κυκλώματος ακολουθεί την εξής ροή:

- Λαμβάνεται η τροφοδοσία από το USB ή την μπαταρία.
- Ενεργοποιείται ο μικροελεγκτής.
- Ο χρήστης εισάγει δεδομένα μέσω του πληκτρολογίου.
- Ο μικροελεγκτής εκτελεί τον κώδικα LFSR και επεξεργάζεται τα δεδομένα
- Το αποτέλεσμα εμφανίζεται στην οθόνη LCD.

Όπως φαίνεται, το σχηματικό είναι χωρισμένο σε 4 μέρη. Κάθε ένα από αυτά παρουσιάζει και ένα ξεχωριστό μέρος του κυκλώματος, τα οποία θα αναλυθούν παρακάτω.

SUPPLY

Πρόκειται για το κύκλωμα τροφοδοσίας του κυκλώματος, η οποία παρέχεται είτε με τη χρήση μπαταρίας είτε μέσω θύρας USB.



Σχήμα 4.16: Κύκλωμα τροφοδοσίας

Σχετικά με το κύκλωμα USB:

- Τα PIN1(VCC) και PIN5 (GND) χρησιμοποιούνται για την τροφοδοσία.
- Τα PIN2 (D-) και PIN3 (D+) δεν είναι συνδεδεμένα καθώς δεν θα υπάρχει στο κύκλωμα επικοινωνία με USB.
- Το L1 χρησιμοποιείται ως φίλτρο θορύβου για τις υψηλές συχνότητες της τροφοδοσίας.
- Ο πυκνωτής αποσύζευξης C1 χρησιμοποιείται με σκοπό την σταθεροποίηση της τάσης.

Σχετικά με το κύκλωμα του Step-up DC/DC converter (MAX1675)

Σκοπός του είναι να αυξήσει και να σταθεροποιήσει την τάση από τη μπαταρία στα 3,3V (VBAT).

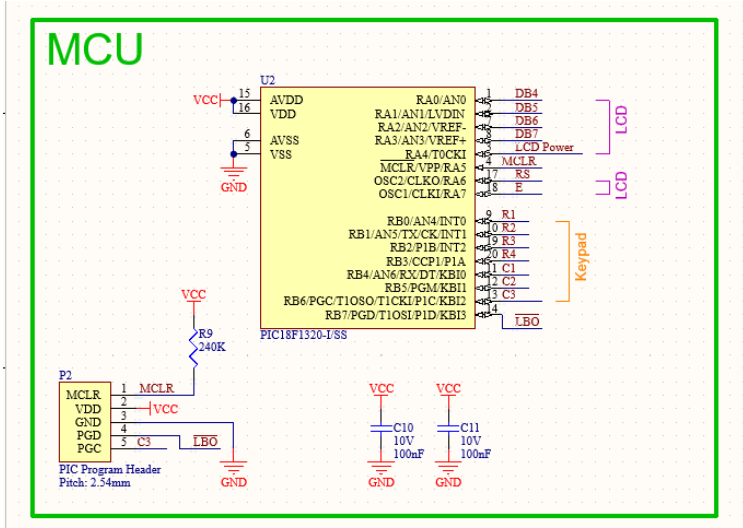
- Το L2 χρησιμοποιείται για boost converter.
- Οι πυκνωτές C2 και C3 λειτουργούν ως φίλτρα εισόδου και εξόδου.
- Οι αντιστάσεις R5 και R6 σχηματίζουν έναν διαιρέτη τάσης με σκοπό την παρακολούθηση μέσω του LBI.
- Τα στοιχεία SHDN, LBI και LBO χρησιμοποιούνται για τη διαχείριση της κατανάλωσης.

Τέλος, το S1 αποτελεί τον επιλογέα της πηγής ανάμεσα VUSB και VBAT.

MCU

Αποτελεί το κύκλωμα του μικροελεγκτή που αποτελεί τον κεντρικό ελεγκτή του συστήματος. Είναι υπεύθυνος για την εκτέλεση του κώδικα LFSR/OTP και αυτός που διαχειρίζεται το πληκτρολόγιο, την οθόνη και την τροφοδοσία.

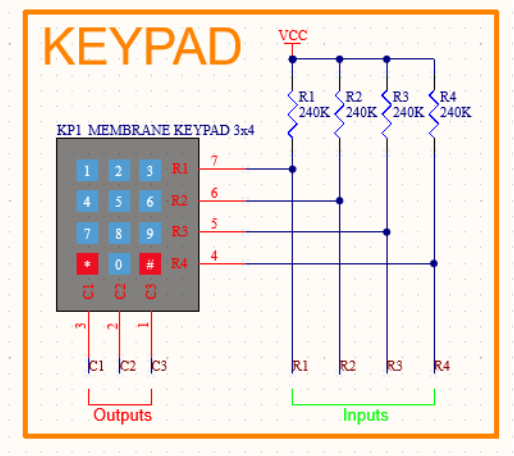
- Τα PIN RAx, RBx αποτελούν τις ψηφιακές εισόδου και εξόδους του PIC.
- Τα PIN OSC1, OSC2 χρησιμοποιούνται για τον χρονισμό .
- Το PIN MCLR είναι για επαναφορά.
- Τα PIN PGC, PGD σχετίζονται με τον προγραμματισμό μέσω ICSP.
- Το P2 αποτελεί το ICSP Header μέσω του οποίου γίνεται η σύνδεση για τον προγραμματισμό του PIC.
- Οι πυκνωτές αποσύζευξης C10 και C11 χρησιμοποιούνται για την σταθεροποίηση της τροφοδοσίας κοντά στον μικροελεγκτή.
- Η αντίσταση R9 χρησιμοποιείται ως pull-up στο MCLR με σκοπό την ενεργοποίηση του reset.



Σχήμα 4.17: Κύκλωμα μικροελεγκτή

KEYPAD

Είναι το κύκλωμα του πληκτρολογίου το οποίο συνδέεται με τον MCU ώστε να μπορεί ο χρήστης να εισάγει δεδομένα (είτε την πρόκληση είτε το PIN). Πρόκειται για ένα πληκτρολόγιο μεμβρανικής αριθμητικής διάταξης 3x4, ενώ περιέχει τόσο αριθμητικά όσο και λειτουργικά πλήκτρα.

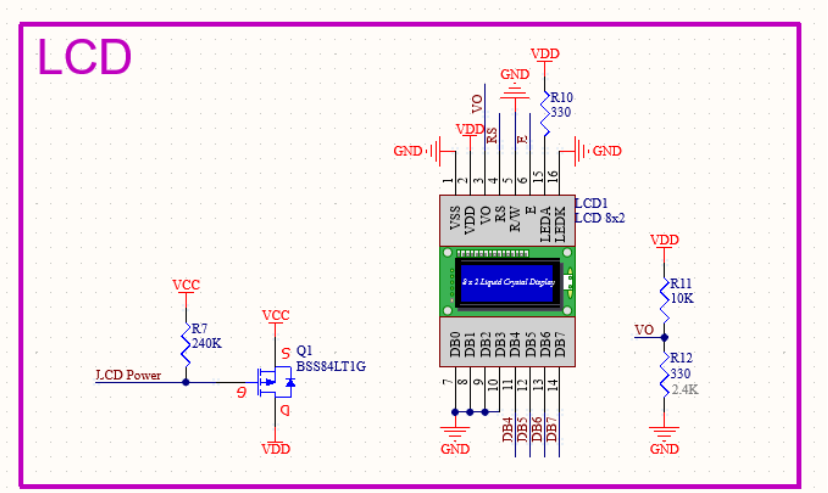


Σχήμα 4.18: Κύκλωμα πληκτρολογίου

- Οι γραμμές R1-R4 αποτελούν τις γραμμές εξόδου (σύνδεση στα αντίστοιχα pin του MCU).
- Οι γραμμές C1-C3 αποτελούν τις γραμμές εισόδου (σύνδεση στα αντίστοιχα pin του MCU).
- Οι αντιστάσεις R1-R4 χρησιμοποιούνται ως pull-up για την αποφυγή floating στις εισόδους.

LCD

Αποτελεί το κύκλωμα της οθόνης. Πρόκειται για οθόνη LCD 2x16 χαρακτήρων με HD44780 driver. Χρησιμοποιείται για την προβολή των διάφορων μηνυμάτων (π.χ. πρόκληση).



Σχήμα 4.19: Κύκλωμα οθόνης

- Τα PIN DB4-DB7 χρησιμοποιούνται για την μετάδοση δεδομένων 4-Bit από και προς τον MCU.
- Τα PIN RS και E αποτελούν τα pin ελέγχου.
- Η αντίσταση R10 χρησιμοποιείται για τον περιορισμό του ρεύματος στο backlight.
- Η αντίσταση R11 χρησιμοποιείται για το contrast μέσω VO.
- Η αντίσταση R12 χρησιμοποιείται για contrast adjust και σχηματίζει με την R11 διαιρέτη τάσης.
- Το Q1 είναι P-MOSFET και χρησιμοποιείται για την ενεργοποίηση και απενεργοποίηση της τροφοδοσίας της οθόνης LCD.
- Η αντίσταση R7 αποτελεί pull-up αντίσταση στην πύλη του Q1.
- Σημειώνεται πως η οθόνη μπορεί να απενεργοποιηθεί για εξοικονόμηση ενέργειας.

Στον παρακάτω πίνακα καταγράφονται τα υλικά που χρησιμοποιήθηκαν για τη δημιουργία του schematic.

Πίνακας 4.3: Λίστα υλικών schematic

Είδος/Υλικό	Volt	Περιγραφή	Συμβολισμός	Footprint	Ποσότητα
Battery		Multicell Battery	BT1	BAT-HLD-001	**1**
1uF	10V	Capacitor (Semiconductor SIM Model)	C1	0603-1608	**1**
47uF	10V	Polarized Capacitor (Surface Mount)	C2	TC3528-1210	**1**
100uF	10V	Polarized Capacitor (Surface Mount)	C3	TC7343-2917	**1**
100nF	10V	Capacitor	C4, C5, C10, C11	0402-1005	**4**

		(Semiconductor SIM Model)			
USB Mini Type B Female		USB Mini Type B Female	J1	USB Mini Type B Female SMD	**1**
MEMBRANE KEYPAD 3x4		Membrane Keypad 3x4	KP1	HDR1X7	**1**
Inductor		Inductor	L1	0603-1608	**1**
Inductor - ferroxcube core		Inductor - ferroxcube core	L2	Inductor UP1B Series	**1**
LCD 8x2		Liquid Crystal Display 8x2	LCD1	LCD 2X8	**1**
PIC Program Header		Program Header	P2	HDR1X5	**1**
BSS84LT1G		P-Channel Power MOSFET	Q1	SOT-23-3	**1**
10K		Resistor	R1, R2, R3, R4, R7, R9, R11	0603-1608	**7**
0ohm		Resistor	R5	0603-1608	**1**
240K		Resistor	R6	0603-1608	**1**
100K		Resistor	R8	0603-1608	**1**
330		Resistor	R10, R12	0603-1608	**2**
SW-SPDT		Single Pole Double Throw	S1	SPDT Mini SMD Slide Switch	**1**
MAX1675EUA		High-Efficiency, Low-Supply-Current, Compact, Step-Up DC-DC Converter	U1	UMAX8_L	**1**
PIC18F1320-I/SS		High Performance, Enhanced Flash Microcontroller with8K Flash, 20-Pin SSOP	U2	SSOP-SS20_M	**1**

4.3 Software συσκευής

Για την υλοποίηση του token απαραίτητη προϋπόθεση είναι η δημιουργία κατάλληλου κώδικα. Ο γενικός τρόπος λειτουργίας του συστήματος περιγράφεται στη συνέχεια.

- Ο χρήστης πληκτρολογεί 2 αριθμούς, τον ΑΦΜ του και το barcode της συσκευής. Τα στοιχεία αυτά αποτελούν την πρόκληση (challenge).
- Στη συνέχεια μέσω κώδικα LFSR υπολογίζεται ένας OTP ο οποίος αποτελεί τελικά την απόκριση (response).
- Ο server (backend) από την άλλη γνωρίζει τον ΑΦΜ και το barcode της συσκευής του χρήστη και εκτελεί τον ίδιο κώδικα LFSR με τις ίδιες εισόδους.
- Αν ο OTP που στέλνει ο χρήστης ταιριάζει με το υπολογισμένο, τότε δίνεται έγκριση πρόσβασης.

Οι κώδικες που χρησιμοποιήθηκαν για τον προγραμματισμό του μικροελεγκτή συντάχθηκαν σε γλώσσα C. Στη συνέχεια θα γίνει ανάλυση των βασικών σημείων του κώδικα ώστε να γίνει κατανοητός ο τρόπος λειτουργίας της συσκευής, ενώ ολόκληροι οι κώδικες μπορούν να βρεθούν στα παραρτήματα της εργασίας.

Η εκτέλεση του κώδικα ακολουθεί συνοπτικά την εξής ροή:

- Ο κώδικας διαβάζει την είσοδο από το πληκτρολόγιο σε διαδοχικά ψηφία.
- Στη συνέχεια γίνεται έλεγχος των ψηφίων και συγκρίνονται με προκαθορισμένους κωδικούς για την ορθότητά τους.
- Δημιουργείται ένας πίνακας OTP μέσω του αλγορίθμου LFSR, ο οποίος βασίζεται στο ΑΦΜ του χρήστη και το barcode της συσκευής.
- Ως επιπλέον επίπεδο ασφαλείας, γίνεται ανταλλαγή των τελευταίων ψηφίων του barcode.
- Για κάθε θέση δημιουργείται ένα ζευγάρι κωδικών εισόδου-εξόδου (site numbers).
- Τέλος, ο κωδικός εμφανίζεται στην LCD ώστε να μπορέσει ο χρήστης να τον χρησιμοποιήσει και ο κώδικας μπαίνει σε αναμονή για ολοκλήρωση της διαδικασίας.

Τα σημαντικότερα σημεία του κώδικα LFSR αναλύονται στη συνέχεια.

1. Αρχικά εισάγονται από τον χρήστη οι αριθμοί που αντιπροσωπεύουν τον ΑΦΜ και το barcode της συσκευής.

```
scanf( "%u", &AFM );
scanf( "%u", &BARCODE );
```

2. Ακολουθεί η αρχικοποίηση LFSR όπου το περιεχόμενο είναι το αποτέλεσμα της πράξης XOR των 2 παραπάνω αρθμών.

```
lfsr = AFM ^ BARCODE;
```

3. Στη συνέχεια δημιουργείται το loop για την παραγωγή 36 τυχαίων αριθμών των 8 bit. Το LFSR εφαρμόζει το πολυώνυμο:

```
x^32 + x^30 + x^28 + x^27 + x^24 + x^19 + x^14 + 1
```

και αυτό μεταφράζεται στον εξής υπολογισμό:

```
xor_out  = ( ( lfsr >> 0 ) ^ ( lfsr >> 2 ) ^ ( lfsr >> 4 ) ^
```

```
            ( lfsr >> 5 ) ^ ( lfsr >> 8 ) ^ ( lfsr >> 13 ) ^
            ( lfsr >> 18 ) ) & 0x00000001;
lfsr =  ( lfsr >> 1 ) | ( xor_out << 31 );
```

Έτσι, για κάθε 8 κύκλους, δημιουργείται ένας 8-bit αριθμός.

Οι αριθμοί που δημιουργούνται αναπαριστούν τους OTP κωδικούς οι οποίοι εκτυπώνονται σε μορφή HEX (δεκαεξαδικοί αριθμοί) και αποθηκεύονται στον πίνακα `temp_table[36]` για περαιτέρω χρήση.

4. Ακολουθεί η κάθετη εκτύπωση των αριθμών. Κάθε OTP προβάλλεται με αριθμό θέσης.

```
fprintf( OTP, "Table place: %d ---> Element: *0x%02x*\n", i,
temp_table[ i ] );
```

5. Έπειτα γίνεται η τελική επεξεργασία όπου δημιουργούνται τα site input/output codes. Για κάθε θέση 0–35:

   a) Υπολογισμός site number:

   ```
   site_number = i ^ BARCODE_two_last_digits;
   ```

   `BARCODE_two_last_digits` = τα 2 τελευταία ψηφία του barcode.

   b) **Αντιστροφή ψηφίων (swap_digit):**

   ```
   lsb = 82 & 0xF0; // Ανώτερο nibble
   lsb = lsb >> 4;

   msb = 82 & 0x0F; // Κατώτερο nibble
   msb = msb << 4;

   swap_digit= msb | lsb;
   ```

   c) Δημιουργία δύο αριθμών:

   ```
   first_number  = temp_table[ i ] ^ swap_digit;
   second_number = temp_table[ i ] | swap_digit;
   ```

   d) Εκτύπωση:

   ```
   fprintf( OTP, "%d --> Site output: *%03d*  --->  Site input: *%03d
   %03d*\n", i, site_number, first_number, second_number );
   ```

6. Τέλος, παράγεται το αρχείο OTP.txt το οποίο περιέχει τη λίστα με τους OTPs σε μορφή HEX, την κάθετη λίστα OTPs με την αντίστοιχη θέση στον πίνακα καθώς και τα site input/output codes.

Κεφάλαιο 5ο: Υλοποίηση της συσκευής

5.1 Δημιουργία PCB

Μετά την ολοκλήρωση της μελέτης του σχεδιασμού του Token, ακολουθεί η υλοποίησή της σε πλακέτα. Αρχικά, θα πρέπει μέσω του λογισμικού Altium Designer να δημιουργηθεί το PCB αρχείο ώστε να μπορέσει να τυπωθεί η πλακέτα. Για τις ανάγκες της εργασίας χρησιμοποιήθηκε πλακέτα διπλής όψης και βάσει αυτού έγινε η δημιουργία του PCB.

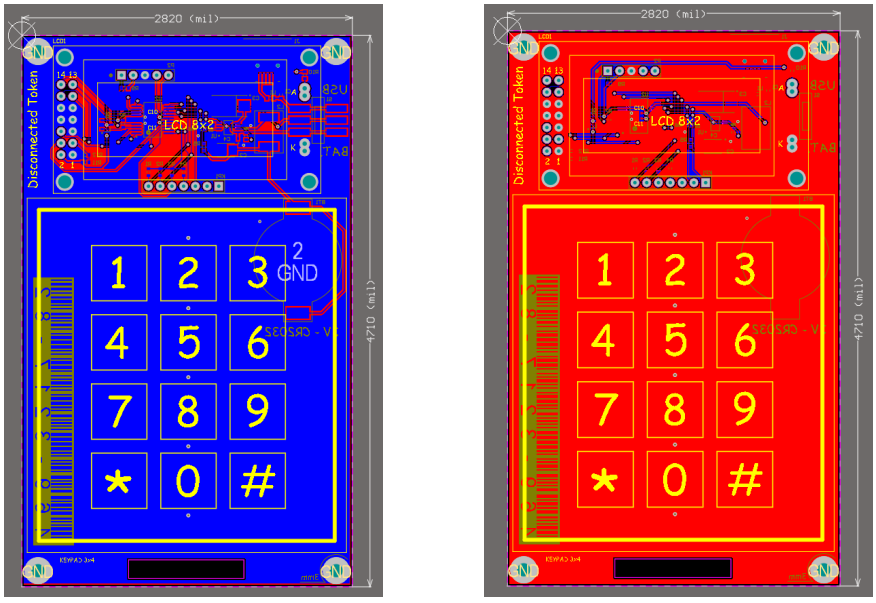
Για να εξάγουμε το αρχείο μέσω του λογισμικού σχεδίασης ακολουθούμε την εξής διαδικασία:

Κάνουμε δεξί κλικ στο Project που είχαμε δημιουργήσει και επιλέγουμε Add New to Project →PCB. Στη συνέχεια ανοίγουμε το Schematic και επιλέγουμε Design → Import Changes to PCB. Σε αυτό το σημείο όλα τα εξαρτήματα (components) που ήταν τοποθετημένα στο σχηματικό μεταφέρονται στην πλακέτα. Τα εξαρτήματα εμφανίζονται εκτός πλαισίου οπότε θα πρέπει αρχικά να σχεδιαστεί το outline του board και στη συνέχεια να τοποθετηθούν και αυτά στο Board.

Τέλος, γίνονται οι συνδέσεις μεταξύ των εξαρτημάτων με tracks, δημιουργούμε δηλαδή τις αγώγιμες διαδρομές. Επιλέγουμε Design → Netlist → Update free primitives. Πολλές φορές, με σκοπό το κέρδος χρόνου επιλέγουμε να κάνουμε Auto-Routing. Με τον τρόπο αυτό, το λογισμικό συνδέει τα εξαρτήματα μέσω tracks και παράλληλα ελέγχει ώστε να μην υπάρχουν βραχυκυκλώματα. Αυτός ο τρόπος διασύνδεσης επιλέγεται κυρίως σε πιο απλά κυκλώματα και μπορεί να ολοκληρωθεί επιλέγοντας Route → Auto Route → All.

Το επόμενο στάδιο είναι ο έλεγχος του σχεδίου. Στο στάδιο του schematic είχε ολοκληρωθεί ο έλεγχος ERC και τώρα θα γίνει έλεγχος DRC με σκοπό να εντοπιστούν σφάλματα στο PCB Που δημιουργούμε. Εφόσον ο έλεγχος δεν επισημάνει κάποιο λάθος μπορούμε να προχωρήσουμε στην εξαγωγή των αρχείων Gerber. Επιλέγουμε File → Fabrication Outputs → Gerber Files και τα αρχεία δημιουργούνται αφού πρώτα ρυθμίσουμε όλα τα layers.

Μετά την ολοκλήρωση της παραπάνω διαδικασίας τα αρχεία που εξήχθησαν εμφανίζονται παρακάτω.



Σχήμα 5.20: Bottom Layer (αριστερά) και Top Layer (δεξιά) του PCB

Σχήμα 5.21: Τελικό Layout χωρίς Ground Plane

Τα σχήματα αυτά είναι διαφορετικές όψεις του σχεδίου που δημιουργήθηκε για την πλακέτα και συγκεκριμένα:

- Bottom Layer (μπλε φόντο)

Αφορά το κάτω στρώμα της πλακέτας. Οι κόκκινες γραμμές υποδεικνύουν τα tracks του Top Layer (κόκκινο φόντο) ενώ οι κίτρινες γραμμές και κείμενα είναι αυτά που θα τυπωθούν στην πλακέτα ώστε να τα βλέπει ο χρήστης.

- Top Layer (κόκκινο φόντο)

Αφορά το πάνω στρώμα της πλακέτας. Το κόκκινο χρώμα υποδεικνύει τις αγώγιμες περιοχές της πλακέτας, δηλαδή τα tracks και τα pads. Πρόκειται για τις διαδρομές από χαλκό που θα εκτυπωθούν στην επάνω πλευρά.

Οι μπλε γραμμές είναι τα tracks του Bottom Layer (μπλε φόντο) και φαίνονται για συγκριτικούς και σχεδιαστικούς σκοπούς

Τέλος, και σε αυτή την περίπτωση τα κίτρινα στοιχεία υποδηλώνουν οτιδήποτε θα τυπωθεί στην πλακέτα για να καθοδηγείται ο χρήστης.

Επιπλέον, σε αυτό το Layer φαίνονται τα pads που είναι τα σημεία όπου θα κολληθούν τα εξαρτήματα και τα vias που είναι μικρές τρύπες οι οποίες ενώνουν τα δύο επίπεδα (Layers) με χαλκό.

- Τελικό Layer χωρίς Ground Plane

Η έλλειψη Ground Plane σημαίνει ουσιαστικά ότι δεν εμφανίζεται η πλήρης κάλυψη του GND χαλκού στο επάνω και κάτω επίπεδο όπως φαινόταν προηγουμένως σε κόκκινο και μπλε χρώμα. Το Layer αυτό βοηθάει στο να φαίνονται καθαρά τα tracks, να εντοπίζονται πιθανά σφάλματα στις συνδέσεις αλλά και συγκρούσεις routing. Η απουσία του ενδέχεται να αυξήσει τον θόρυβο (EMI), να δημιουργήσει return paths, δηλαδή κακές επιστροφές ρεύματος αλλά και να δυσκολέψει τη διαδικασία του debugging επομένως θα πρέπει να παραμένει ενεργό στο τελικό PCB.

Και σε αυτή την περίπτωση με κίτρινο φαίνονται οι γραμμές πλαισίων για τα κουμπιά, τις ετικέτες και τους οδηγούς καθώς και οι ενδείξεις από τα πλήκτρα του keypad, η οθόνη, η γείωση (GND) και τα header pins.

Οι αγώγιμες διαδρομές που ενώνουν τα εξαρτήματα στο επάνω επίπεδο και στο κάτω επίπεδο είναι αντίστοιχα μπλε και κόκκινες γραμμές.

Τέλος, φαίνονται τα vias που είναι λευκά ή μαύρα σημεία με τρύπες και υποδεικνύουν τις τρύπες που συνδέουν τα δύο επίπεδα (Top/Bottom) και μεταφέρουν σήματα.

Το επόμενο στάδιο της υλοποίησης αποτελεί την τύπωση του PCB και την συγκόλληση των εξαρτημάτων στην πλακέτα. Το τελικό αποτέλεσμα αποτυπώνεται στο παρακάτω σχήμα.

Σχήμα 5.22: Τυπωμένη πλακέτα
Μπροστινή όψη (αριστερά), πίσω όψη (δεξιά)

Όπως φαίνεται, η μπροστινή όψη (Top Layer) περιλαμβάνει την οθόνη LCD και το πληκτρολόγιο ενώ στην πίσω όψη (Bottom Layer) διακρίνονται τα υπόλοιπα υλικά και εξαρτήματα καθώς και το barcode της συσκευής το οποίο απαιτείται για την παραγωγή κωδικού OTP μέσω του Token.

Στον παρακάτω πίνακα είναι καταγεγραμμένα όλα τα υλικά και εξαρτήματα που χρησιμοποιήθηκαν για την κατασκευή του A/D μετατροπέα, το κόστος καθενός από αυτά καθώς και Link το οποίο κατευθύνει στο επιλεγμένο προϊόν στη σελίδα Mouser. Το συνολικό κόστος των υλικών υπολογίζεται σε 30,65€.

Πίνακας 5.4: Υλικά για A/D Converter

Είδος	Volt	Περιγραφή	Ποσότητα	Συνολικό κόστος (€)	Mouser Links
27pF	10V	Multilayer Ceramic Capacitors MLCC - SMD/SMT 0402 27pF 10volts C0G 5%	**50**	2,4	http://gr.mouser.com/ProductDetail
NTR0202PLT1G		MOSFET -20V -400mA P-Channel	**20**	4,66	http://gr.mouser.com/ProductDetai
BAT-HLD-001		Battery Holders, Clips & Contacts Linx CR2032 Battery Holder	**10**	2,16	http://gr.mouser.com/ProductDetail
LT-K		Soldering Irons Weller Chisel Tip .047" x .73"	**2**	6,4	http://gr.mouser.com/ProductDetail
LPS6235-106MLB		Fixed Inductors Power Inductor 10000 uH 20% 0.095 A	**2**	2,38	http://gr.mouser.com/Search/ProductDetail
LT-H		Soldering Irons Weller Chisel Tip .031" x .43" Reach	2	6,56	http://gr.mouser.com/ProductDetail
10uF	35V	Multilayer Ceramic Capacitors MLCC - SMD/SMT 1206 10uF 35volts X7R 20%	10	3,25	http://gr.mouser.com/Search/ProductDetail
24ohm		Thick Film Resistors - SMD 1/10watt 24ohms 1%	100	0,8	http://gr.mouser.com/ProductDetail
LED		Standard LEDs - SMD	5	2,04	http://

RGB		120DG RD/GR/BL BK FC 480,540,624NM			gr.mouser.com/Search/ProductDetail

5.2 Έλεγχος λειτουργικότητας συσκευής

Με την ολοκλήρωση της κατασκευής, ακολουθεί ο έλεγχός της. Αρχικά φορτώνεται ο κώδικας στον μικροελεγκτή. Υπενθυμίζεται η διαδικασία που ακολουθείται με βάση τον κώδικα:

- Εκκίνηση συσκευής

Η πλακέτα τροφοδοτείται, η LCD ενεργοποιείται και το πρόγραμμα είναι σε αναμονή για έναν κωδικό $4^{ων}$ ψηφίων.

- Εισαγωγή αρχικού κωδικού

Ο χρήστης πληκτρολογεί τον κωδικό $4^{ων}$ ψηφίων και το σύστημα ελέγχει εάν ο κωδικός είναι σωστός.

- Εισαγωγή Verification Code

Ο χρήστης πληκτρολογεί έναν κωδικό $3^{ων}$ ψηφίων

- Αντιστοίχιση Verification Code με Security Code

Η συσκευή ελέγχει εάν ο 3ψήφιος αριθμός που πληκτρολογήθηκε υπάρχει στη μνήμη του PIC και εάν βρεθεί αντιστοιχία εμφανίζει τον αντίστοιχο κωδικό στην οθόνη.

Έτσι, αρχικά βλέπουμε ότι η πρόσβαση στο σύστημα γίνεται αρχικά με κάποιον Access κωδικό, στη συνέχεια πληκτρολογείται το Verification Code το οποίο αποτελεί την πρόκληση και τέλος λαμβάνουμε το Security Code το οποίο αποτελεί την απόκριση.

Στη συνέχεια, παρουσιάζονται τα παραπάνω βήματα χρησιμοποιώντας την υλοποιημένη συσκευή.

1. Αρχικά συνδέεται η συσκευή μέσω καλωδίου USB και λαμβάνει τροφοδοσία. Αμέσως παίρνει ρεύμα η LCD και εμφανίζεται το μήνυμα “A. CODE” όπου ο χρήστης θα πρέπει να πληκτρολογήσει τον αρχικό κωδικό.
2. Εφόσον ο χρήστης πληκτρολογήσει τον κωδικό «1571», πατάει στη συνέχεια το πλήκτρο «E» το οποίο αντιπροσωπεύει το “ENTER” και εισέρχεται στο σύστημα.
3. Έπειτα στην οθόνη εμφανίζεται το μήνυμα “Ver. Code” στο οποίο ο χρήστης πληκτρολογεί έναν 3ψηφιο κωδικό.
4. Στην συγκεκριμένη περίπτωση πληκτρολογείται ο αριθμός «000» και στην συνέχεια «E» (ENTER).
5. Το πρόγραμμα ελέγχει τον 3ψήφιο κωδικό και αφού ελέγξει στη μνήμη του μικροελεγκτή εμφανίζει στην οθόνη τον Sec. Code ο οποίος στην προκειμένη είναι ο «037037»

Σχήμα 5.23: Παραγωγή OTP – Βήμα 1ο

Σχήμα 5.24: Παραγωγή OTP – Βήμα 2ο

Σχήμα 5.25: Παραγωγή OTP – Βήμα 3ο

Σχήμα 5.26: Παραγωγή OTP – Βήμα 4ο

Σχήμα 5.27: Παραγωγή OTP – Βήμα 5ο

Ο κωδικός μια χρήσης έχει δημιουργηθεί και μπορεί πλέον να χρησιμοποιηθεί στο σύστημα ως επιπλέον παράγοντας για πρόσβαση.

Κεφάλαιο 6ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

Τα Security Tokens αποτελούν ένα σημαντικό μέτρο προστασίας στην ταυτοποίηση των χρηστών. Παρ' όλο που οι μορφές τους ολοένα και εξελίσσονται, βασικός στόχος είναι να ανταποκρίνονται στις επιθέσεις μην πιστοποιημένων χρηστών και να διαφυλάσσουν τα προσωπικά δεδομένα των εξουσιοδοτημένων.

Μέσω της μελέτης που έγινε για τη συγγραφή της παρούσας εργασίας, έγινε κατανοητό πως υπάρχουν διάφορες μορφές συσκευών Security Token και κάθε μία είναι σχεδιασμένη να εξυπηρετεί και κάποιον σκοπό. Μερικές είναι πιο ευάλωτες σε επιθέσεις ενώ κάποιες άλλες είναι πολύ δύσκολο να παραβιαστούν αλλά ίσως και αρκετά δύσκολο να υλοποιηθούν. Το Token που εξετάστηκε στην εργασία αποτελεί μια παλαιότερη μορφή τέτοιας συσκευής η οποία όμως χρησιμοποιείται μέχρι σήμερα καθώς αποτελεί μια πολύ ασφαλή συσκευή και δύσκολα παραβιάσιμη.

Η σχεδίαση και η υλοποίηση της συσκευής απαιτούσε γνώσεις γύρω από διάφορα υλικά και εξαρτήματα με κυριότερο τον μικροελεκγτή ο οποίος αποτελεί τον «εγκέφαλο» του συστήματος. Παράλληλα, ήταν απαραίτητες οι γνώσεις πάνω στον προγραμματισμό του μικροελεγκτή καθώς η λειτουργία του Token στηρίζεται στον κώδικα και στην αλγόριθμό που τον συνοδεύει. Πρόκειται για μια πρακτικά εύκολη κατασκευή η οποία αποδείχθηκε πλήρως λειτουργική και ανταποκρίθηκε στις απαιτήσεις της εργασίας.

Όπως αναφέρεται και στο αντίστοιχο κεφάλαιο, η συσκευή που υλοποιήθηκε αποτελεί ένα "Disconnected Token" και δεν επικοινωνεί ούτε με τη χρήση Bluetooth ούτε με τη χρήση Internet. Επομένως, θα μπορούσε η κατασκευή να βελτιωθεί προσθέτοντας το κατάλληλο υλικό και χρησιμοποιώντας το απαραίτητο επιπλέον λογισμικό.

Συμπερασματικά, η μελέτη και η υλοποίηση ενός Challenge-Response Security Token ολοκληρώθηκε επιτυχώς. Το σύστημα λειτουργεί ορθά, παράγοντας τους κωδικούς οι οποίοι προκύπτουν από τον κώδικα που συντάχθηκε για τις ανάγκες της εργασίας και μπορεί με μικρές αλλαγές ή/και προσθήκες να χρησιμοποιηθεί για την ταυτοποίηση ενός χρήστη σε απλά ψηφιακά συστήματα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed. Upper Saddle River, NJ: Pearson, 2016.

[2] A. Barabanov and D. Makrushin, "Authentication and authorization in microservice-based systems: survey of architecture patterns," *arXiv preprint*, Sep. 2020. arXiv

[3] E. De Cristofaro, H. Du, J. Freudiger, and G. Norcie, "A Comparative Usability Study of Two-Factor Authentication," *arXiv preprint*, Sep. 2013. arXiv

[4] M. Schink, A. Wagner, F. Unterstein, and J. Heyszl, "Security and Trust in Open Source Security Tokens," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2021, no. 3, pp. 176–201, Jul. 2021. tches.iacr.orgeprint.iacr.org

[5] SecurityScorecard, "What Is a Hardware Token? Comparing Authentication Methods," *SecurityScorecard Blog*, 2023. SecurityScorecard

[6] GateKeeper, "What is the difference between hard token and soft token?," GateKeeper Help, 2023. gatekeeperhelp.zendesk.com

[7] FIDO Alliance, "FIDO2: Web Authentication (WebAuthn) & CTAP," 2023. [Online]. Available: https://fidoalliance.org MDPIΒικιπαίδειαedtechmagazine.com

[8] Emmanouel T. Michailidis and Demosthenes Vouyioukas, "A Review on Software-Based and Hardware-Based Authentication Mechanisms for the Internet of Drones," *Drones*, vol. 6, no. 2, Feb. 2022. MDPI

[9] Yubico, "YubiKey 5 Series Technical Manual," Yubico AB, 2020. [Online]. Available: https://www.yubico.com

[10] IETF, "OCRA: OATH Challenge-Response Algorithm," *RFC 6287*, Jun. 2011. datatracker.ietf.org+1

[11] "Challenge–response authentication," *Wikipedia*, last updated recently. Βικιπαίδεια

[12] Thor Pedersen, "Challenge/Response Token," *ThorTeaches.com Cybersecurity Glossary*. thorteaches.com

[13] Cryptnox Docs, "Hardware Wallet V1.6 – Challenge-Response Process," 2025. Cryptnox Docs

[14] A. Yubetsu Codex, "Challenges in Implementing Token-Based Authentication Systems: A Review of Usability and Deployability Issues," *Yubetsu Codex*, Nov. 2023. codex.yubetsu.com

[15] D. Rahaeimehr and M. van Dijk, "Recursive Augmented Fernet (RAF) Token: Alleviating the Pain of Stolen Tokens," *arXiv preprint*, Dec. 2023. arXiv

[16] Wikipedia contributors, "HMAC-based one-time password (HOTP)," *Wikipedia*, last updated. Βικιπαίδεια

[17] Wikipedia contributors, "Software token," *Wikipedia*, last updated. Βικιπαίδεια

[18] D. Stallings, *Cryptography and Network Security*, όπως στο [1].

[19] Jeunese Payne, G. Jenkinson, F. Stajano, M. A. Sasse, and M. Spencer, "Responsibility and Tangible Security: Towards a Theory of User Acceptance of Security Tokens," *arXiv preprint*, May 2016. arXiv

[20] Yubico User Discussion (Reddit) Insights on Challenge-Response vs FIDO Security

[21] Microchip Technology Inc., *PIC18F1220/1320 Data Sheet: 18/20/28-Pin High-Performance, Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology*, Microchip Technology, DS39605F (Rev. F), Jul. 2007. datasheetspdf.com

[22] Microchip Technology Inc., *PIC18F1320-I/SO Datasheet*, Microchip Technology.

ΠΑΡΑΡΤΗΜΑ Α : ΚΩΔΙΚΑΣ

```
#include <P18F1320.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <delays.h>

//#############################################################################
// PIC18F1320 Configuration Bit Settings
#pragma config  OSC = INTIO2                    // Internal RC, OSC1 as RA7, OSC2 as RA6
#pragma config FSCM = OFF                       // No fail safe clock monitor
#pragma config IESO = OFF                       // Oscillator switchover disabled
#pragma config  PWRT = ON                       // Oscillator power up timer enabled (release version only)
#pragma config  BOR = OFF                       // Brown-out Reset enabled in hardware
#pragma config  BORV = 42                       // Brown-out Voltage at 4.2Volt
#pragma config WDT = OFF                        // Watchdog timer disabled
#pragma config MCLRE = ON                       // MCLR pin enabled
#pragma config LVP = OFF                        // Low voltage programming disabled
#pragma config  STVR = OFF                      // Stack overflow will cause reset
#pragma config CP0 = ON                         // Code Protection Block 0
#pragma config CP1 = ON                         // Code Protection Block 1
//#############################################################################

//
================================================================================
============================================================
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ DELAYS FUNCTIONS
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//
================================================================================
============================================================

//#define _XTAL_FREQ 20000000
```

```
//__delay_us(10);
//__delay_ms(10);

#define delay_us            Delay10TCYx                       // 10* [ 4 *(1/ 8MHz )] = 10 * 0.5 = 5uS

//8Mhz - C.M.=4*(1/8)=0.5us
void delay_ms( long int i )
{
   long int j;
   for( j = 0; j < i ; j++ )
   {
      Delay1KTCYx( 2 );                              // 2 * 1000 * 0.5us = 1000us  = 1ms - [ x = Time /
( ( 4 / Xtal_Freq ) * In_Cycles ) =>  x = 1000us / [ 4 *(1/ 8MHz )] * 1000 ) = 2
   }
}
//#########################################################################################
////
==================================================================================
=================================================================
////~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          LCD          FUNCTION
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
////
==================================================================================
=================================================================
////////////////////////////////////////////////////
//                                  //
// Configure your MCU. Any mode            //
// Pin Set digital mode. your code         //
//                                  //
////////////////////////////////////////////////////

#define LCD_RD7     LATAbits.LATA3      // D7
#define TRIS_RD7    TRISAbits.TRISA3

#define LCD_RD6     LATAbits.LATA2      // D6
```

```
#define TRIS_RD6    TRISAbits.TRISA2

#define LCD_RD5     LATAbits.LATA1      // D5
#define TRIS_RD5    TRISAbits.TRISA1

#define LCD_RD4     LATAbits.LATA0     // D4
#define TRIS_RD4    TRISAbits.TRISA0

#define LCD_EN      LATAbits.LATA7      // EN
#define TRIS_EN     TRISAbits.TRISA7

#define LCD_RS      LATAbits.LATA6      // RS
#define TRIS_RS     TRISAbits.TRISA6

////////////////////////////////////////////
//                                        //
// Available Lcd Commands                 //
//                                        //
////////////////////////////////////////////

#define     LCD_FIRST_ROW           128
#define     LCD_SECOND_ROW          192
#define     LCD_THIRD_ROW           148
#define     LCD_FOURTH_ROW          212
#define     LCD_CLEAR               1
#define     LCD_RETURN_HOME         2
#define     LCD_CURSOR_OFF          12
#define     LCD_UNDERLINE_ON        14
#define     LCD_BLINK_CURSOR_ON     15
#define     LCD_MOVE_CURSOR_LEFT    16
#define     LCD_MOVE_CURSOR_RIGHT   20
#define     LCD_TURN_OFF            0
#define     LCD_TURN_ON             8
```

```
#define     LCD_SHIFT_LEFT         24
#define     LCD_SHIFT_RIGHT        28

void Lcd_Init(void);
void Lcd_Out(unsigned char y, unsigned char x, const  char *buffer);
void Lcd_Out2(unsigned char y, unsigned char x, char *buffer);
void Lcd_Chr(unsigned char y, unsigned char x, char Chr);
void Lcd_Chr_CP(char Chr_CP);
void Lcd_Cmd(unsigned char Cmd);

void Delay_5us(void);
void Delay_5500us(void);

////////////////////////////////////////////
//                                        //
// Set delays, based on the               //
// frequency of a XTAL.                   //
//                                        //
////////////////////////////////////////////

void Delay_5us(void){
// Delay of 5us
// Cycles = (5us * 20MHz) / 4
// Cycles = 25
// Put 25 more
//Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
//Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
//Nop(); Nop(); Nop(); Nop(); Nop();
//Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
//Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
//Nop(); Nop(); Nop(); Nop(); Nop();

// Delay of 5us
```

```
// Cycles = (5us * 8MHz) / 4
// Cycles = 10
// Put 10 more
Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( );
Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( ); Nop( );
}

void Delay_5500us(void){
// Delay of 5.5ms
// Cycles = (5.5ms * 20MHz) / 4
// Cycles = 27,500 = 28,000
//Delay1KTCYx(28);

// Delay of 5.5ms
// Cycles = (5.5ms * 8MHz) / 4
// Cycles = 11
Delay1KTCYx( 11 );
}

void Lcd_Init(void){
unsigned char data;
TRIS_RD7 = 0; TRIS_RD6 = 0; TRIS_RD5 = 0; TRIS_RD4 = 0; TRIS_EN = 0; TRIS_RS = 0;
LCD_RD7 = 0; LCD_RD6 = 0; LCD_RD5 = 0; LCD_RD4 = 0; LCD_EN = 0; LCD_RS = 0;
Delay_5500us(); Delay_5500us(); Delay_5500us();
Delay_5500us(); Delay_5500us(); Delay_5500us();
for(data = 1; data < 4; data ++)
  {
  LCD_RD7 = 0; LCD_RD6 = 0; LCD_RD5 = 1; LCD_RD4 = 1; LCD_EN = 0; LCD_RS = 0;
  LCD_RD7 = 0; LCD_RD6 = 0; LCD_RD5 = 1; LCD_RD4 = 1; LCD_EN = 1; LCD_RS = 0;
  Delay_5us();
  LCD_RD7 = 0; LCD_RD6 = 0; LCD_RD5 = 1; LCD_RD4 = 1; LCD_EN = 0; LCD_RS = 0;
  Delay_5500us();
  }
```

```
LCD_RD7 = 0; LCD_RD6 = 0; LCD_RD5 = 1; LCD_RD4 = 0; LCD_EN = 0; LCD_RS = 0;
LCD_RD7 = 0; LCD_RD6 = 0; LCD_RD5 = 1; LCD_RD4 = 0; LCD_EN = 1; LCD_RS = 0;
Delay_5us();
LCD_RD7 = 0; LCD_RD6 = 0; LCD_RD5 = 1; LCD_RD4 = 0; LCD_EN = 0; LCD_RS = 0;
Delay_5500us();
data = 40; Lcd_Cmd(data);
data = 16; Lcd_Cmd(data);
data = 1;  Lcd_Cmd(data);
data = 15; Lcd_Cmd(data);
}

void Lcd_Out(unsigned char y, unsigned char x, const  char *buffer){
unsigned char data;
switch(y){
case 1: data = 127 + x; break;
case 2: data = 191 + x; break;
case 3: data = 147 + x; break;
case 4: data = 211 + x; break;
default: break;}
Lcd_Cmd(data);
while(*buffer)              // Write data to LCD up to null
    {
     Lcd_Chr_CP(*buffer);
     buffer++;              // Increment buffer
    }
return;
}

void Lcd_Out2(unsigned char y, unsigned char x, char *buffer){
unsigned char data;
switch(y){
case 1: data = 127 + x; break;
case 2: data = 191 + x; break;
```

```
case 3: data = 147 + x; break;
case 4: data = 211 + x; break;
default: break;}
Lcd_Cmd(data);
while(*buffer)              // Write data to LCD up to null
    {
     Lcd_Chr_CP(*buffer);
     buffer++;              // Increment buffer
    }
return;
}

void Lcd_Chr(unsigned char y, unsigned char x, char Chr){
unsigned char data;
switch(y){
case 1: data = 127 + x; break;
case 2: data = 191 + x; break;
case 3: data = 147 + x; break;
case 4: data = 211 + x; break;
default: break;}
Lcd_Cmd(data);
Lcd_Chr_CP(Chr);
}

void Lcd_Chr_CP(char Chr_CP){
LCD_EN = 0; LCD_RS = 1;
LCD_RD7 = (Chr_CP & 0b10000000)>>7; LCD_RD6 = (Chr_CP & 0b01000000)>>6;
LCD_RD5 = (Chr_CP & 0b00100000)>>5; LCD_RD4 = (Chr_CP & 0b00010000)>>4;
Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
LCD_EN = 1; Delay_5us(); LCD_EN = 0;

LCD_RD7 = (Chr_CP & 0b00001000)>>3; LCD_RD6 = (Chr_CP & 0b00000100)>>2;
LCD_RD5 = (Chr_CP & 0b00000010)>>1; LCD_RD4 = (Chr_CP & 0b00000001);
```

```
Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
LCD_EN = 1; Delay_5us(); LCD_EN = 0;
Delay_5us(); Delay_5500us();
}

void Lcd_Cmd(unsigned char Cmd){
LCD_EN = 0; LCD_RS = 0;
LCD_RD7 = (Cmd & 0b10000000)>>7; LCD_RD6 = (Cmd & 0b01000000)>>6;
LCD_RD5 = (Cmd & 0b00100000)>>5; LCD_RD4 = (Cmd & 0b00010000)>>4;
Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
LCD_EN = 1; Delay_5us(); LCD_EN = 0;

LCD_RD7 = (Cmd & 0b00001000)>>3; LCD_RD6 = (Cmd & 0b00000100)>>2;
LCD_RD5 = (Cmd & 0b00000010)>>1; LCD_RD4 = (Cmd & 0b00000001);
Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop(); Nop();
LCD_EN = 1; Delay_5us(); LCD_EN = 0;
Delay_5500us();//Delay_5us();
}

//Lcd_Out( 1, 1, "Hello" );

//sprintf(buffer1,"%3d",x);      //Right aligned text in three positions
//sprintf(buffer1,"%03d",i);   //Right aligned text and adding zeros if necessary
//sprintf(buffer1,"%d   ",i);  //Left  aligned text
//##################################################################################




////
=======================================================================================================
==================================================================================

////~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~     KEYPAD     3x4     FUNCTION
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
////
================================================================================
============================================================

#define R1      PORTBbits.RB0                   // Keypad - Row 1
#define R2      PORTBbits.RB1                   // Keypad - Row 2
#define R3      PORTBbits.RB2                   // Keypad - Row 3
#define R4      PORTBbits.RB3                   // Keypad - Row 4
#define C1      PORTBbits.RB4                   // Keypad - Column 1
#define C2      PORTBbits.RB5                   // Keypad - Column 2
#define C3      PORTBbits.RB6                   // Keypad - Column 3



//------------------------------------------------------------------------------------------------------------------
----------------------------------
int keypad( void )
{
   unsigned int key;

   C1 = 0;
   C2 = 0;
   C3 = 0;
   delay_ms( 1 );



//------------------------------------------------------------------------------------------------------------------
----------------------------------
// Debounce
   while( R1 == 1 && R2 == 1 && R3 == 1 && R4 == 1 ) { key = 0x10; }
   delay_ms( 20 ); // or 10ms
   while( R1 == 1 && R2 == 1 && R3 == 1 && R4 == 1 ) { key = 0x10; }
//------------------------------------------------------------------------------------------------------------------
----------------------------------
```

```
C1 = 0;
C2 = 1;
C3 = 1;
delay_ms( 1 );  // Delay for stabilization
if( R1 == 0 && C1 == 0 && C2 == 1 && C3 == 1 ) key = 0x01;
if( R2 == 0 && C1 == 0 && C2 == 1 && C3 == 1 ) key = 0x04;
if( R3 == 0 && C1 == 0 && C2 == 1 && C3 == 1 ) key = 0x07;
if( R4 == 0 && C1 == 0 && C2 == 1 && C3 == 1 ) key = 0x0A;

C1 = 1;
C2 = 0;
C3 = 1;
delay_ms( 1 );  // Delay for stabilization
if( R1 == 0 && C1 == 1 && C2 == 0 && C3 == 1 ) key = 0x02;
if( R2 == 0 && C1 == 1 && C2 == 0 && C3 == 1 ) key = 0x05;
if( R3 == 0 && C1 == 1 && C2 == 0 && C3 == 1 ) key = 0x08;
if( R4 == 0 && C1 == 1 && C2 == 0 && C3 == 1 ) key = 0x00;

C1 = 1;
C2 = 1;
C3 = 0;
delay_ms( 1 );  // Delay for stabilization
if( R1 == 0 && C1 == 1 && C2 == 1 && C3 == 0 ) key = 0x03;
if( R2 == 0 && C1 == 1 && C2 == 1 && C3 == 0 ) key = 0x06;
if( R3 == 0 && C1 == 1 && C2 == 1 && C3 == 0 ) key = 0x09;
if( R4 == 0 && C1 == 1 && C2 == 1 && C3 == 0 ) key = 0x0B;

C1 = 0;
C2 = 0;
C3 = 0;
```

```
//-------------------------------------------------------------------------------------------------------------------------
-----------------------------------
// Debounce
   while( R1 == 0 || R2 == 0 || R3 == 0 || R4 == 0 ) { }
   delay_ms( 20 ); // or 10ms
   while( R1 == 0 || R2 == 0 || R3 == 0 || R4 == 0 ) { }
//-------------------------------------------------------------------------------------------------------------------------
-----------------------------------
   return key;
 }
//##################################################################################

//
===============================================================================
==============================================================
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ MAIN PROGRAM
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
//
===============================================================================
==============================================================
__EEPROM_DATA(0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00);      //The macro must be given
blocks of 8 bytes to write each time it is called,

#define LCD_POWER      PORTAbits.RA4           // Keypad - Row 1
#define LBO            PORTBbits.RB7           // Low Battery Input { MAX1675EUA }
#define BARCODE 82                             // The last two digits from the barcode of the
device  { barcode -->  469-3217-82 }
#define DIGIT_1  1
#define DIGIT_2  5
#define DIGIT_3  7
#define DIGIT_4  1

void main( void )
{
   unsigned int OTP[  ] = { 0x89, 0x42, 0x8a, 0x71, 0x2a, 0x31, 0xb6, 0x28, 0xb8, 0x0b, 0x16, 0x83,
0xea, 0x3d, 0xa2, 0x94, 0xa5, 0x07,
```

```
                    0xc7, 0x02, 0x22, 0xb3, 0x87, 0xd6, 0x09, 0x10, 0x2c, 0x44, 0x3a, 0xe3, 0xf9,
0x79, 0x5a, 0x1f, 0x9e, 0x45 };

   char lcd_char[20];
   unsigned int number, units, dozens, hundreds, enter;
   unsigned int p, first_number,  second_number, swap_digit, lsb,msb;
   unsigned int dig1, dig2, dig3, dig4, dig5, attempt_counter = 0;


//#############################################################################
// Microcontroller initializations
   OSCCON = 0b01110000;
   delay_ms( 1000 );                    // Delay for stabilization

   ADCON1 = 0x7F;                       // Disable Analog inputs
   TRISA = 0x20;                        // Set PORTA
   TRISB = 0x8F;                        // Set PORTB
   delay_ms( 50 );                      // Delay for stabilization

   LCD_POWER = 0;
   delay_ms( 50 );                      // Delay for stabilization
//#############################################################################


//#############################################################################
// LCD initializations
   Lcd_Init( );
   Lcd_Cmd( LCD_CLEAR );
   Lcd_Cmd( LCD_CURSOR_OFF );
//#############################################################################


//#############################################################################
```

```
// Check battery condition
   if( LBO == 0 )
   {
      Lcd_Out( 1, 1, "Low" );
      Lcd_Out( 2, 1, "Battery" );
      delay_ms( 3000 );
   }

   Lcd_Cmd( LCD_CLEAR );
//#############################################################################



//#############################################################################
// Access Code
   Start :
   attempt_counter = eeprom_read( 0x00 );

   while( attempt_counter >= 3)
   {
      if( attempt_counter == 3)
      {
      Lcd_Cmd( LCD_CLEAR );
      Lcd_Out( 1, 1, "Locked" );
      attempt_counter++;
      }
   }

   if( attempt_counter == 2 )
   {
      Lcd_Cmd( LCD_CLEAR );
      Lcd_Out( 1, 1, "Last" );
      Lcd_Out( 2, 1, "Attempt" );
      delay_ms( 1000 );
```

```
    while( keypad( ) == 0x0A ) { }
  }

  Lcd_Cmd( LCD_CLEAR );
  Lcd_Out( 1, 1, "A.Code" );

//-------------------------------------------------------------------------------------------------------------------
// Get access password
  Digit_1 :
  dig1 = keypad( );
  if( dig1 == 0x0A ) goto Check_Digits;
  if( dig1 == 0x0B ) goto Digit_1;
  Lcd_Out( 2, 1, "*" );

  Digit_2 :
  dig2 = keypad( );
  if( dig2 == 0x0A ) goto Check_Digits;
  if( dig2 == 0x0B )
  {
    Lcd_Out( 2, 1, " " );
    goto Digit_1;
  }
  Lcd_Out( 2, 2, "*" );

  Digit_3 :
  dig3 = keypad( );
  if( dig3 == 0x0A ) goto Check_Digits;
  if( dig3 == 0x0B )
  {
    Lcd_Out( 2, 2, " " );
    goto Digit_2;
```

```
    }
    Lcd_Out( 2, 3, "*" );

    Digit_4 :
    dig4 = keypad( );
    if( dig4 == 0x0A ) goto Check_Digits;
    if( dig4 == 0x0B )
    {
       Lcd_Out( 2, 3, " " );
       goto Digit_3;
    }
    Lcd_Out( 2, 4, "*" );



    dig5 = keypad( );
    if( dig5 == 0x0B )
    {
       Lcd_Out( 2, 4, " " );
       goto Digit_4;
    }

    while( dig5 != 0x0A )
    {
       dig5 = keypad( );
       if( dig5 == 0x0B )
       {
          Lcd_Out( 2, 4, " " );
          goto Digit_4;
       }
    }
//-------------------------------------------------------------------------------------------------------------------
-----------------------------------
    Check_Digits :
```

```
    attempt_counter++;
    eeprom_write( 0x00, attempt_counter );


    if( dig1 != DIGIT_1 ) goto Start;
    if( dig2 != DIGIT_2) goto Start;
    if( dig3 != DIGIT_3 ) goto Start;
    if( dig4 != DIGIT_4 ) goto Start;


    attempt_counter = 0;
    eeprom_write( 0x00, attempt_counter );
//#############################################################################################


//#############################################################################################
// Swap the last two digits from the barcode of the device
    lsb = BARCODE & 0xF0;
    lsb = lsb >> 4;

    msb = BARCODE & 0x0F;
    msb = msb << 4;

    swap_digit= msb | lsb;                          // swap_digit --> Swapping BARCODE number
//#############################################################################################


//#############################################################################################
// Display the inserting code & Ccreating three digit number
    Lcd_Cmd( LCD_CLEAR );
    Lcd_Out( 1, 1, "Ver.Code" );
```

```
//-------------------------------------------------------------------------------------------------------------------
-----------------------------------
// Get three digits
   Get_Hundrends :
   hundreds = keypad( );
   if( hundreds == 0x0A ) goto Create_3_digits;
   if( hundreds == 0x0B )
   {
      Lcd_Out( 2, 1, " " );
      goto Get_Hundrends;
   }
   sprintf(lcd_char,"%d", hundreds );                    //Second line, First place
   Lcd_Out2(2, 1, lcd_char);

   Get_Dozens :
   dozens  = keypad( );
   if( dozens == 0x0A ) goto Create_3_digits;
   if( dozens == 0x0B )
   {
      Lcd_Out( 2, 1, " " );
      goto Get_Hundrends;
   }
   sprintf(lcd_char,"%d", dozens );                      //Second line, First place
   Lcd_Out2(2, 2, lcd_char);

   Get_Units :
   units  = keypad( );
   if( units == 0x0A ) goto Create_3_digits;
   if( units == 0x0B )
   {
      Lcd_Out( 2, 2, " " );
      goto Get_Dozens;
   }
```

```
   sprintf(lcd_char,"%d", units );                    //Second line, First place
   Lcd_Out2(2, 3, lcd_char);



   enter = keypad( );
   if( enter == 0x0B )
   {
      Lcd_Out( 2, 3, " " );
      goto Get_Units;
   }

   while( enter != 0x0A )
   {
      enter = keypad( );
      if( enter == 0x0B )
      {
         Lcd_Out( 2, 3, " " );
         goto Get_Units;
      }
   }
//-------------------------------------------------------------------------------------------------------------------
---------------------------------
   Create_3_digits :
   hundreds =  100 * hundreds;
   dozens  = 10 * dozens;
   units  = 1 * units;
   number = hundreds + dozens + units;
//#############################################################################################



//#############################################################################################
   p = number ^ BARCODE;                              // Place of the table's elements     //barcode -->
469-3217-82//
```

```
   // Create two numbers with three digits
   first_number = OTP[ p ] ^ swap_digit;                // swap_digit --> Swapping BARCODE number
   second_number = OTP[ p ] | swap_digit;               // swap_digit --> Swapping BARCODE number
//#############################################################################################


//#############################################################################################
// Display 6 digit code
   Lcd_Cmd(LCD_CLEAR);

   Lcd_Out( 1, 1, "Sec.Code" );
   sprintf(lcd_char,"%03d%03d", first_number, second_number );      //Second line, First place
   Lcd_Out2(2, 1, lcd_char);
   delay_ms( 60000 );                                   // 1 minute Delay 60000

   Lcd_Cmd(LCD_CLEAR);
   TRISA = 0xff;                                        // Set PORTA as input
//#############################################################################################



   while ( 1 ) { }

}
```

ΠΑΡΑΡΤΗΜΑ Β : ΚΩΔΙΚΑΣ LFSR

```
# include <stdint.h>

# include <stdio.h>


int main( int argc, char *argv[ ] )

{

   uint32_t xor_out, lfsr, AFM, BARCODE, BARCODE_two_last_digits;   //
AFM=123456789, BARCODE=469321782;

   uint16_t limit = 0;

   uint16_t  period = 0, number = 0, lfsr_out_bit = 0;

   uint32_t two_last_digits[ ] = { 4, 6, 9, 3, 2, 1, 7, 8, 2 };

   uint16_t  i = 0, place = 0, temp_table [ 36 ], lsb, msb, first_number, second_number,
swap_digit, site_number;


   FILE *OTP;

   OTP=fopen( "OTP.txt", "w" );  // "w" Open for writing and create the file if it does not
exist. If the file exists then make it blank.


   printf( "Customer's VAT Number: " );

   scanf( "%u", &AFM );

   printf("\nToken's BARCODE: " );

   scanf( "%u", &BARCODE );


   lfsr = AFM ^ BARCODE;


   fprintf( OTP, "One Time Passwords:\n" );


//
##########################################################################
##########################################################################
########

   while( limit < 36 )
```

```
   {
//-------------------------------------------------------------------------------------------------------------------
-------------------------------------------
      // Characteristic Polynomial: x^32 + x^30 + x^28 + x^27 + x^24 + x^19 + x^14 + 1
      // Taps: 32 30 28 27 24 19 14
      xor_out  = ( ( lfsr >> 0 ) ^ ( lfsr >> 2 ) ^ ( lfsr >> 4 ) ^ ( lfsr >> 5 ) ^ ( lfsr >> 8 ) ^ ( lfsr
>> 13 ) ^ ( lfsr >> 18 ) ) & 0x00000001;
      lfsr =  ( lfsr >> 1 ) | ( xor_out << 31 );

      period++;
//-------------------------------------------------------------------------------------------------------------------
-------------------------------------------

      lfsr_out_bit = lfsr & 1;
      number = number | lfsr_out_bit;

      if( period < 8 ) number = number << 1;

      if( period == 8 )
      {
         fprintf( OTP, "0x%02x", number );
         if( limit <  35 ) fprintf( OTP, ", ", number );
         if( limit == 17 ) fprintf( OTP, "\n", number );

         temp_table[ i ] = number;
         i++;

         limit++;
         number = 0;
         period = 0;
      }
```

```
    }
//
################################################################################
################################################################################
########

    fprintf( OTP, "\n\n\n\n" );


//
################################################################################
################################################################################
########
    fprintf( OTP, "One Time Passwords - Vertically printed:\n" );


    for( i = 0; i <36; i++ )
    {


        fprintf( OTP, "Table place: %d  --->  Element: *0x%02x*\n", i, temp_table[ i ] );
    }
//
################################################################################
################################################################################
########


    fprintf( OTP, "\n\n\n" );


//
################################################################################
################################################################################
########
    fprintf( OTP, "Site input and output codes:\n" );


    BARCODE_two_last_digits = BARCODE % 100;


    for( i = 0; i <36; i++ )
```

```
   {

      site_number = i ^ BARCODE_two_last_digits;
 //-------------------------------------------------------------------------------------------------------------
-----------------------------------------------
// Swap the last two digits from the barcode of the device
      lsb = 82 & 0xF0;
      lsb = lsb >> 4;

      msb = 82 & 0x0F;
      msb = msb << 4;

      swap_digit= msb | lsb;   // swap_digit --> Swapping BARCODE number
//-------------------------------------------------------------------------------------------------------------
-----------------------------------------------

//-------------------------------------------------------------------------------------------------------------
-----------------------------------------------
// Create two numbers with three digits
      first_number = temp_table[ i ] ^ swap_digit;
      second_number = temp_table[ i ] | swap_digit;
//-------------------------------------------------------------------------------------------------------------
-----------------------------------------------

      fprintf( OTP, "%d --> Site output: *%03d*  --->  Site input: *%03d%03d*\n", i,
site_number, first_number, second_number );
   }
//
#############################################################################
#############################################################################
########

   fclose( OTP );  // Close file OTP
```

```
    printf( "\n\nSuccessful generation of codes.\n\n\n\n\n" );
        system("pause");
    return 0;
}
```

ΠΑΡΑΡΤΗΜΑ Γ: ΑΡΧΕΙΟ OTP.txt

One Time Passwords:

0x89, 0x42, 0x8a, 0x71, 0x2a, 0x31, 0xb6, 0x28, 0xb8, 0x0b, 0x16, 0x83, 0xea, 0x3d, 0xa2, 0x94, 0xa5, 0x07,

0xc7, 0x02, 0x22, 0xb3, 0x87, 0xd6, 0x09, 0x10, 0x2c, 0x44, 0x3a, 0xe3, 0xf9, 0x79, 0x5a, 0x1f, 0x9e, 0x45

One Time Passwords - Vertically printed:

Table place: 0 ---> Element: *0x89*

Table place: 1 ---> Element: *0x42*

Table place: 2 ---> Element: *0x8a*

Table place: 3 ---> Element: *0x71*

Table place: 4 ---> Element: *0x2a*

Table place: 5 ---> Element: *0x31*

Table place: 6 ---> Element: *0xb6*

Table place: 7 ---> Element: *0x28*

Table place: 8 ---> Element: *0xb8*

Table place: 9 ---> Element: *0x0b*

Table place: 10 ---> Element: *0x16*

Table place: 11 ---> Element: *0x83*

Table place: 12 ---> Element: *0xea*

Table place: 13 ---> Element: *0x3d*

Table place: 14 ---> Element: *0xa2*

Table place: 15 ---> Element: *0x94*

Table place: 16 ---> Element: *0xa5*

Table place: 17 ---> Element: *0x07*

Table place: 18 ---> Element: *0xc7*

Table place: 19 ---> Element: *0x02*

Table place: 20 ---> Element: *0x22*

Table place: 21 ---> Element: *0xb3*

Table place: 22 ---> Element: *0x87*

Table place: 23 ---> Element: *0xd6*

Table place: 24 ---> Element: *0x09*

Table place: 25 ---> Element: *0x10*

Table place: 26 ---> Element: *0x2c*

Table place: 27 ---> Element: *0x44*

Table place: 28 ---> Element: *0x3a*

Table place: 29 ---> Element: *0xe3*

Table place: 30 ---> Element: *0xf9*

Table place: 31 ---> Element: *0x79*

Table place: 32 ---> Element: *0x5a*

Table place: 33 ---> Element: *0x1f*

Table place: 34 ---> Element: *0x9e*

Table place: 35 ---> Element: *0x45*

Site input and output codes:

0 --> Site output: *082* ---> Site input: *172173*

1 --> Site output: *083* ---> Site input: *103103*

2 --> Site output: *080* ---> Site input: *175175*

3 --> Site output: *081* ---> Site input: *084117*

4 --> Site output: *086* ---> Site input: *015047*

5 --> Site output: *087* ---> Site input: *020053*

6 --> Site output: *084* ---> Site input: *147183*

7 --> Site output: *085* ---> Site input: *013045*

8 --> Site output: *090* ---> Site input: *157189*

9 --> Site output: *091* ---> Site input: *046047*

10 --> Site output: *088* ---> Site input: *051055*

11 --> Site output: *089* ---> Site input: *166167*

12 --> Site output: *094* ---> Site input: *207239*

13 --> Site output: *095* ---> Site input: *024061*

14 --> Site output: *092* ---> Site input: *135167*

15 --> Site output: *093* ---> Site input: *177181*

16 --> Site output: *066* ---> Site input: *128165*

17 --> Site output: *067* ---> Site input: *034039*

18 --> Site output: *064* ---> Site input: *226231*

19 --> Site output: *065* ---> Site input: *039039*

20 --> Site output: *070* ---> Site input: *007039*

21 --> Site output: *071* ---> Site input: *150183*

22 --> Site output: *068* ---> Site input: *162167*

23 --> Site output: *069* ---> Site input: *243247*

24 --> Site output: *074* ---> Site input: *044045*

25 --> Site output: *075* ---> Site input: *053053*

26 --> Site output: *072* ---> Site input: *009045*

27 --> Site output: *073* ---> Site input: *097101*

28 --> Site output: *078* ---> Site input: *031063*

29 --> Site output: *079* ---> Site input: *198231*

30 --> Site output: *076* ---> Site input: *220253*

31 --> Site output: *077* ---> Site input: *092125*

32 --> Site output: *114* ---> Site input: *127127*

33 --> Site output: *115* ---> Site input: *058063*

34 --> Site output: *112* ---> Site input: *187191*

35 --> Site output: *113* ---> Site input: *096101*