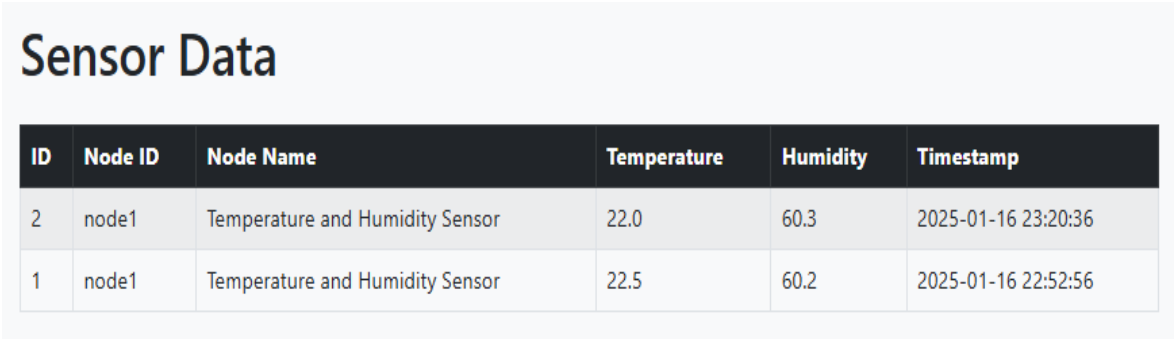
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ασφάλεια στην επικοινωνία συσκευών στο IoT»



Sensor Data

ID	Node ID	Node Name	Temperature	Humidity	Timestamp
2	node1	Temperature and Humidity Sensor	22.0	60.3	2025-01-16 23:20:36
1	node1	Temperature and Humidity Sensor	22.5	60.2	2025-01-16 22:52:56

Φοιτητής

ΧΡΗΣΤΟΣ ΑΜΠΕΡΙΑΔΗΣ **512005**

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Φεβρουάριος 2025

Ασφάλεια στην επικοινωνία συσκευών στο IoT

Κωδικός: 24300
Φοιτητής: Χρήστος Αμπεριάδης
Εισηγητής: Δρ Κυριάκος Τσιακμάκης
Ημερομηνία ανάληψης Π.Ε. 31-10-2024
Ημερομηνία περάτωσης Π.Ε. 20-01-2025

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 πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής 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.

Περίληψη

Η ασφάλεια στην επικοινωνία συσκευών στο IoT είναι ζωτικής σημασίας για την προστασία των δεδομένων και τη διατήρηση της ακεραιότητας των δικτύων. Η εργασία αυτή εστιάζει στην ανάπτυξη ενός συστήματος ασφαλούς επικοινωνίας, εφαρμόζοντας κρυπτογράφηση και μηχανισμούς πιστοποίησης για την εξασφάλιση της ταυτότητας των συσκευών. Επιπλέον, θα υλοποιηθούν μέθοδοι για την ανίχνευση κακόβουλων δραστηριοτήτων, ενισχύοντας την προστασία από επιθέσεις. Συγκεκριμένα, το έργο περιλαμβάνει την κρυπτογραφημένη επικοινωνία μεταξύ δύο κόμβων (PC/Raspberry Pi) χρησιμοποιώντας Python και τη διασύνδεση ενός ESP32 με έναν κόμβο PC/Raspberry Pi μέσω C/C++ και Python αντίστοιχα. Με αυτά τα βήματα, στοχεύεται η ανάπτυξη ενός συστήματος που προσφέρει υψηλή ασφάλεια σε εφαρμογές IoT.

« Security in IoT Device Communication»

Abstract

Security in IoT device communication is crucial for data protection and maintaining network integrity. This work focuses on developing a secure communication system, implementing encryption and authentication mechanisms to ensure device identity. In addition, methods for detecting malicious activities will be implemented, enhancing protection against attacks. Specifically, the project includes encrypted communication between two nodes (PC/Raspberry Pi) using Python and the interfacing of an ESP32 with a PC/Raspberry Pi node via C/C++ and Python respectively. With these steps, the goal is to develop a system that offers high security in IoT applications.

Ευχαριστίες

Θέλω να ευχαριστήσω τον επιβλέπων κ. Τσιακμάκη Κυριάκο για τη συνεχή παιδαγωγική καθοδήγηση, οδηγίες του και τη βοήθεια του στους κώδικες.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η ασφάλεια στην επικοινωνία συσκευών του Διαδικτύου των Πραγμάτων (IoT) αποτελεί έναν από τους σημαντικότερους παράγοντες για την υλοποίηση αξιόπιστων και ασφαλών συστημάτων. Η εκρηκτική αύξηση του αριθμού των IoT συσκευών και η εξάρτηση από αυτές σε κρίσιμους τομείς όπως η βιομηχανία, η υγεία, και τα έξυπνα σπίτια, καθιστά επιτακτική την ανάγκη για μέτρα που διασφαλίζουν την ακεραιότητα, την εμπιστευτικότητα και την ασφάλεια των δεδομένων που διακινούνται.

Το έργο αυτό επικεντρώνεται στην ανάπτυξη ενός ολοκληρωμένου συστήματος ασφαλούς επικοινωνίας μεταξύ IoT συσκευών, το οποίο περιλαμβάνει απομακρυσμένους κόμβους (Senders) και έναν κεντρικό κόμβο (Receiver). Οι Senders συλλέγουν δεδομένα από αισθητήρες, όπως θερμοκρασία και υγρασία, και τα αποστέλλουν στον Receiver για επεξεργασία και αποθήκευση. Παράλληλα, ο Receiver μπορεί να στέλνει εντολές πίσω στους Senders για τη διαχείριση συσκευών, όπως ενεργοποίηση ή απενεργοποίηση relay.

Το σύστημα βασίζεται σε ένα υβριδικό μοντέλο κρυπτογράφησης, που συνδυάζει την ασύμμετρη κρυπτογράφηση RSA για την ασφαλή ανταλλαγή κλειδιών και τη συμμετρική κρυπτογράφηση AES για την προστασία των δεδομένων. Επιπλέον, περιλαμβάνει μηχανισμούς επαλήθευσης ταυτότητας για τον έλεγχο των κόμβων και συστήματα ανίχνευσης κακόβουλων δραστηριοτήτων, όπως μη εξουσιοδοτημένες προσπάθειες σύνδεσης. Όλες οι δραστηριότητες καταγράφονται σε βάση δεδομένων MySQL, ενώ οι χρήστες έχουν πρόσβαση στα δεδομένα μέσω μιας web διεπαφής, που παρέχει δυνατότητες παρακολούθησης και ανάλυσης.

Το έργο αυτό στοχεύει στην ανάδειξη ενός συστήματος που προσφέρει υψηλά επίπεδα ασφάλειας, επεκτασιμότητας και ευελιξίας, ανταποκρινόμενο στις ανάγκες των σύγχρονων IoT εφαρμογών. Στα επόμενα κεφάλαια, αναλύονται τα επιμέρους τεχνικά χαρακτηριστικά και οι λειτουργίες του συστήματος, παρέχοντας μια ολοκληρωμένη εικόνα των τεχνολογιών που χρησιμοποιήθηκαν, των προκλήσεων που αντιμετωπίστηκαν και των αποτελεσμάτων που επιτεύχθηκαν.

Ο κύριος στόχος της παρούσας εργασίας είναι η ανάπτυξη ενός ασφαλούς συστήματος επικοινωνίας για IoT συσκευές, το οποίο διασφαλίζει την εμπιστευτικότητα, την ακεραιότητα και την ασφάλεια των δεδομένων που διακινούνται. Το σύστημα περιλαμβάνει απομακρυσμένους κόμβους (Senders) που συλλέγουν δεδομένα από αισθητήρες και έναν κεντρικό κόμβο (Receiver) που επεξεργάζεται τα δεδομένα και αποθηκεύει τις σχετικές πληροφορίες.

Οι επιμέρους στόχοι είναι:

1. **Ασφαλής επικοινωνία**: Υλοποίηση ενός υβριδικού μοντέλου κρυπτογράφησης, που χρησιμοποιεί RSA για την ασφαλή ανταλλαγή κλειδιών και AES για την κρυπτογράφηση των δεδομένων.
2. **Επαλήθευση ταυτότητας**: Διασφάλιση ότι μόνο εξουσιοδοτημένοι κόμβοι μπορούν να επικοινωνούν με το σύστημα, μέσω μηχανισμών επαλήθευσης ταυτότητας.
3. **Καταγραφή ύποπτων δραστηριοτήτων**: Εντοπισμός και καταγραφή μη εξουσιοδοτημένων ενεργειών, όπως απόπειρες σύνδεσης από άγνωστους κόμβους ή IP διευθύνσεις.
4. **Αποθήκευση δεδομένων**: Ανάπτυξη μιας ασφαλούς βάσης δεδομένων για την καταγραφή δεδομένων αισθητήρων, ύποπτων δραστηριοτήτων και επιβεβαιωμένων συνδέσεων.
5. **Παρακολούθηση**: Δημιουργία μιας web διεπαφής που επιτρέπει στους διαχειριστές να παρακολουθούν τις δραστηριότητες του συστήματος και να διαχειρίζονται τα δεδομένα.

Η παρούσα εργασία συνεισφέρει στους ακόλουθους τομείς:

- **Ασφαλής επικοινωνία IoT συσκευών**: Αναδεικνύει τη σημασία της χρήσης υβριδικών μοντέλων κρυπτογράφησης για την προστασία δεδομένων σε δίκτυα IoT.
- **Πρακτική υλοποίηση ασφαλών συστημάτων**: Παρέχει έναν οδηγό για την ανάπτυξη και εφαρμογή ενός ασφαλούς συστήματος επικοινωνίας που περιλαμβάνει αποθήκευση και προβολή δεδομένων.
- **Ανίχνευση κακόβουλων δραστηριοτήτων**: Ενσωματώνει μηχανισμούς παρακολούθησης και εντοπισμού μη εξουσιοδοτημένων ενεργειών, ενισχύοντας την ασφάλεια του συστήματος.

1.2 Δομή της εργασίας

Στο πρώτο κεφάλαιο παρουσιάζεται μια εισαγωγή στην εργασία, περιγράφονται ο κύριος και οι επιμέρους στόχοι της, καθώς και η συνεισφορά της.

Στο δεύτερο κεφάλαιο αναλύεται η προσέγγιση παρόμοιων συστημάτων IoT

Στο τρίτο κεφάλαιο παρουσιάζονται οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη του συστήματος, όπως η κρυπτογράφηση RSA/AES, η πλατφόρμα ανάπτυξης του ESP32, η γλώσσα Python και η βάση δεδομένων MySQL.

Στο τέταρτο κεφάλαιο περιγράφεται η υλοποίηση του συστήματος με αναλυτική παρουσίαση του κώδικα, διαγράμματα ροής δεδομένων και Εικόνες. Περιλαμβάνεται επίσης η ανάλυση της βάσης δεδομένων και των δοκιμών που πραγματοποιήθηκαν για την αξιολόγηση της λειτουργίας του συστήματος.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας, συνοψίζοντας τη συνεισφορά της και την αποτελεσματικότητα του συστήματος. Παράλληλα, προτείνονται μελλοντικές επεκτάσεις και βελτιώσεις που θα μπορούσαν να γίνουν. Στο τέλος της εργασίας περιλαμβάνεται ένα παράρτημα με τους βασικούς κώδικες που χρησιμοποιήθηκαν στην υλοποίηση.

Κεφάλαιο 2ο: Συστήματα IoT

2.1 Εισαγωγή

Το Διαδίκτυο των Πραγμάτων (Internet of Things - IoT) αναφέρεται στη σύνδεση φυσικών αντικειμένων στο διαδίκτυο, επιτρέποντάς τους να συλλέγουν και να ανταλλάσσουν δεδομένα. Αυτή η διασύνδεση επιτυγχάνεται μέσω ενσωματωμένων συστημάτων, αισθητήρων και λογισμικού, δημιουργώντας ένα δίκτυο έξυπνων συσκευών που επικοινωνούν μεταξύ τους και με τους χρήστες. Η ανάπτυξη του IoT έχει επιφέρει σημαντικές αλλαγές σε διάφορους τομείς, όπως η υγεία, η βιομηχανία, οι μεταφορές και τα έξυπνα σπίτια, βελτιώνοντας την αποδοτικότητα και την ποιότητα ζωής.[1]

Αρχιτεκτονική των Συστημάτων IoT

Η αρχιτεκτονική ενός συστήματος IoT συνήθως αποτελείται από τα ακόλουθα επίπεδα:

- **Επίπεδο Αντίληψης (Perception Layer):** Περιλαμβάνει τους αισθητήρες και τους ενεργοποιητές που συλλέγουν δεδομένα από το περιβάλλον ή εκτελούν ενέργειες.
- **Επίπεδο Δικτύου (Network Layer):** Ασχολείται με τη μεταφορά των δεδομένων από το επίπεδο αντίληψης στα συστήματα επεξεργασίας, χρησιμοποιώντας διάφορες τεχνολογίες επικοινωνίας.
- **Επίπεδο Επεξεργασίας (Processing Layer):** Αναλύει και επεξεργάζεται τα δεδομένα, συχνά με τη χρήση cloud ή edge computing, για την εξαγωγή χρήσιμων πληροφοριών.
- **Επίπεδο Εφαρμογής (Application Layer):** Περιλαμβάνει τις εφαρμογές που χρησιμοποιούν τα επεξεργασμένα δεδομένα για την παροχή υπηρεσιών στους τελικούς χρήστες.

Η κατανόηση αυτών των επιπέδων είναι κρίσιμη για τον σχεδιασμό και την ανάπτυξη αποτελεσματικών IoT συστημάτων. [2]

Τεχνολογίες και Μέθοδοι στο Προτεινόμενο Σύστημα

Στο προτεινόμενο σύστημα, χρησιμοποιούνται συγκεκριμένες τεχνολογίες και μέθοδοι για την επίτευξη ασφαλούς και αποδοτικής επικοινωνίας μεταξύ των συσκευών:

- **Ασύρματη Επικοινωνία:** Η επικοινωνία μεταξύ των συσκευών επιτυγχάνεται μέσω ασύρματων πρωτοκόλλων, επιτρέποντας την ευελιξία και την ευκολία εγκατάστασης.
- **Κρυπτογράφηση Δεδομένων:** Για την προστασία των δεδομένων, εφαρμόζονται τεχνικές κρυπτογράφησης, διασφαλίζοντας την εμπιστευτικότητα και την ακεραιότητα των πληροφοριών που μεταδίδονται.

- **Αποθήκευση και Ανάλυση Δεδομένων:** Τα συλλεγόμενα δεδομένα αποθηκεύονται σε βάσεις δεδομένων και αναλύονται για την εξαγωγή χρήσιμων συμπερασμάτων, υποστηρίζοντας τη λήψη αποφάσεων.
- **Ασφάλεια και Ιδιωτικότητα:** Εφαρμόζονται μέτρα για την προστασία του συστήματος από κακόβουλες επιθέσεις και για τη διασφάλιση της ιδιωτικότητας των χρηστών.

Η ενσωμάτωση αυτών των τεχνολογιών και μεθόδων στο προτεινόμενο σύστημα στοχεύει στη δημιουργία ενός ασφαλούς, αποδοτικού και αξιόπιστου IoT περιβάλλοντος, ικανού να ανταποκριθεί στις σύγχρονες απαιτήσεις και προκλήσεις. [3]

Ασφάλεια στα Συστήματα IoT

Η ασφάλεια αποτελεί έναν από τους πιο σημαντικούς παράγοντες στα συστήματα IoT, καθώς τα δεδομένα που μεταδίδονται μεταξύ των συσκευών είναι ευαίσθητα και πολλές φορές κρίσιμα για τη λειτουργία του συστήματος. Σύμφωνα με μελέτες, οι κύριες προκλήσεις ασφαλείας στα IoT συστήματα περιλαμβάνουν:

- **Αδυναμίες στην επικοινωνία:** Οι συσκευές IoT συχνά χρησιμοποιούν ασύρματα δίκτυα για τη μετάδοση δεδομένων, γεγονός που τις καθιστά ευάλωτες σε επιθέσεις όπως υποκλοπή δεδομένων ή παραποίηση.
- **Περιορισμένοι πόροι συσκευών:** Πολλές IoT συσκευές έχουν περιορισμένες δυνατότητες σε υπολογιστική ισχύ και αποθήκευση, γεγονός που καθιστά δύσκολη την εφαρμογή ισχυρών αλγορίθμων ασφαλείας.
- **Αδυναμία επαλήθευσης ταυτότητας:** Η απουσία αξιόπιστων μηχανισμών ταυτοποίησης συσκευών μπορεί να οδηγήσει σε μη εξουσιοδοτημένη πρόσβαση στο δίκτυο. (ACM Transactions on IoT)

Στο δικό μας σύστημα, εφαρμόζονται οι εξής τεχνολογίες και πρακτικές για την ενίσχυση της ασφάλειας:

- **Υβριδική Κρυπτογράφηση (RSA/AES):** Συνδυάζεται η ασύμμετρη κρυπτογράφηση RSA για την ασφαλή ανταλλαγή κλειδιών και η συμμετρική AES για τη γρήγορη και αποδοτική κρυπτογράφηση δεδομένων.
- **Επαλήθευση Ταυτότητας:** Κάθε κόμβος διαθέτει μοναδικό ID, το οποίο ελέγχεται από τον κεντρικό κόμβο (Receiver). Αν το ID δεν είναι έγκυρο, η πρόσβαση απορρίπτεται.
- **Ανίχνευση Κακόβουλων Ενεργειών:** Εφαρμόζονται αλγόριθμοι για την παρακολούθηση της δραστηριότητας του δικτύου και την ανίχνευση ύποπτων συμπεριφορών, όπως μη εξουσιοδοτημένες προσπάθειες σύνδεσης.

Ανάλυση Δεδομένων στα Συστήματα IoT

Η αποθήκευση και ανάλυση δεδομένων είναι ουσιαστικής σημασίας για τη λειτουργία ενός συστήματος IoT, καθώς επιτρέπει την εξαγωγή χρήσιμων πληροφοριών για τη βελτίωση της αποδοτικότητας και της ασφάλειας.

Βάση Δεδομένων MySQL

Στο σύστημά μας χρησιμοποιείται μια σχεσιακή βάση δεδομένων MySQL για την αποθήκευση δεδομένων από τους αισθητήρες, καθώς και για την καταγραφή ύποπτων δραστηριοτήτων. Οι πίνακες της βάσης περιλαμβάνουν:

- Δεδομένα αισθητήρων (π.χ., θερμοκρασία, υγρασία).
- Εγκεκριμένους κόμβους (Senders) με τα μοναδικά τους IDs.
- Ύποπτες δραστηριότητες που ανιχνεύονται από το σύστημα.

Επεξεργασία Δεδομένων

Η ανάλυση των δεδομένων επιτυγχάνεται μέσω Python, χρησιμοποιώντας βιβλιοθήκες όπως Pandas και Matplotlib για την οπτικοποίηση και την εξαγωγή μοτίβων. Αυτή η διαδικασία συμβάλλει στη βελτιστοποίηση της λειτουργίας του συστήματος και στον εντοπισμό ανωμαλιών. [4-8]

2.2 Συστήματα Κρυπτογράφησης και Ταυτοποίησης κόμβου

Η ασφάλεια στην επικοινωνία των συσκευών IoT αποτελεί κρίσιμη παράμετρο για τη διασφάλιση της εμπιστευτικότητας, της ακεραιότητας και της αυθεντικότητας των δεδομένων που μεταδίδονται. Στο συγκεκριμένο έργο, εφαρμόζονται μέθοδοι κρυπτογράφησης και ταυτοποίησης κόμβων για την πρόληψη μη εξουσιοδοτημένης πρόσβασης και κακόβουλων ενεργειών.

Κρυπτογράφηση Δεδομένων

Η κρυπτογράφηση αποτελεί βασικό μηχανισμό για την προστασία των δεδομένων που διακινούνται μεταξύ των κόμβων (Senders) και του κεντρικού κόμβου (Receiver). Στο έργο μας χρησιμοποιείται ένα υβριδικό σύστημα κρυπτογράφησης, το οποίο συνδυάζει την ασύμμετρη κρυπτογράφηση RSA και τη συμμετρική κρυπτογράφηση AES.

Ασύμμετρη Κρυπτογράφηση (RSA)

Η RSA είναι ένας αλγόριθμος ασύμμετρης κρυπτογράφησης, ο οποίος χρησιμοποιεί δύο κλειδιά:

- Ένα **δημόσιο κλειδί** για την κρυπτογράφηση των δεδομένων.
- Ένα **ιδιωτικό κλειδί** για την αποκρυπτογράφηση των δεδομένων.

Στο σύστημά μας, η RSA χρησιμοποιείται για την ασφαλή ανταλλαγή του συμμετρικού κλειδιού AES. Ο Sender κρυπτογραφεί το AES κλειδί με το δημόσιο κλειδί του Receiver, εξασφαλίζοντας ότι μόνο ο Receiver μπορεί να το αποκρυπτογραφήσει.

Πλεονεκτήματα RSA:

- Εξασφαλίζει την ασφαλή μετάδοση κλειδιών.
- Παρέχει υψηλό επίπεδο ασφάλειας ακόμα και σε μη ασφαλή δίκτυα.

Συμμετρική Κρυπτογράφηση (AES)

Ο αλγόριθμος AES (Advanced Encryption Standard) χρησιμοποιείται για την κρυπτογράφηση των δεδομένων που αποστέλλονται από τον Sender στον Receiver.

Πλεονεκτήματα AES:

  - Γρήγορη και αποδοτική κρυπτογράφηση δεδομένων.
  - Κατάλληλη για συσκευές IoT με περιορισμένους πόρους.

Η συνδυαστική χρήση RSA και AES διασφαλίζει τόσο την ασφάλεια όσο και την αποδοτικότητα της επικοινωνίας.

Ταυτοποίηση Κόμβων

Η ταυτοποίηση των κόμβων είναι ζωτικής σημασίας για την πρόληψη μη εξουσιοδοτημένων προσβάσεων στο δίκτυο IoT. Στο σύστημά μας χρησιμοποιείται ένας μηχανισμός επαλήθευσης ταυτότητας, ο οποίος βασίζεται σε μοναδικά αναγνωριστικά (IDs).

Μηχανισμός Επαλήθευσης

1. Κάθε Sender διαθέτει ένα **μοναδικό ID** που αποστέλλεται μαζί με τα δεδομένα στον Receiver.
2. Ο Receiver συγκρίνει το ID με μια λίστα εγκεκριμένων κόμβων που τηρείται στη βάση δεδομένων.
3. Αν το ID δεν υπάρχει στη λίστα, η πρόσβαση απορρίπτεται και η ενέργεια καταγράφεται ως ύποπτη δραστηριότητα.

Πλεονεκτήματα Μηχανισμού Ταυτοποίησης

- Προστατεύει το δίκτυο από μη εξουσιοδοτημένες συσκευές.
- Επιτρέπει την παρακολούθηση της δραστηριότητας των κόμβων.

Εντοπισμός και Αντιμετώπιση Κακόβουλων Ενεργειών

Για την ενίσχυση της ασφάλειας, το σύστημα διαθέτει μηχανισμούς ανίχνευσης ύποπτων δραστηριοτήτων. Όταν ένας κόμβος αποτυγχάνει στην ταυτοποίηση ή επιχειρεί μη εξουσιοδοτημένη σύνδεση, η IP του καταγράφεται στον πίνακα ύποπτων δραστηριοτήτων (suspicious_activities).

Στάδια Ανίχνευσης

1. **Επαλήθευση ID**: Έλεγχος αν το ID υπάρχει στη λίστα εγκεκριμένων κόμβων.
2. **Ανίχνευση Υπερβολικών Αιτημάτων**: Καταγραφή IPs που επιχειρούν υπερβολικό αριθμό συνδέσεων.
3. **Αποκλεισμός Κακόβουλων IPs**: Οι IPs που παρουσιάζουν κακόβουλη συμπεριφορά αποκλείονται από το δίκτυο.

Ο συνδυασμός τεχνικών κρυπτογράφησης (RSA/AES) και μηχανισμών ταυτοποίησης συμβάλλει στη δημιουργία ενός ασφαλούς και αξιόπιστου συστήματος IoT. Η χρήση μοναδικών αναγνωριστικών IDs προστατεύει από μη εξουσιοδοτημένες συνδέσεις, ενώ οι μηχανισμοί ανίχνευσης ύποπτων ενεργειών ενισχύουν την ανθεκτικότητα του συστήματος σε κακόβουλες επιθέσεις.

Κεφάλαιο 3ο: Τεχνολογία

Στο κεφάλαιο αυτό θα περιγράφει η τεχνολογία που χρησιμοποιήθηκε για την υλοποίηση της εργασίας.

3.1 Python

Η Python είναι μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού γενικής χρήσης, γνωστή για την ευκολία χρήσης της και την αναγνωσιμότητα του κώδικα. Δημιουργήθηκε από τον Guido van Rossum το 1991 και από τότε εξελίσσεται διαρκώς, αποτελώντας ένα πανίσχυρο εργαλείο για διάφορες εφαρμογές, από την ανάπτυξη λογισμικού έως την ανάλυση δεδομένων.

Η Python είναι γλώσσα υψηλού επιπέδου, με σαφή και κατανοητή σύνταξη, γεγονός που την καθιστά κατάλληλη τόσο για αρχάριους όσο και για έμπειρους προγραμματιστές. Ένα από τα κύρια χαρακτηριστικά της είναι η ευελιξία της, καθώς υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού, όπως αντικειμενοστραφή, διαδικαστικό και λειτουργικό προγραμματισμό.

Η Python γεννήθηκε στα τέλη της δεκαετίας του 1980, όταν ο Guido van Rossum εργαζόταν στο Centrum Wiskunde & Informatica (CWI) στην Ολλανδία. Η γλώσσα κυκλοφόρησε επίσημα το 1991 ως Python 0.9.0, περιλαμβάνοντας δομές όπως οι συναρτήσεις, οι κλάσεις και οι εξαιρέσεις.

- **1994**: Κυκλοφορεί η Python 1.0, με βασικά χαρακτηριστικά όπως οι μονάδες (modules), οι συναρτήσεις και οι κλάσεις.
- **2000**: Η Python 2.0 εισάγει νέες δυνατότητες όπως η συλλογή απορριμμάτων (garbage collection) και οι λίστες κατανόησης (list comprehensions).
- **2008**: Κυκλοφορεί η Python 3.0, μια πιο σύγχρονη έκδοση με βελτιώσεις στη σύνταξη και στις δυνατότητες.
- **Σήμερα**: Η Python είναι μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού, χρησιμοποιούμενη σε τομείς όπως η τεχνητή νοημοσύνη, η ανάλυση δεδομένων και η ανάπτυξη web εφαρμογών.

Η Python έχει καθιερωθεί ως γλώσσα πολλαπλών χρήσεων, με εφαρμογές σε διάφορους τομείς:

1. **Ανάλυση Δεδομένων και Επιστήμη Δεδομένων**:
    - Χρησιμοποιείται για επεξεργασία και ανάλυση δεδομένων με εργαλεία όπως το Pandas, το NumPy και το Matplotlib.
    - Ιδανική για μηχανική μάθηση και τεχνητή νοημοσύνη μέσω βιβλιοθηκών όπως το TensorFlow και το Scikit-learn.
2. **Ανάπτυξη Web Εφαρμογών**:
    - Το Flask και το Django είναι δημοφιλή frameworks που χρησιμοποιούνται για τη δημιουργία ισχυρών και αποδοτικών web εφαρμογών.
3. **Αυτοματισμοί και Διαχείριση Συστημάτων**:

- Η Python είναι εξαιρετική για τη δημιουργία scripts που αυτοματοποιούν επαναλαμβανόμενες εργασίες.

4. **Ανάπτυξη Εφαρμογών και Παιχνιδιών**:
   - Χρησιμοποιείται για την ανάπτυξη παιχνιδιών (Pygame) και εφαρμογών με γραφικά περιβάλλοντα (Tkinter).
5. **Δίκτυα και IoT**:
   - Χρησιμοποιείται για επικοινωνία με συσκευές IoT και ανάλυση δεδομένων σε πραγματικό χρόνο.

Πλεονεκτήματα της Python

1. **Ευκολία στη Μάθηση**: Η Python διαθέτει σαφή και κατανοητή σύνταξη, διευκολύνοντας την εκμάθησή της από αρχάριους.
2. **Μεγάλη Κοινότητα Χρηστών**: Η Python διαθέτει μία από τις μεγαλύτερες κοινότητες προγραμματιστών, παρέχοντας υποστήριξη και πολλές βιβλιοθήκες.
3. **Πολυπλατφορμικότητα**: Μπορεί να εκτελεστεί σε πολλαπλές πλατφόρμες (Windows, Linux, macOS).
4. **Ευελιξία**: Υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού.
5. **Εκτενής Βιβλιοθήκη**: Η Python διαθέτει χιλιάδες βιβλιοθήκες που καλύπτουν διάφορους τομείς, από τη μηχανική μάθηση έως την ανάπτυξη web εφαρμογών.

Μειονεκτήματα της Python

1. **Χαμηλή Ταχύτητα Εκτέλεσης**: Ως γλώσσα υψηλού επιπέδου, είναι πιο αργή σε σύγκριση με γλώσσες όπως η C++ ή η Java.
2. **Κατανάλωση Μνήμης**: Οι δυναμικοί τύποι δεδομένων απαιτούν μεγαλύτερη μνήμη.
3. **Περιορισμοί σε Mobile Εφαρμογές**: Δεν χρησιμοποιείται ευρέως για ανάπτυξη εφαρμογών για κινητά τηλέφωνα.

Πού χρησιμοποιείται στο έργο μας;

Στο προτεινόμενο σύστημα, η Python παίζει κεντρικό ρόλο, καλύπτοντας τα παρακάτω:

1. **Υλοποίηση του Receiver**:
   - Παραλαβή κρυπτογραφημένων δεδομένων από τους Senders.
   - Αποκρυπτογράφηση των δεδομένων με RSA και AES.

- Αποθήκευση δεδομένων στη βάση MySQL.

2. **Επεξεργασία και Ανάλυση Δεδομένων**:
   - Χρήση βιβλιοθηκών όπως Pandas και Matplotlib για την ανάλυση και οπτικοποίηση των δεδομένων αισθητήρων.
3. **Ανάπτυξη Web Εφαρμογής**:
   - Ενσωμάτωση του Flask για τη δημιουργία διεπαφής χρήστη που επιτρέπει την προβολή δεδομένων και τον έλεγχο εντολών.
4. **Διαχείριση Ασφάλειας**:
   - Εφαρμογή μηχανισμών ανίχνευσης ύποπτων δραστηριοτήτων και καταγραφή τους στη βάση.

Πηγές: [9-13]

3.2 Flask

Το Flask είναι ένα ελαφρύ και ευέλικτο web framework για τη γλώσσα προγραμματισμού Python, σχεδιασμένο για τη δημιουργία web εφαρμογών και API. Αναπτύχθηκε από τον Armin Ronacher το 2010 και βασίζεται στη βιβλιοθήκη Werkzeug και στη μηχανή templates Jinja2. Το Flask ακολουθεί τη φιλοσοφία της ελάχιστης παρέμβασης (micro-framework), δίνοντας στους προγραμματιστές τη δυνατότητα να επιλέξουν τα εργαλεία και τις βιβλιοθήκες που χρειάζονται.

Σε αντίθεση με πιο ολοκληρωμένα frameworks όπως το Django, το Flask δεν περιλαμβάνει πολλές ενσωματωμένες λειτουργίες, παρέχοντας μεγαλύτερη ευελιξία και ελευθερία στον προγραμματιστή.

- **2010**: Το Flask κυκλοφόρησε ως ένα ελαφρύ web framework, ιδανικό για μικρές εφαρμογές και API.
- **2013**: Έγινε δημοφιλές λόγω της απλότητας και της ευελιξίας του, με αυξανόμενη υποστήριξη από την κοινότητα Python.
- **Σήμερα**: Το Flask αποτελεί ένα από τα πιο διαδεδομένα web frameworks, κατάλληλο τόσο για μικρές εφαρμογές όσο και για πιο σύνθετα έργα, χάρη στην υποστήριξη από πληθώρα επεκτάσεων.

Χρήσεις και Εφαρμογές

Το Flask χρησιμοποιείται ευρέως για την ανάπτυξη:

1. **Web Εφαρμογών**:
   - Δημιουργία δυναμικών σελίδων με δυνατότητα διαχείρισης δεδομένων από χρήστες.
   - Χρήση template rendering μέσω Jinja2.

2. **RESTful APIs**:
    - Ιδανικό για την ανάπτυξη API που επικοινωνούν με εξωτερικές εφαρμογές ή συσκευές IoT.
3. **Διαχείριση Εφαρμογών IoT**:
    - Ανάπτυξη εφαρμογών που επιτρέπουν την προβολή και διαχείριση δεδομένων σε συστήματα IoT.
4. **Πειραματικές Εφαρμογές**:
    - Λόγω της απλότητάς του, χρησιμοποιείται ευρέως για την ταχεία ανάπτυξη πρωτοτύπων.

Πλεονεκτήματα του Flask

1. **Απλότητα και Ευελιξία**:
    - Επιτρέπει στον προγραμματιστή να διαμορφώσει την εφαρμογή ακριβώς όπως θέλει, χωρίς περιττές λειτουργίες.
2. **Ελαφρύ και Επεκτάσιμο**:
    - Κατάλληλο για μικρές εφαρμογές, αλλά και για μεγαλύτερες με τη χρήση επεκτάσεων.
3. **Πλούσια Υποστήριξη**:
    - Μεγάλη κοινότητα χρηστών και πληθώρα βιβλιοθηκών και επεκτάσεων.
4. **Γρήγορη Ανάπτυξη**:
    - Ευκολία στη δημιουργία API και web εφαρμογών, ιδιαίτερα για έργα IoT.

Μειονεκτήματα του Flask

1. **Περιορισμένες Ενσωματωμένες Δυνατότητες**:
    - Σε σύγκριση με πιο ολοκληρωμένα frameworks όπως το Django, απαιτείται περισσότερη χειροκίνητη εργασία για βασικές λειτουργίες.
2. **Διαχείριση Πολυπλοκότητας**:
    - Για μεγαλύτερα έργα, μπορεί να γίνει δύσκολη η οργάνωση και διαχείριση του κώδικα χωρίς τη χρήση επιπλέον εργαλείων.
3. **Έλλειψη Ενσωματωμένου ORM**:

- Δεν παρέχει ενσωματωμένο σύστημα διαχείρισης βάσεων δεδομένων, αν και μπορεί να συνδυαστεί με βιβλιοθήκες όπως το SQLAlchemy.

Πού χρησιμοποιείται στο έργο μας;

Στο προτεινόμενο σύστημα IoT, το Flask παίζει σημαντικό ρόλο ως το framework για την ανάπτυξη της διεπαφής χρήστη. Χρησιμοποιείται για:

1. **Απεικόνιση Δεδομένων**:
   - Παρουσίαση δεδομένων αισθητήρων (π.χ., θερμοκρασία, υγρασία) που αποθηκεύονται στη βάση δεδομένων.
   - Εμφάνιση ύποπτων δραστηριοτήτων που έχουν καταγραφεί από το σύστημα.
2. **Διαχείριση Εντολών**:
   - Επιτρέπει στον χρήστη να στέλνει εντολές πίσω στους κόμβους (Senders), π.χ., για ενεργοποίηση ή απενεργοποίηση ενός relay.
3. **Διασύνδεση με MySQL**:
   - Χρήση του Flask για την επικοινωνία με τη βάση δεδομένων MySQL μέσω ερωτημάτων SQL.
4. **Ασφάλεια και Διαχείριση Πρόσβασης**:
   - Ενσωμάτωση βασικών μηχανισμών ασφάλειας, όπως η επαλήθευση ταυτότητας για την πρόσβαση στις λειτουργίες του συστήματος.

Πηγές: [14-16]

3.3 MySQL

Η MySQL είναι ένα δημοφιλές σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System - RDBMS), το οποίο βασίζεται στη γλώσσα SQL (Structured Query Language). Αποτελεί ανοιχτού κώδικα λογισμικό και υποστηρίζεται από την Oracle Corporation. Η MySQL χρησιμοποιείται ευρέως για την αποθήκευση και τη διαχείριση δεδομένων σε εφαρμογές που απαιτούν αξιόπιστη και αποτελεσματική διαχείριση πληροφορίας.

Η ευκολία εγκατάστασης, η σταθερότητα, και η υψηλή απόδοση καθιστούν τη MySQL μία από τις κορυφαίες επιλογές για διαχείριση δεδομένων σε εφαρμογές web, IoT, και επιχειρηματικά συστήματα.

- **1995**: Η MySQL δημιουργήθηκε από τον Michael Widenius και τον David Axmark. Σχεδιάστηκε ως ένα ελαφρύ, γρήγορο σύστημα βάσεων δεδομένων για μικρούς και μεσαίους οργανισμούς.
- **2000**: Η MySQL έγινε ανοιχτού κώδικα και κυκλοφόρησε υπό την άδεια GNU General Public License (GPL), αποκτώντας ευρεία υποστήριξη από την κοινότητα.
- **2010**: Η Oracle Corporation απέκτησε τη MySQL, ενισχύοντας την ανάπτυξη και υποστήριξή της.
- **Σήμερα**: Η MySQL χρησιμοποιείται ευρέως για την αποθήκευση και διαχείριση δεδομένων σε μικρές και μεγάλες εφαρμογές, υποστηρίζοντας εκατομμύρια χρήστες παγκοσμίως.

Χρήσεις και Εφαρμογές

Η MySQL είναι ένα ισχυρό εργαλείο για τη διαχείριση δεδομένων και χρησιμοποιείται ευρέως σε διάφορους τομείς, όπως:

1. **Web Εφαρμογές**:
    - Ενσωματώνεται με frameworks όπως το Flask και το Django για την αποθήκευση και ανάκτηση δεδομένων.
    - Χρησιμοποιείται από μεγάλες πλατφόρμες όπως το Facebook και το WordPress.
2. **Εφαρμογές IoT**:
    - Αποθήκευση δεδομένων αισθητήρων από απομακρυσμένες συσκευές.
    - Παρακολούθηση κακόβουλων ενεργειών και ανάλυση δεδομένων σε πραγματικό χρόνο.
3. **Επιχειρηματικά Συστήματα**:
    - Διαχείριση δεδομένων πελατών και αποθεμάτων.
    - Επεξεργασία μεγάλου όγκου πληροφοριών για αναλυτικά και στατιστικά εργαλεία.
4. **Αποθήκευση Big Data**:
    - Παρά το γεγονός ότι είναι σχεσιακή βάση δεδομένων, η MySQL μπορεί να συνεργαστεί με Big Data εργαλεία για ανάλυση δεδομένων.

Πλεονεκτήματα της MySQL

1. **Αξιοπιστία και Σταθερότητα**:

- Παρέχει σταθερή απόδοση ακόμα και σε εφαρμογές υψηλού φορτίου.

2. **Ευκολία στη Χρήση**:
    - Διαθέτει φιλική προς τον χρήστη διεπαφή και υποστηρίζει εύκολη διαχείριση μέσω εργαλείων όπως το phpMyAdmin.
3. **Επεκτασιμότητα**:
    - Υποστηρίζει την αποθήκευση μεγάλου όγκου δεδομένων.
4. **Διαλειτουργικότητα**:
    - Ενσωματώνεται με διάφορες γλώσσες προγραμματισμού όπως Python, Java και PHP.
5. **Ασφάλεια**:
    - Παρέχει μηχανισμούς ασφαλείας για την προστασία δεδομένων, όπως έλεγχος πρόσβασης και κρυπτογράφηση.

Μειονεκτήματα της MySQL

1. **Περιορισμοί σε Συμπλέγματα Δεδομένων**:
    - Δεν είναι η ιδανική επιλογή για εφαρμογές που απαιτούν πολύπλοκες σχέσεις δεδομένων.
2. **Έλλειψη Εργαλείων Ανάλυσης Big Data**:
    - Παρόλο που μπορεί να ενσωματωθεί με εργαλεία Big Data, δεν είναι σχεδιασμένη για αυτόνομη ανάλυση μεγάλων δεδομένων.
3. **Αναβάθμιση Ενδιάμεσου Λογισμικού**:
    - Μπορεί να απαιτεί συχνές αναβαθμίσεις για την υποστήριξη σύγχρονων λειτουργιών.

Πού χρησιμοποιείται στο έργο μας;

Στο προτεινόμενο σύστημα IoT, η MySQL χρησιμοποιείται ως το κεντρικό εργαλείο αποθήκευσης δεδομένων. Ο ρόλος της MySQL περιλαμβάνει:

1. **Αποθήκευση Δεδομένων Αισθητήρων**:
    - Όλες οι μετρήσεις των αισθητήρων, όπως θερμοκρασία και υγρασία, αποθηκεύονται σε έναν πίνακα, επιτρέποντας την εύκολη ανάκτηση και ανάλυση.
2. **Καταγραφή Ύποπτων Δραστηριοτήτων**:

- Καταγράφονται όλες οι μη εξουσιοδοτημένες προσπάθειες σύνδεσης ή κακόβουλες ενέργειες, υποστηρίζοντας την ανίχνευση και την ανάλυση της ασφάλειας.

3. **Διαχείριση Εγκεκριμένων Κόμβων**:
    - Υπάρχει πίνακας που περιέχει όλα τα έγκυρα IDs των Senders, διασφαλίζοντας την ταυτοποίηση και αποτροπή μη εξουσιοδοτημένων συνδέσεων.
4. **Διασύνδεση με Flask**:
    - Η MySQL επικοινωνεί με τη web εφαρμογή που αναπτύχθηκε στο Flask για την προβολή δεδομένων και την αποστολή εντολών.

Πηγές: [17-19]

3.4 RSA/ AES

Το RSA και το AES είναι δύο δημοφιλείς αλγόριθμοι κρυπτογράφησης που χρησιμοποιούνται για την προστασία των δεδομένων κατά τη μεταφορά τους. Αν και λειτουργούν με διαφορετικό τρόπο, η συνδυαστική χρήση τους (υβριδικό σύστημα) προσφέρει αυξημένη ασφάλεια και αποδοτικότητα.

- **RSA (Rivest–Shamir–Adleman):** Είναι ένας αλγόριθμος ασύμμετρης κρυπτογράφησης που χρησιμοποιεί δύο κλειδιά:
    - Ένα δημόσιο κλειδί για την κρυπτογράφηση.
    - Ένα ιδιωτικό κλειδί για την αποκρυπτογράφηση. Το RSA θεωρείται πολύ ασφαλές για την κρυπτογράφηση μικρών δεδομένων, όπως κλειδιά συμμετρικής κρυπτογράφησης.
- **AES (Advanced Encryption Standard):** Είναι ένας αλγόριθμος συμμετρικής κρυπτογράφησης που χρησιμοποιεί το ίδιο κλειδί τόσο για την κρυπτογράφηση όσο και για την αποκρυπτογράφηση. Το AES είναι γρήγορο και αποδοτικό για μεγάλες ποσότητες δεδομένων.

RSA:

- **1977:** Δημιουργήθηκε από τους Ronald Rivest, Adi Shamir και Leonard Adleman στο MIT.
- **1983:** Έγινε η πρώτη εμπορική χρήση του RSA για την προστασία δεδομένων.
- **Σήμερα:** Χρησιμοποιείται ευρέως για την κρυπτογράφηση κλειδιών και την ασφαλή ανταλλαγή δεδομένων.

AES:

- **1997:** Ξεκίνησε ο διαγωνισμός για την αντικατάσταση του DES (Data Encryption Standard).

- **2001:** Ο αλγόριθμος Rijndael επιλέχθηκε ως το νέο πρότυπο κρυπτογράφησης και ονομάστηκε AES.
- **Σήμερα:** Το AES είναι το πρότυπο κρυπτογράφησης για κυβερνητικές υπηρεσίες και επιχειρήσεις.

Χρήσεις και Εφαρμογές

RSA:

- Ασφαλής ανταλλαγή κλειδιών.
- Υπογραφές για την επαλήθευση της ταυτότητας χρηστών ή συσκευών.
- Εφαρμογές ηλεκτρονικού εμπορίου, όπως πιστωτικές κάρτες και διαδικτυακές πληρωμές.

AES:

- Κρυπτογράφηση μεγάλου όγκου δεδομένων.
- Αποθήκευση δεδομένων στο cloud.
- Εφαρμογές σε IoT συστήματα, όπου η γρήγορη επεξεργασία δεδομένων είναι απαραίτητη.

Πλεονεκτήματα του RSA και του AES

RSA:

1. **Ασφάλεια:** Βασίζεται στη δυσκολία παραγοντοποίησης μεγάλων αριθμών.
2. **Αυθεντικοποίηση:** Εξασφαλίζει την επαλήθευση της ταυτότητας των κόμβων.
3. **Διαλειτουργικότητα:** Υποστηρίζεται από πολλά πρωτόκολλα ασφαλείας (SSL/TLS).

AES:

1. **Αποδοτικότητα:** Πολύ γρήγορος αλγόριθμος για κρυπτογράφηση μεγάλων δεδομένων.
2. **Ισχυρή ασφάλεια:** Βασίζεται σε πολύπλοκες μαθηματικές πράξεις που είναι ανθεκτικές σε σύγχρονες επιθέσεις.
3. **Ευελιξία:** Υποστηρίζει κλειδιά διαφόρων μεγεθών (128-bit, 192-bit, 256-bit).

Μειονεκτήματα του RSA και του AES

RSA:

1. **Χαμηλή ταχύτητα:** Είναι αργός για την κρυπτογράφηση μεγάλων δεδομένων.

2. **Απαίτηση μεγάλων κλειδιών:** Η ασφάλεια εξαρτάται από το μέγεθος του κλειδιού (τουλάχιστον 2048 bits για σύγχρονες εφαρμογές).

AES:

1. **Κοινό Κλειδί:** Η διαχείριση και η ανταλλαγή του κλειδιού απαιτεί πρόσθετες διαδικασίες ασφαλείας.
2. **Ευαισθησία στο κλειδί:** Αν το κλειδί εκτεθεί, όλα τα δεδομένα είναι ευάλωτα.

Πού χρησιμοποιούνται στο έργο μας;

Στο προτεινόμενο σύστημα, το RSA και το AES χρησιμοποιούνται ως μέρος ενός υβριδικού συστήματος κρυπτογράφησης:

1. **Ασφαλής Ανταλλαγή Κλειδιών (RSA):**
    - Ο Sender δημιουργεί ένα τυχαίο AES κλειδί.
    - Το κρυπτογραφεί με το δημόσιο κλειδί του Receiver.
    - Στέλνει το κρυπτογραφημένο AES κλειδί στον Receiver, διασφαλίζοντας ότι μόνο ο Receiver μπορεί να το αποκρυπτογραφήσει.
2. **Κρυπτογράφηση Δεδομένων (AES):**
    - Ο Sender κρυπτογραφεί τα δεδομένα αισθητήρων (π.χ., θερμοκρασία, υγρασία) με το AES κλειδί.
    - Τα δεδομένα αποστέλλονται στον Receiver σε κρυπτογραφημένη μορφή.
3. **Αποκρυπτογράφηση και Επεξεργασία (Receiver):**
    - Ο Receiver αποκρυπτογραφεί το AES κλειδί με το ιδιωτικό του RSA κλειδί.
    - Με το AES κλειδί, αποκρυπτογραφεί τα δεδομένα και τα αποθηκεύει στη βάση δεδομένων.
4. **Εντοπισμός και Ασφάλεια:**
    - Οι μηχανισμοί RSA/AES διασφαλίζουν την προστασία των δεδομένων από υποκλοπές ή παραποιήσεις.

Πηγές: [20-24]

3.5 Esp32

Το ESP32 είναι ένα ολοκληρωμένο σύστημα μικροελεγκτή και μονάδας Wi-Fi/Bluetooth, σχεδιασμένο για εφαρμογές Internet of Things (IoT). Αναπτύχθηκε από την Espressif Systems και είναι μια προηγμένη έκδοση του ESP8266, με μεγαλύτερη επεξεργαστική ισχύ, βελτιωμένη συνδεσιμότητα και πρόσθετες δυνατότητες.

Το ESP32 διαθέτει ενσωματωμένο Wi-Fi, Bluetooth και πολλαπλά GPIO pins, καθιστώντας το ιδανικό για έργα IoT που απαιτούν συνδεσιμότητα και επεξεργασία δεδομένων σε πραγματικό χρόνο.

Το ESP32 είναι μία από τις πιο δημοφιλείς πλατφόρμες για έργα IoT, λόγω του συνδυασμού χαμηλού κόστους, υψηλής απόδοσης και ευκολίας χρήσης.

Το ESP32 χρησιμοποιείται ευρέως σε διάφορους τομείς λόγω της πολυλειτουργικότητάς του:

1. **IoT Συσκευές**:
    - Δημιουργία συσκευών που συλλέγουν και μεταδίδουν δεδομένα μέσω Wi-Fi.
    - Παρακολούθηση περιβάλλοντος με αισθητήρες (π.χ., θερμοκρασία, υγρασία).
2. **Έξυπνα Σπίτια**:
    - Έλεγχος φώτων, θερμοστάτη, κλειδαριών και άλλων έξυπνων συσκευών.
3. **Βιομηχανικός Αυτοματισμός**:
    - Παρακολούθηση και έλεγχος εξοπλισμού σε πραγματικό χρόνο.
4. **Ανάλυση Δεδομένων IoT**:
    - Αποστολή δεδομένων αισθητήρων σε κεντρικό διακομιστή ή cloud για επεξεργασία.
5. **Φορητές Συσκευές**:
    - Χρήση σε συσκευές που απαιτούν Bluetooth Low Energy (BLE) για επικοινωνία.

Χαρακτηριστικά του ESP32

1. **Υποστήριξη Διπλής Συνδεσιμότητας**:
    - Ενσωματωμένο Wi-Fi και Bluetooth, ιδανικό για IoT εφαρμογές.
2. **Επεξεργαστική Ισχύς**:
    - Διπύρηνος επεξεργαστής (Tensilica Xtensa) που προσφέρει υψηλή απόδοση.
3. **Χαμηλή Κατανάλωση Ενέργειας**:
    - Υποστηρίζει λειτουργίες χαμηλής κατανάλωσης για φορητές συσκευές.

4. **Υποστήριξη Πολλών Αισθητήρων**:
    - Διαθέτει πολλαπλές εισόδους/εξόδους (GPIOs) για σύνδεση με αισθητήρες και ενεργοποιητές.
5. **Χαμηλό Κόστος**:
    - Αποτελεί οικονομική λύση για έργα IoT.

Μειονεκτήματα του ESP32

1. **Επικοινωνία σε Μεγάλες Αποστάσεις**:
    - Περιορισμένη εμβέλεια Wi-Fi και Bluetooth σε σχέση με άλλες τεχνολογίες, όπως το LoRa.
2. **Περιορισμοί Μνήμης**:
    - Αν και επαρκής για πολλές εφαρμογές, η μνήμη μπορεί να είναι περιορισμένη σε έργα με πολύπλοκους αλγορίθμους.
3. **Απαιτήσεις Ρεύματος**:
    - Παρά τη χαμηλή κατανάλωση, ορισμένες λειτουργίες όπως το Wi-Fi απαιτούν μεγαλύτερη ισχύ.
4. **Καμπύλη Μάθησης**:
    - Απαιτείται εξοικείωση με εργαλεία ανάπτυξης όπως το ESP-IDF ή το Arduino IDE.

Πού χρησιμοποιείται στο έργο μας;

Στο προτεινόμενο σύστημα IoT, το ESP32 χρησιμοποιείται ως ο **Sender** για τη συλλογή δεδομένων αισθητήρων και την αποστολή τους στον κεντρικό κόμβο (Receiver). Ο ρόλος του ESP32 στο έργο περιλαμβάνει:

1. **Συλλογή Δεδομένων από Αισθητήρες**:
    - Συνδέεται με αισθητήρες θερμοκρασίας και υγρασίας για τη μέτρηση περιβαλλοντικών δεδομένων.
2. **Ασφαλής Επικοινωνία με τον Receiver**:
    - Χρησιμοποιεί πρωτόκολλα όπως το Wi-Fi για την αποστολή δεδομένων σε πραγματικό χρόνο.

- Εφαρμόζει RSA για την ανταλλαγή κλειδιών και AES για την κρυπτογράφηση των δεδομένων.

3. **Αποστολή Δεδομένων**:
   - Δημιουργεί πακέτα δεδομένων JSON που περιλαμβάνουν τις τιμές των αισθητήρων και το μοναδικό ID του κόμβου.
4. **Λήψη Εντολών από τον Receiver**:
   - Αποκρυπτογραφεί εντολές που στέλνει ο Receiver (π.χ., ενεργοποίηση relay).
5. **Προσαρμογή και Ευελιξία**:
   - Οι λειτουργίες του ESP32 μπορούν να επεκταθούν για την υποστήριξη περισσότερων αισθητήρων ή ενεργοποιητών.

Πηγές: [25-26]

Κεφάλαιο 4ο: Το σύστημα IoT κρυπτογράφησης και ταυτοποίησης

4.1 Εισαγωγή

Το σύστημα βασίζεται στην ασφαλή επικοινωνία συσκευών IoT. Αποτελείται από:

- **Sender (Αισθητήρες):** Αποστολή δεδομένων (π.χ., θερμοκρασία, υγρασία) από απομακρυσμένους κόμβους στον κεντρικό server (Receiver).
- **Receiver (Κεντρικός Κόμβος):** Λήψη δεδομένων, αποθήκευση σε βάση δεδομένων και αποστολή εντολών πίσω στον sender (π.χ., ενεργοποίηση relay).
- **Frontend (Selida.py):** Προβολή δεδομένων και παρακολούθηση ύποπτων δραστηριοτήτων μέσω ενός απλού interface.

Το σύστημα χρησιμοποιεί:

1. Κρυπτογράφηση (RSA/AES) για την ασφάλεια της επικοινωνίας.
2. Έλεγχο ταυτότητας για την επαλήθευση των κόμβων.
3. Μηχανισμό αποκλεισμού IP για την αποτροπή κακόβουλων ενεργειών.

Το σύστημα που υλοποιήσαμε αφορά μια πλήρη τοπολογία επικοινωνίας μεταξύ απομακρυσμένων συσκευών IoT και ενός κεντρικού κόμβου που διαχειρίζεται τα δεδομένα και τις εντολές. Η κύρια λειτουργία του συστήματος είναι η ανταλλαγή πληροφοριών και η λήψη αποφάσεων σε πραγματικό χρόνο, εξασφαλίζοντας παράλληλα την ασφάλεια, την ακεραιότητα, και την αποτελεσματικότητα της επικοινωνίας.

Αυτό το σύστημα αποτελείται από δύο βασικές οντότητες: τον **Sender** και τον **Receiver**. Ο Sender είναι μια απομακρυσμένη συσκευή εξοπλισμένη με αισθητήρες που συλλέγουν δεδομένα όπως θερμοκρασία και υγρασία. Αυτά τα δεδομένα πρέπει να μεταφερθούν με ασφαλή τρόπο στον Receiver, ο οποίος λειτουργεί ως κεντρικός κόμβος και είναι υπεύθυνος για την αποθήκευση, ανάλυση και επεξεργασία των πληροφοριών. Ο Receiver, μετά τη λήψη των δεδομένων, έχει τη δυνατότητα να αποστέλλει εντολές πίσω στον Sender, δίνοντας οδηγίες για ενεργοποίηση ή απενεργοποίηση συσκευών, όπως ένα relay.

Η ασφάλεια είναι κεντρικό στοιχείο του συστήματος. Για την προστασία των δεδομένων που διακινούνται, χρησιμοποιείται ένα υβριδικό πρωτόκολλο κρυπτογράφησης. Τα ευαίσθητα δεδομένα που συλλέγονται από τους αισθητήρες κρυπτογραφούνται με τον αλγόριθμο AES, ο οποίος είναι γνωστός για την αποτελεσματικότητα και την ασφάλειά του. Ωστόσο, για την ασφαλή ανταλλαγή του

AES κλειδιού, χρησιμοποιείται RSA κρυπτογράφηση. Αυτό το συνδυαστικό μοντέλο διασφαλίζει ότι μόνο ο Receiver μπορεί να αποκρυπτογραφήσει και να επεξεργαστεί τα δεδομένα που αποστέλλει ο Sender.

Ένα άλλο σημαντικό χαρακτηριστικό του συστήματος είναι ο έλεγχος ταυτότητας. Κάθε Sender διαθέτει ένα μοναδικό ID, το οποίο επαληθεύεται από τον Receiver κατά την αρχική σύνδεση. Αν το ID του Sender δεν υπάρχει στον κατάλογο εγκεκριμένων συσκευών που διατηρεί ο Receiver, η σύνδεση απορρίπτεται, και το γεγονός καταγράφεται ως ύποπτη δραστηριότητα. Αυτός ο μηχανισμός προστατεύει το σύστημα από μη εξουσιοδοτημένες συσκευές και κακόβουλες επιθέσεις.

Ο Receiver λειτουργεί και ως κεντρικό σημείο αποθήκευσης δεδομένων. Όλα τα δεδομένα που συλλέγονται από τους Senders καταγράφονται σε μια βάση δεδομένων MySQL. Η δομή της βάσης περιλαμβάνει πίνακες για τα δεδομένα των αισθητήρων, για τις συσκευές που είναι εγκεκριμένες να συμμετέχουν στο δίκτυο, και για την καταγραφή ύποπτων δραστηριοτήτων. Αυτή η οργάνωση διευκολύνει τη διαχείριση των δεδομένων και την παραγωγή αναφορών.

Η επικοινωνία μεταξύ των συσκευών δεν είναι μόνο μονόδρομη. Μετά την επεξεργασία των δεδομένων, ο Receiver έχει τη δυνατότητα να στέλνει εντολές πίσω στον Sender. Αυτές οι εντολές είναι κρυπτογραφημένες με AES και μπορεί να περιλαμβάνουν οδηγίες όπως η ενεργοποίηση ή απενεργοποίηση συσκευών. Ο Sender λαμβάνει το μήνυμα, το αποκρυπτογραφεί, και εκτελεί την αντίστοιχη ενέργεια, επιτρέποντας έτσι τη διευρυμένη διαχείριση και τον έλεγχο του δικτύου.

Η απεικόνιση των δεδομένων γίνεται μέσω μιας εφαρμογής βασισμένης σε Flask. Η εφαρμογή αυτή παρέχει μια φιλική προς τον χρήστη διεπαφή όπου μπορούν να προβληθούν τα δεδομένα των αισθητήρων, οι καταγεγραμμένες ύποπτες δραστηριότητες, και οι αποκλεισμένες διευθύνσεις IP. Αυτή η δυνατότητα επιτρέπει στους διαχειριστές του συστήματος να παρακολουθούν σε πραγματικό χρόνο τη λειτουργία του δικτύου, να εντοπίζουν προβλήματα, και να λαμβάνουν μέτρα για τη βελτίωση της ασφάλειας και της απόδοσης.

Το σύστημα έχει σχεδιαστεί για να είναι επεκτάσιμο και να υποστηρίζει περισσότερους κόμβους καθώς και πρόσθετες λειτουργίες. Η χρήση ανοιχτών πρωτοκόλλων και τεχνολογιών διασφαλίζει τη συμβατότητα και τη δυνατότητα προσαρμογής στις ανάγκες διαφορετικών εφαρμογών. Τέλος, η ενσωμάτωση μέτρων ασφαλείας, όπως η κρυπτογράφηση και ο έλεγχος ταυτότητας, καθιστά το σύστημα ανθεκτικό απέναντι σε κακόβουλες επιθέσεις, εξασφαλίζοντας την ακεραιότητα και την εμπιστευτικότητα των δεδομένων που διακινούνται.

4.2 Ενότητες Λειτουργιών με Αναλυτική Περιγραφή και Κώδικα

4.2.1 Sender

Ο **Sender** είναι υπεύθυνος για:

1. Την κρυπτογράφηση των δεδομένων αισθητήρων.
2. Την αποστολή δεδομένων στον receiver.
3. Τη λήψη εντολών από τον receiver.

Κώδικας Sender με Σχόλια

```
import socket
import os
import json
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes


# Δημιουργία RSA Key Pair
private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
public_key = private_key.public_key()


# ID του κόμβου
node_id = "node1"


# Τιμές αισθητήρων
sensor_data = {"temperature": 22.5, "humidity": 60.2}


# Δημιουργία σύνδεσης με τον receiver
HOST = '192.168.1.200'  # IP του receiver
PORT = 65432
```

```
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect((HOST, PORT))

# Λήψη δημόσιου κλειδιού του receiver
receiver_pub_key_pem = sock.recv(1024)
receiver_pub_key = serialization.load_pem_public_key(receiver_pub_key_pem)

# Δημιουργία AES Key για την κρυπτογράφηση των δεδομένων
aes_key = os.urandom(32)
iv = os.urandom(16)

# Κρυπτογράφηση του AES Key με το δημόσιο κλειδί του receiver
encrypted_aes_key = receiver_pub_key.encrypt(
    aes_key,
    padding.OAEP(mgf=padding.MGF1(hashes.SHA256()), algorithm=hashes.SHA256(), label=None)
)

# Αποστολή ID, κρυπτογραφημένου AES Key και IV
sock.sendall(json.dumps({"id": node_id, "encrypted_aes_key": encrypted_aes_key.hex(), "iv": iv.hex()}).encode())

# Κρυπτογράφηση δεδομένων αισθητήρων με AES
aes_cipher = Cipher(algorithms.AES(aes_key), modes.CFB(iv))
encryptor = aes_cipher.encryptor()
encrypted_sensor_data = encryptor.update(json.dumps(sensor_data).encode())

# Αποστολή κρυπτογραφημένων δεδομένων αισθητήρων
sock.sendall(encrypted_sensor_data)
```

```
# Λήψη απάντησης από τον receiver
encrypted_message = sock.recv(1024)
decryptor = aes_cipher.decryptor()
control_message = json.loads(decryptor.update(encrypted_message).decode())
print(f"Received control message: {control_message}")

sock.close()
```

Ακολουθεί η αναλυτική παρουσίαση του κώδικα του **Sender** με διαχωρισμό σε κομμάτια και περιγραφή της λειτουργίας του.

1.Εισαγωγή Βιβλιοθηκών

```
import socket
import os
import json
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
```

Αυτό το τμήμα κώδικα εισάγει τις απαιτούμενες βιβλιοθήκες:

- **socket**: Χρησιμοποιείται για τη δικτυακή επικοινωνία μεταξύ του Sender και του Receiver.
- **os**: Παρέχει εργαλεία για τη διαχείριση συστήματος και τη δημιουργία τυχαίων δεδομένων (π.χ., AES key).
- **json**: Χρησιμοποιείται για τη μετατροπή δεδομένων σε μορφή JSON, ώστε να μεταφερθούν μέσω δικτύου.
- **cryptography.hazmat**: Περιλαμβάνει αλγορίθμους και εργαλεία για την κρυπτογράφηση (RSA, AES) και την επαλήθευση δεδομένων.

2. Δημιουργία Κλειδιών RSA

```
private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
```

```
public_key = private_key.public_key()
```

- **private_key**: Δημιουργείται ένα ιδιωτικό RSA κλειδί με μέγεθος 2048 bits. Αυτό το κλειδί χρησιμοποιείται για την κρυπτογράφηση και την αποκρυπτογράφηση.
- **public_key**: Το δημόσιο κλειδί εξάγεται από το ιδιωτικό. Το δημόσιο κλειδί χρησιμοποιείται για την κρυπτογράφηση των δεδομένων που προορίζονται για τον Receiver.

Η RSA είναι ένας αλγόριθμος ασύμμετρης κρυπτογράφησης, που εξασφαλίζει ότι μόνο ο κάτοχος του ιδιωτικού κλειδιού μπορεί να αποκρυπτογραφήσει τα δεδομένα που έχουν κρυπτογραφηθεί με το δημόσιο κλειδί.

3. Ορισμός ID Κόμβου και Συλλογή Δεδομένων

```
node_id = "node1"

sensor_data = {"temperature": 22.5, "humidity": 60.2}
```

- **node_id**: Το μοναδικό αναγνωριστικό του κόμβου. Αυτό το ID χρησιμοποιείται από τον Receiver για την επαλήθευση της ταυτότητας του Sender.
- **sensor_data**: Ένα λεξικό (dictionary) που περιέχει τις μετρήσεις από τους αισθητήρες, συγκεκριμένα την τρέχουσα θερμοκρασία και την υγρασία.

4. Δημιουργία Σύνδεσης με τον Receiver

```
HOST = '192.168.1.2'  # IP του receiver

PORT = 65432

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

sock.connect((HOST, PORT))
```

- **HOST**: Η διεύθυνση IP του Receiver. Εδώ, η διεύθυνση είναι τοπική στο δίκτυο.
- **PORT**: Η θύρα στην οποία "ακούει" ο Receiver για εισερχόμενες συνδέσεις.
- **socket.socket**: Δημιουργεί ένα νέο αντικείμενο socket για τη δικτυακή επικοινωνία.
- **sock.connect**: Εγκαθιστά τη σύνδεση με τον Receiver χρησιμοποιώντας το IP και τη θύρα.

5. Λήψη Δημόσιου Κλειδιού από τον Receiver

```
receiver_pub_key_pem = sock.recv(1024)
receiver_pub_key = serialization.load_pem_public_key(receiver_pub_key_pem)
```

- **sock.recv(1024)**: Ο Sender λαμβάνει το δημόσιο κλειδί του Receiver σε μορφή PEM (Privacy-Enhanced Mail).
- **serialization.load_pem_public_key**: Φορτώνει το δημόσιο κλειδί από τη μορφή PEM, ώστε να μπορεί να χρησιμοποιηθεί για κρυπτογράφηση.

Το δημόσιο κλειδί του Receiver χρησιμοποιείται για την ασφαλή αποστολή του AES key.

6. Δημιουργία AES Key και IV

```
aes_key = os.urandom(32)
iv = os.urandom(16)
```

- **aes_key**: Δημιουργείται ένα τυχαίο συμμετρικό κλειδί 256-bit για τον αλγόριθμο AES.
- **iv**: Δημιουργείται ένας τυχαίος IV (Initialization Vector) μήκους 16 bytes, που είναι απαραίτητος για τον τρόπο λειτουργίας CFB του AES.

Το AES χρησιμοποιείται για την κρυπτογράφηση των δεδομένων αισθητήρων.

7. Κρυπτογράφηση AES Key με το Δημόσιο Κλειδί του Receiver

```
encrypted_aes_key = receiver_pub_key.encrypt(
    aes_key,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
```

```
    )
)
```

- **receiver_pub_key.encrypt**: Χρησιμοποιείται το δημόσιο κλειδί του Receiver για την κρυπτογράφηση του AES key.
- **OAEP (Optimal Asymmetric Encryption Padding)**: Μια μέθοδος padding για την ενίσχυση της ασφάλειας της RSA κρυπτογράφησης. Το OAEP χρησιμοποιεί SHA-256 ως αλγόριθμο κατακερματισμού.

Αυτό το βήμα εξασφαλίζει ότι το AES key μπορεί να αποκρυπτογραφηθεί μόνο από τον Receiver, ο οποίος διαθέτει το αντίστοιχο ιδιωτικό κλειδί.

8. Αποστολή ID, Κρυπτογραφημένου AES Key και IV

```
sock.sendall(json.dumps({
    "id": node_id,
    "encrypted_aes_key": encrypted_aes_key.hex(),
    "iv": iv.hex()
}).encode())
```

- **node_id**: Το μοναδικό αναγνωριστικό του Sender αποστέλλεται για την επαλήθευση ταυτότητας.
- **encrypted_aes_key.hex()**: Το κρυπτογραφημένο AES key μετατρέπεται σε μορφή hexadecimal για αποστολή μέσω δικτύου.
- **iv.hex()**: Το IV μετατρέπεται σε μορφή hexadecimal.

Το μήνυμα περιέχει όλα τα απαραίτητα στοιχεία για την αποκρυπτογράφηση και επεξεργασία των δεδομένων από τον Receiver.

9. Κρυπτογράφηση Δεδομένων Αισθητήρων

```
aes_cipher = Cipher(algorithms.AES(aes_key), modes.CFB(iv))
encryptor = aes_cipher.encryptor()
encrypted_sensor_data = encryptor.update(json.dumps(sensor_data).encode())
```

- **Cipher(algorithms.AES(aes_key), modes.CFB(iv))**: Δημιουργείται ένα αντικείμενο κρυπτογράφησης AES με το κλειδί και το IV.

- **encryptor.update**: Κρυπτογραφεί τα δεδομένα αισθητήρων (sensor_data) αφού πρώτα μετατραπούν σε JSON και κωδικοποιηθούν σε bytes.

Η χρήση του AES σε λειτουργία CFB (Cipher Feedback) επιτρέπει την κρυπτογράφηση δεδομένων χωρίς να απαιτείται padding.

10. Αποστολή Κρυπτογραφημένων Δεδομένων

```
sock.sendall(encrypted_sensor_data)
```

- **sock.sendall**: Στέλνει τα κρυπτογραφημένα δεδομένα αισθητήρων στον Receiver.

Σε αυτό το σημείο, ο Receiver μπορεί να αποκρυπτογραφήσει τα δεδομένα χρησιμοποιώντας το AES key και το IV που έχουν προηγουμένως μεταδοθεί.

---

11. Λήψη και Αποκρυπτογράφηση Μηνύματος από τον Receiver

```
encrypted_message = sock.recv(1024)

decryptor = aes_cipher.decryptor()

control_message = json.loads(decryptor.update(encrypted_message).decode())

print(f"Received control message: {control_message}")
```

- **sock.recv(1024)**: Λαμβάνει το κρυπτογραφημένο μήνυμα από τον Receiver.
- **decryptor.update**: Αποκρυπτογραφεί το μήνυμα χρησιμοποιώντας το AES key και το IV.
- **json.loads**: Μετατρέπει το αποκρυπτογραφημένο μήνυμα από JSON σε Python dictionary.

Το μήνυμα περιέχει οδηγίες από τον Receiver (π.χ., ενεργοποίηση ή απενεργοποίηση ενός relay).

12. Κλείσιμο Σύνδεσης

```
sock.close()
```

- **sock.close**: Τερματίζει τη σύνδεση με τον Receiver, απελευθερώνοντας πόρους του συστήματος.

Ο Sender συλλέγει δεδομένα από τους αισθητήρες και τα αποστέλλει με ασφάλεια στον Receiver. Χρησιμοποιεί ασύμμετρη κρυπτογράφηση για την ανταλλαγή του AES key και συμμετρική

κρυπτογράφηση για τα ίδια τα δεδομένα. Μετά την επιτυχή αποστολή, λαμβάνει ένα κρυπτογραφημένο μήνυμα από τον Receiver, το αποκρυπτογραφεί και εκτελεί την αντίστοιχη ενέργεια.

4.2.2 Receiver

Ο **Receiver** αναλαμβάνει:

1. Τη λήψη και αποκρυπτογράφηση δεδομένων.
2. Την αποθήκευση δεδομένων στη βάση.
3. Την αποστολή εντολών στον sender.

Κώδικας Receiver με Σχόλια

```
import socket
import json
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
import mysql.connector


# Δημιουργία RSA Key Pair
private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
public_key = private_key.public_key()


# Σύνδεση με MySQL
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="secure_iot"
)
cursor = db.cursor()
```

```
# Έλεγχος αν ο κόμβος είναι εγκεκριμένος
def is_node_authorized(node_id):
    query = "SELECT COUNT(*) FROM nodes WHERE id = %s"
    cursor.execute(query, (node_id,))
    return cursor.fetchone()[0] > 0

# Αποθήκευση δεδομένων αισθητήρων
def save_sensor_data(node_id, temperature, humidity):
    query = "INSERT INTO sensor_data (node_id, temperature, humidity) VALUES (%s, %s, %s)"
    cursor.execute(query, (node_id, temperature, humidity))
    db.commit()

# Ρύθμιση σύνδεσης
HOST = '0.0.0.0'
PORT = 65432
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((HOST, PORT))
sock.listen(1)

while True:
    conn, addr = sock.accept()

    # Λήψη δημόσιου κλειδιού
    pub_key_pem = public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )
```

```
    conn.sendall(pub_key_pem)

    # Λήψη ID και AES Key
    data = json.loads(conn.recv(1024).decode())
    node_id = data["id"]

    # Έλεγχος αν ο κόμβος είναι εγκεκριμένος
    if not is_node_authorized(node_id):
        conn.close()
        continue

    # Αποκρυπτογράφηση και αποθήκευση δεδομένων
    encrypted_sensor_data = conn.recv(1024)
    # Αποθήκευση κώδικα όπως προηγουμένως...

    # Αποστολή μηνύματος στον sender
    response_message = {"command": "ON"}
    encrypted_response = aes_cipher.encryptor().update(json.dumps(response_message).encode())
    conn.sendall(encrypted_response)

    conn.close()
```

Ακολουθεί η αναλυτική παρουσίαση του κώδικα του **Receiver** με διαχωρισμό σε κομμάτια και εξήγηση της λειτουργίας του.

1. Εισαγωγή Βιβλιοθηκών

```
import socket
import json
```

```
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
import mysql.connector
```

- **Περιγραφή:**

Αυτό το τμήμα εισάγει τις απαραίτητες βιβλιοθήκες:

- **socket**: Για τη δικτυακή επικοινωνία με τον Sender.
- **json**: Για τη μετατροπή δεδομένων σε JSON και την αποστολή/λήψη μέσω δικτύου.
- **cryptography.hazmat**: Για την κρυπτογράφηση/αποκρυπτογράφηση (RSA, AES).
- **mysql.connector**: Για τη σύνδεση με τη βάση δεδομένων MySQL, όπου αποθηκεύονται οι πληροφορίες των αισθητήρων και των κόμβων.

---

2. Δημιουργία Κλειδιών RSA

```
private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)
public_key = private_key.public_key()
```

- **Περιγραφή:**
- **private_key**: Δημιουργείται το ιδιωτικό RSA κλειδί του Receiver.
- **public_key**: Το δημόσιο κλειδί εξάγεται από το ιδιωτικό και αποστέλλεται στον Sender, ώστε να μπορεί να κρυπτογραφήσει το AES key.

Το RSA είναι υπεύθυνο για την ασφάλεια στην ανταλλαγή του AES key.

---

3. Σύνδεση με τη Βάση Δεδομένων

```
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="secure_iot"
)
```

```
cursor = db.cursor()
```

- **Περιγραφή:**
- **db**: Δημιουργείται σύνδεση με τη MySQL βάση δεδομένων.
- **cursor**: Το αντικείμενο αυτό χρησιμοποιείται για την εκτέλεση SQL εντολών.

Η βάση δεδομένων περιέχει τους πίνακες για τους κόμβους, τα δεδομένα αισθητήρων, και τις ύποπτες δραστηριότητες.

---

4. Έλεγχος Εγκεκριμένων Κόμβων

```
def is_node_authorized(node_id):
    query = "SELECT COUNT(*) FROM nodes WHERE id = %s"
    cursor.execute(query, (node_id,))
    return cursor.fetchone()[0] > 0
```

- **Περιγραφή:**
- **query**: Ελέγχει αν το node_id του Sender υπάρχει στον πίνακα nodes.
- **cursor.execute**: Εκτελεί την SQL ερώτηση.
- **return**: Επιστρέφει True αν ο κόμβος είναι εγκεκριμένος, αλλιώς False.

Αυτός ο έλεγχος διασφαλίζει ότι μόνο εγκεκριμένοι κόμβοι μπορούν να στέλνουν δεδομένα.

---

5. Αποθήκευση Δεδομένων Αισθητήρων

```
def save_sensor_data(node_id, temperature, humidity):
    query = "INSERT INTO sensor_data (node_id, temperature, humidity) VALUES (%s, %s, %s)"
    cursor.execute(query, (node_id, temperature, humidity))
    db.commit()
    print(f"Sensor data saved: Node ID: {node_id}, Temperature: {temperature}, Humidity: {humidity}")
```

- **Περιγραφή:**
- **query**: Εισάγει δεδομένα αισθητήρων στον πίνακα sensor_data.
- **db.commit()**: Αποθηκεύει μόνιμα τις αλλαγές στη βάση δεδομένων.

- **print()**: Εμφανίζει μήνυμα επιβεβαίωσης για την αποθήκευση.

---

6. Ρύθμιση Σύνδεσης

```
HOST = '0.0.0.0'
PORT = 65432
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.bind((HOST, PORT))
sock.listen(1)
```

- **Περιγραφή:**
- **HOST**: Ο Receiver είναι έτοιμος να δεχτεί συνδέσεις από οποιοδήποτε IP στο δίκτυο.
- **PORT**: Η θύρα στην οποία "ακούει" ο Receiver για εισερχόμενες συνδέσεις.
- **sock.bind**: Συνδέει το socket με το συγκεκριμένο IP και θύρα.
- **sock.listen(1)**: Ο Receiver περιμένει για μία σύνδεση κάθε φορά.

---

7. Λήψη Δημόσιου Κλειδιού από τον Receiver

```
pub_key_pem = public_key.public_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo
)
conn.sendall(pub_key_pem)
```

- **Περιγραφή:**
- **public_bytes**: Το δημόσιο κλειδί μετατρέπεται σε μορφή PEM.
- **conn.sendall**: Αποστέλλει το δημόσιο κλειδί στον Sender, ώστε να μπορέσει να κρυπτογραφήσει το AES key.

---

8. Λήψη ID και AES Key

```
data = json.loads(conn.recv(1024).decode())
node_id = data["id"]
```

```
encrypted_aes_key = bytes.fromhex(data["encrypted_aes_key"])
iv = bytes.fromhex(data["iv"])
```

- **Περιγραφή:**
- **conn.recv**: Λαμβάνει δεδομένα από τον Sender.
- **json.loads**: Μετατρέπει τα δεδομένα από JSON σε Python dictionary.
- **node_id**: Το ID του κόμβου για επαλήθευση.
- **encrypted_aes_key**: Το κρυπτογραφημένο AES key σε μορφή bytes.
- **iv**: Το Initialization Vector για την αποκρυπτογράφηση.

---

9. Επαλήθευση και Αποκρυπτογράφηση AES Key

```
if not is_node_authorized(node_id):
    conn.close()
    continue
aes_key = private_key.decrypt(
    encrypted_aes_key,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    ))
```

- **Περιγραφή:**
- **is_node_authorized**: Ελέγχει αν ο κόμβος είναι εγκεκριμένος.
- **private_key.decrypt**: Αποκρυπτογραφεί το AES key χρησιμοποιώντας το ιδιωτικό RSA κλειδί.
- **OAEP padding**: Διασφαλίζει την ασφάλεια κατά την αποκρυπτογράφηση.

---

10. Λήψη και Αποκρυπτογράφηση Δεδομένων

```
encrypted_sensor_data = conn.recv(1024)
```

```
aes_cipher = Cipher(algorithms.AES(aes_key), modes.CFB(iv))
decryptor = aes_cipher.decryptor()
sensor_data = json.loads(decryptor.update(encrypted_sensor_data).decode())
```

- **Περιγραφή:**
- **conn.recv**: Λαμβάνει τα κρυπτογραφημένα δεδομένα αισθητήρων.
- **Cipher**: Δημιουργεί έναν AES αποκρυπτογραφητή.
- **decryptor.update**: Αποκρυπτογραφεί τα δεδομένα.
- **json.loads**: Μετατρέπει τα δεδομένα σε Python dictionary.

---

11. Αποστολή Μηνύματος στον Sender

```
response_message = {"command": "ON"}
encrypted_response = aes_cipher.encryptor().update(json.dumps(response_message).encode())
conn.sendall(encrypted_response)
```

- **Περιγραφή:**
- **response_message**: Δημιουργεί ένα μήνυμα με εντολές για τον Sender.
- **aes_cipher.encryptor**: Κρυπτογραφεί το μήνυμα με AES.
- **conn.sendall**: Στέλνει το κρυπτογραφημένο μήνυμα πίσω στον Sender.

---

12. Κλείσιμο Σύνδεσης

```
conn.close()
```

- **Περιγραφή:**
- **conn.close**: Τερματίζει τη σύνδεση με τον Sender.

4.2.3 Frontend (Selida.py)

Ακολουθεί η αναλυτική παρουσίαση του **Frontend (Selida.py)** με διαχωρισμό σε κομμάτια και εξήγηση της λειτουργίας του.

---

1. Εισαγωγή Βιβλιοθηκών

```
from flask import Flask, render_template

import mysql.connector
```

- **Περιγραφή:**
- **Flask**: Το Flask είναι ένα framework για τη δημιουργία web εφαρμογών. Χρησιμοποιείται για την προβολή δεδομένων και τη δημιουργία μιας διεπαφής χρήστη.
- **mysql.connector**: Χρησιμοποιείται για τη σύνδεση με τη MySQL βάση δεδομένων και την ανάκτηση δεδομένων.

Αυτές οι βιβλιοθήκες αποτελούν τη βάση της εφαρμογής για την προβολή των δεδομένων σε ένα web περιβάλλον.

---

2. Ρύθμιση της Εφαρμογής Flask

```
app = Flask(__name__)
```

- **Περιγραφή:**
- **app**: Δημιουργείται ένα αντικείμενο Flask που αντιπροσωπεύει την εφαρμογή. Αυτό το αντικείμενο χειρίζεται αιτήματα από τον χρήστη και επιστρέφει απαντήσεις.

---

3. Σύνδεση με τη Βάση Δεδομένων

```
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="secure_iot"
)
```

- **Περιγραφή:**
- **db**: Δημιουργείται σύνδεση με τη βάση δεδομένων MySQL.
- **host**: Καθορίζει τον server της MySQL (τοπικός server σε αυτή την περίπτωση).
- **user**: Το όνομα χρήστη της MySQL (π.χ., root).
- **password**: Ο κωδικός πρόσβασης στη MySQL.

- **database**: Το όνομα της βάσης δεδομένων όπου αποθηκεύονται οι πληροφορίες (π.χ., secure_iot).

Αυτή η σύνδεση είναι απαραίτητη για την ανάκτηση δεδομένων.

---

4. Διαδρομή για Προβολή Δεδομένων Αισθητήρων

```
@app.route("/")
def sensor_data():
    cursor = db.cursor(dictionary=True)
    cursor.execute("""
        SELECT sd.id, sd.node_id, sd.temperature, sd.humidity, sd.timestamp, n.name
        FROM sensor_data sd
        JOIN nodes n ON sd.node_id = n.id
        ORDER BY sd.timestamp DESC
    """)
    data = cursor.fetchall()
    return render_template("sensor_data.html", data=data)
```

- **Περιγραφή:**
- **@app.route("/")**: Καθορίζει τη διαδρομή / (αρχική σελίδα) της εφαρμογής Flask.
- **cursor = db.cursor(dictionary=True)**: Δημιουργεί έναν cursor για την εκτέλεση SQL εντολών. Το dictionary=True επιστρέφει αποτελέσματα ως dictionaries.
- **cursor.execute()**: Εκτελεί την SQL ερώτηση που ανακτά δεδομένα από τους πίνακες sensor_data και nodes.
    - **JOIN**: Συνδυάζει τα δεδομένα από τους δύο πίνακες, ώστε να εμφανίζεται το όνομα του κόμβου μαζί με τα δεδομένα του.
    - **ORDER BY**: Τα δεδομένα ταξινομούνται κατά χρονική σειρά (πιο πρόσφατα πρώτα).
- **data = cursor.fetchall()**: Ανακτά όλα τα αποτελέσματα της ερώτησης σε μια λίστα από dictionaries.
- **render_template**: Επιστρέφει τα δεδομένα στην HTML σελίδα sensor_data.html.

Η διαδρομή αυτή είναι υπεύθυνη για την προβολή των δεδομένων αισθητήρων.

---

5. Διαδρομή για Προβολή Αποκλεισμένων IPs

```
@app.route("/blocked_ips")
def blocked_ips():
    cursor = db.cursor(dictionary=True)
    cursor.execute("SELECT * FROM blocked_ips ORDER BY timestamp DESC")
    ips = cursor.fetchall()
    return render_template("blocked_ips.html", ips=ips)
```

- **Περιγραφή:**
- **@app.route("/blocked_ips")**: Καθορίζει τη διαδρομή /blocked_ips.
- **cursor.execute()**: Εκτελεί μια SQL ερώτηση για την ανάκτηση όλων των αποκλεισμένων IPs από τον πίνακα blocked_ips.
- **ips = cursor.fetchall()**: Επιστρέφει τα αποτελέσματα της ερώτησης.
- **render_template**: Στέλνει τα δεδομένα στην HTML σελίδα blocked_ips.html.

Αυτή η διαδρομή εμφανίζει όλες τις αποκλεισμένες διευθύνσεις IP.

---

6. Εκκίνηση της Εφαρμογής Flask

```
if __name__ == "__main__":
    app.run(debug=True)
```

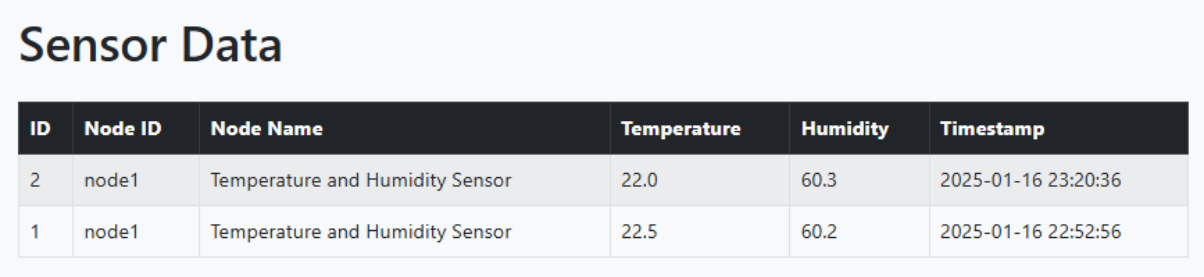
- **Περιγραφή:**
- **if name == "main"**: Εξασφαλίζει ότι ο κώδικας θα εκτελεστεί μόνο αν το αρχείο εκτελεστεί άμεσα.
- **app.run(debug=True)**: Εκκινεί την εφαρμογή Flask σε λειτουργία debugging, η οποία εμφανίζει σφάλματα στην κονσόλα για εύκολη αποσφαλμάτωση.

---

7. HTML Templates - Παράρτημα Α

sensor_data.html

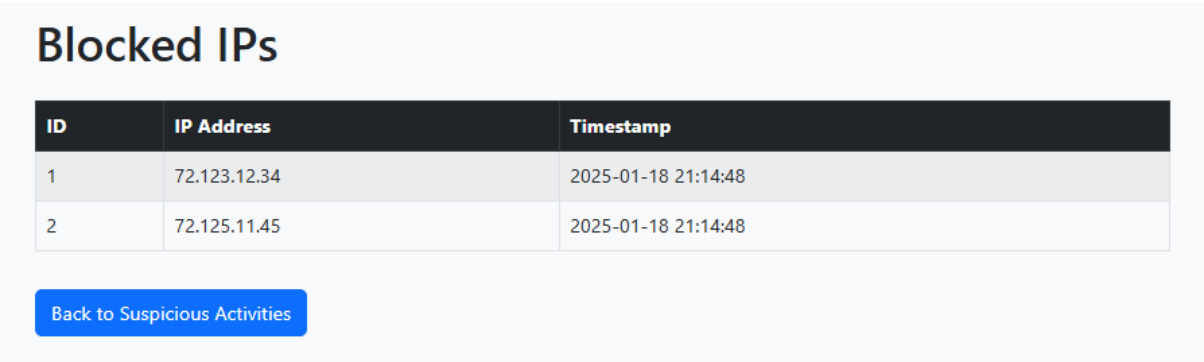
Αυτή η σελίδα εμφανίζει τα δεδομένα που συλλέγονται από τους αισθητήρες των κόμβων IoT. Παρουσιάζει πληροφορίες όπως το αναγνωριστικό της εγγραφής, το αναγνωριστικό και το όνομα του κόμβου, τις μετρήσεις θερμοκρασίας και υγρασίας, καθώς και τη χρονική στιγμή καταγραφής των δεδομένων. Αποτελεί βασικό εργαλείο για την παρακολούθηση των μετρήσεων του συστήματος και την ανάλυση δεδομένων.



Εικόνα 4.1: Σελίδα προβολής των δεδομένων

blocked_ips.html

Αυτή η σελίδα παρουσιάζει τις διευθύνσεις IP που έχουν αποκλειστεί από το σύστημα λόγω ύποπτης ή μη εξουσιοδοτημένης δραστηριότητας. Περιλαμβάνει πληροφορίες όπως το αναγνωριστικό της εγγραφής, τη διεύθυνση IP και τη χρονική στιγμή του αποκλεισμού. Αυτή η λειτουργία βοηθά στη διαχείριση της ασφάλειας του δικτύου και αποτρέπει επαναλαμβανόμενες απόπειρες πρόσβασης από κακόβουλες πηγές.



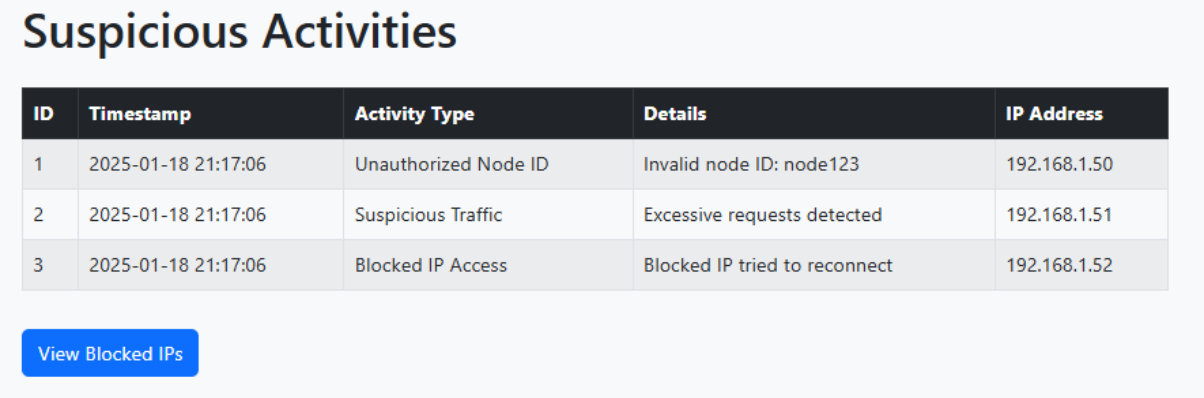
Εικόνα 4.2: Διευθύνσεις IP που έχουν αποκλειστεί από το σύστημα

activities.html

Αυτή η σελίδα εμφανίζει τις καταγεγραμμένες ύποπτες δραστηριότητες που έχουν καταχωρηθεί από το σύστημα. Παρουσιάζει πληροφορίες όπως το αναγνωριστικό της δραστηριότητας (ID), τη χρονική

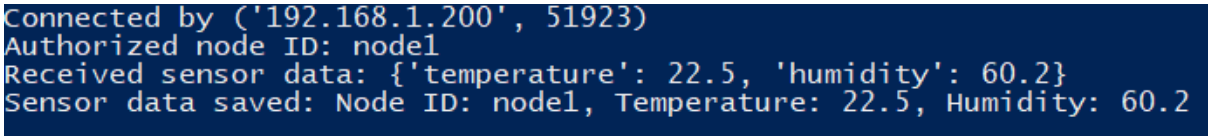
στιγμή της καταγραφής, τον τύπο της δραστηριότητας, λεπτομέρειες για το συμβάν και τη διεύθυνση IP από την οποία προήλθε. Είναι ένα εργαλείο παρακολούθησης για τους διαχειριστές, επιτρέποντάς τους να εντοπίζουν πιθανές απειλές ή ανωμαλίες στο δίκτυο.

Εικόνες από τα sites

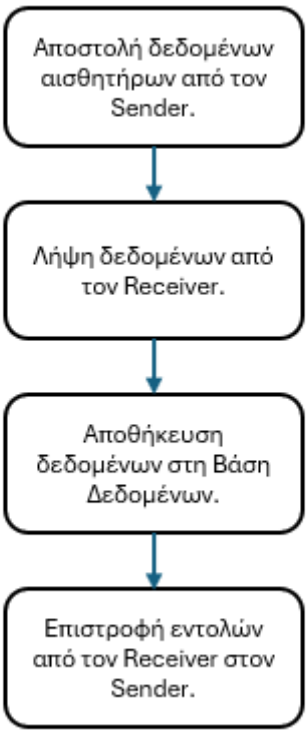


Εικόνα 4.3: Καταγεγραμμένες ύποπτες δραστηριότητες

και το console του receiver όταν δέχεται τιμές από εξουσιοδοτημένο κόμβο



Εικόνα 4.4: Console του receiver όταν δέχεται τιμές από εξουσιοδοτημένο κόμβο

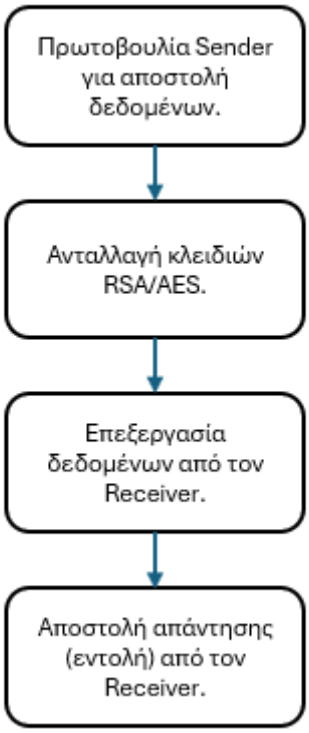


Εικόνα 4.5: Διάγραμμα Ροής Δεδομένων

Στην Εικόνα 4.5 παρουσιάζεται η ροή δεδομένων και οι βασικές λειτουργίες του συστήματος IoT. Το διάγραμμα απεικονίζει τα εξής βασικά στάδια:

1. **Αποστολή Δεδομένων από τον Sender**: Τα δεδομένα που συλλέγονται από τους αισθητήρες (π.χ., θερμοκρασία, υγρασία) κρυπτογραφούνται με τη χρήση αλγορίθμων RSA/AES και αποστέλλονται στον κεντρικό κόμβο (Receiver).
2. **Λήψη Δεδομένων από τον Receiver**: Ο Receiver αποκρυπτογραφεί τα δεδομένα που λαμβάνει, επαληθεύει την ταυτότητα του Sender και τα προετοιμάζει για περαιτέρω επεξεργασία.
3. **Αποθήκευση στη Βάση Δεδομένων**: Τα δεδομένα αισθητήρων αποθηκεύονται στη MySQL βάση δεδομένων, διασφαλίζοντας την ακεραιότητά τους και καθιστώντας τα διαθέσιμα για αναφορές και ανάλυση.
4. **Επιστροφή Εντολών στον Sender**: Ο Receiver, βάσει των δεδομένων που επεξεργάζεται, στέλνει εντολές πίσω στον Sender, όπως η ενεργοποίηση ή απενεργοποίηση ενός relay.

Δείχνει τη ροή δεδομένων μεταξύ του Sender, του Receiver, και της Βάσης Δεδομένων και αντικατοπτρίζει την αλληλεπίδραση των βασικών συστατικών του συστήματος, παρέχοντας μια σαφή εικόνα της ασφάλειας, της λειτουργικότητας και της απόδοσης.

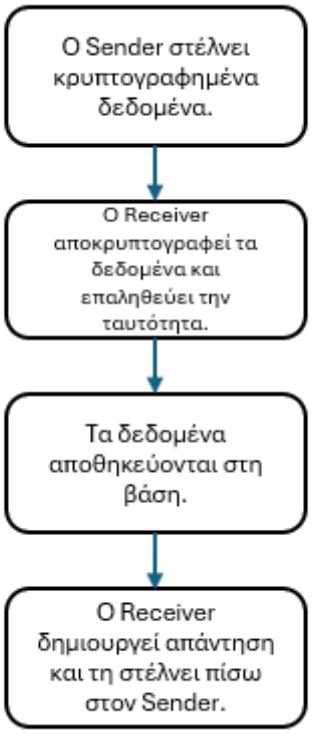


Εικόνα 4.6: Διάγραμμα Αλληλεπίδρασης

Στην Εικόνα 4.6 παρουσιάζεται η διαδικασία επικοινωνίας μεταξύ του Sender και του Receiver, με έμφαση στην ασφάλεια και την επεξεργασία των δεδομένων. Το διάγραμμα απεικονίζει τα εξής βήματα:

1. **Πρωτοβουλία Sender για Αποστολή Δεδομένων**: Ο Sender ξεκινά τη διαδικασία επικοινωνίας, προετοιμάζοντας τα δεδομένα που θα αποσταλούν.
2. **Ανταλλαγή Κλειδιών RSA/AES**: Για να διασφαλιστεί η ασφάλεια της επικοινωνίας, πραγματοποιείται ανταλλαγή κλειδιών. Το δημόσιο κλειδί RSA χρησιμοποιείται για την ασφαλή μετάδοση του AES κλειδιού, το οποίο στη συνέχεια χρησιμοποιείται για την κρυπτογράφηση των δεδομένων.
3. **Επεξεργασία Δεδομένων από τον Receiver**: Ο Receiver αποκρυπτογραφεί τα δεδομένα, επαληθεύει την ταυτότητα του Sender, και προχωρά σε ανάλυση και αποθήκευση.
4. **Αποστολή Απάντησης από τον Receiver**: Μετά την επεξεργασία των δεδομένων, ο Receiver αποστέλλει μια απάντηση (π.χ., εντολή) πίσω στον Sender, ολοκληρώνοντας τη διαδικασία.

Αυτό το διάγραμμα περιγράφει τη βασική ροή επικοινωνίας και την ανταλλαγή πληροφοριών

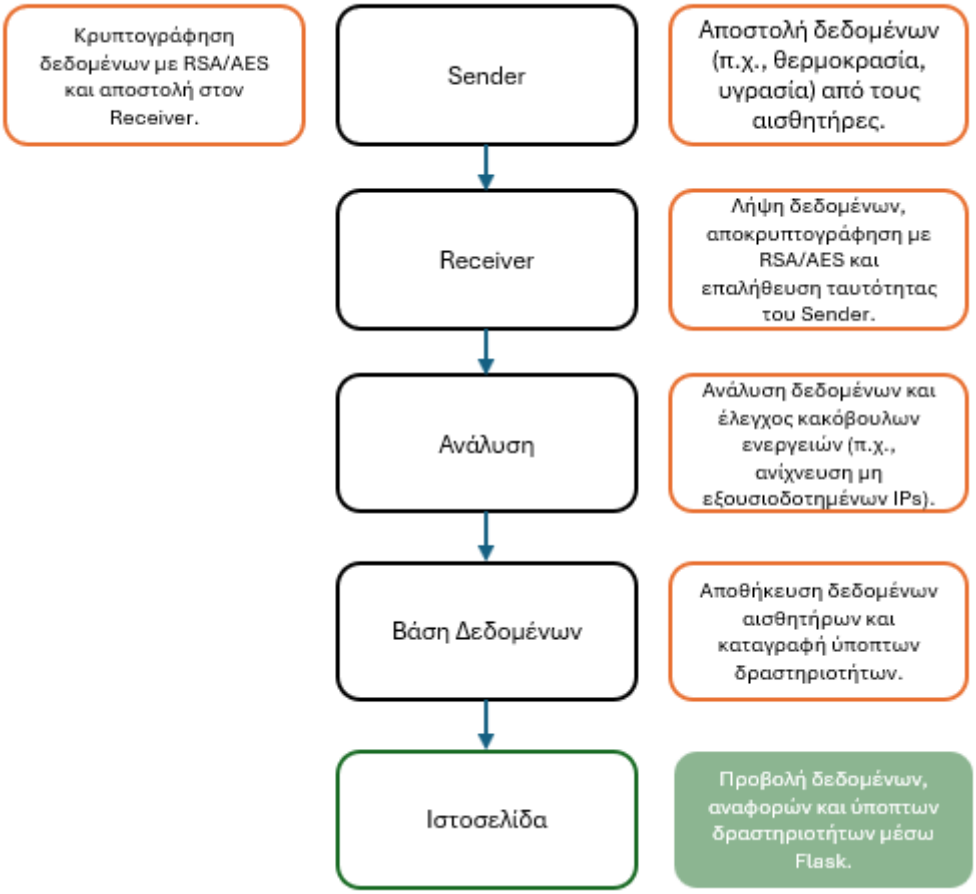


Εικόνα 4.7: Διάγραμμα Διαδικασίας Λήψης και Επεξεργασίας Δεδομένων

Στην Εικόνα 4.7 παρουσιάζεται η διαδικασία ασφαλούς επικοινωνίας και επεξεργασίας δεδομένων στο σύστημα IoT. Το διάγραμμα αναλύει τα παρακάτω βήματα:

1. **Ο Sender στέλνει κρυπτογραφημένα δεδομένα**: Τα δεδομένα που συλλέγονται από τους αισθητήρες του Sender κρυπτογραφούνται με AES/RSA πριν αποσταλούν στον Receiver.
2. **Ο Receiver αποκρυπτογραφεί δεδομένα και επαληθεύει την ταυτότητα**: Ο Receiver αποκρυπτογραφεί τα δεδομένα χρησιμοποιώντας το ιδιωτικό του κλειδί RSA και ελέγχει το αναγνωριστικό (ID) του Sender για να επαληθεύσει αν είναι εξουσιοδοτημένος.
3. **Τα δεδομένα αποθηκεύονται στη βάση**: Μετά την επαλήθευση, τα δεδομένα εισάγονται στον πίνακα της βάσης δεδομένων, όπου καταγράφονται οι μετρήσεις και οποιαδήποτε σχετική πληροφορία.
4. **Ο Receiver δημιουργεί απάντηση και τη στέλνει πίσω στον Sender**: Ο Receiver στέλνει πίσω εντολές ή επιβεβαιώσεις στον Sender, όπως η ενεργοποίηση ή απενεργοποίηση ενός relay.

Περιγράφει τη ροή εργασίας από τη στιγμή που τα δεδομένα αποστέλλονται από τον Sender μέχρι να αποθηκευτούν και να επιστραφεί απάντηση



Εικόνα 4.8: Διάγραμμα που περιγράφει τη ροή δεδομένων και τις λειτουργίες του συστήματος

Στην Εικόνα 4.8 παρουσιάζεται η συνολική αρχιτεκτονική και ροή λειτουργιών του συστήματος IoT, περιλαμβάνοντας τα στάδια επικοινωνίας, επεξεργασίας, αποθήκευσης και προβολής δεδομένων. Αναλυτικά, το διάγραμμα περιγράφει:

1. **Κρυπτογράφηση και Αποστολή από τον Sender**: Τα δεδομένα που συλλέγονται από τους αισθητήρες (π.χ., θερμοκρασία, υγρασία) κρυπτογραφούνται χρησιμοποιώντας RSA/AES και αποστέλλονται στον Receiver.
2. **Λήψη και Αποκρυπτογράφηση από τον Receiver**: Ο Receiver λαμβάνει τα κρυπτογραφημένα δεδομένα, τα αποκρυπτογραφεί και επαληθεύει την ταυτότητα του Sender.

3. **Ανάλυση Δεδομένων και Έλεγχος Κακόβουλων Ενεργειών**: Τα δεδομένα αναλύονται και γίνεται έλεγχος για ύποπτες δραστηριότητες, όπως ανίχνευση μη εξουσιοδοτημένων IPs ή άλλων κακόβουλων ενεργειών.
4. **Αποθήκευση στη Βάση Δεδομένων**: Τα δεδομένα αισθητήρων αποθηκεύονται στη βάση δεδομένων μαζί με πληροφορίες για ύποπτες δραστηριότητες.
5. **Προβολή μέσω Ιστοσελίδας**: Τα αποθηκευμένα δεδομένα παρουσιάζονται μέσω μιας διεπαφής Flask. Οι διαχειριστές μπορούν να δουν μετρήσεις, αναφορές και καταγραφές ύποπτων δραστηριοτήτων.

Αυτό το διάγραμμα παρέχει μια σαφή εικόνα της δομής και της ροής εργασίας του συστήματος, ενσωματώνοντας ασφάλεια, αξιοπιστία και διαχειρισιμότητα.

4.3 Διασύνδεση ενός ESP32 με έναν κόμβο PC/Raspberry Pi μέσω C/C++ και Python.

Για να υλοποιηθεί η επικοινωνία μεταξύ του **ESP32** και ενός **PC** χρησιμοποιώντας τις ίδιες λειτουργίες (ασφάλεια, επικοινωνία, αποστολή/λήψη δεδομένων), μπορούμε να χρησιμοποιήσουμε το **Arduino framework** για το ESP32 και **Python** για τον κεντρικό υπολογιστή (Receiver).

Κώδικας για το ESP32 (Sender)

Χρησιμοποιείται η βιβλιοθήκη **ArduinoJson** για δημιουργία JSON δεδομένων καθώς και βιβλιοθήκες για την κρυπτογράφηση (AES/RSA).

- **ArduinoJson**: Διαχειρίζεται τα JSON δεδομένα.
- **mbedtls**: Παρέχεται εγγενώς στο ESP32 για RSA και AES κρυπτογράφηση.

Τα βασικά σημεία του κώδικα για το **ESP32** είναι:

1. Σύνδεση στο WiFi

```
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.println("Connecting to WiFi...");
```

```
}
Serial.println("Connected to WiFi.");
```

- **Λειτουργία**: Το ESP32 συνδέεται στο δίκτυο WiFi χρησιμοποιώντας τα δεδομένα SSID και κωδικού.

2. Δημιουργία AES Κλειδιού και IV

```
for (int i = 0; i < 32; i++) aesKey[i] = random(0, 256);
for (int i = 0; i < 16; i++) iv[i] = random(0, 256);
```

- **Λειτουργία**: Δημιουργείται ένα τυχαίο κλειδί AES (32 bytes) και ένας τυχαίος IV (Initialization Vector) για την κρυπτογράφηση.

3. Κρυπτογράφηση του AES Κλειδιού με RSA

```
String encryptAESKeyWithRSA() {
  // Χρήση της mbedtls βιβλιοθήκης για την κρυπτογράφηση του AES κλειδιού
}
```

- **Λειτουργία**: Το AES κλειδί κρυπτογραφείται με το δημόσιο RSA κλειδί του Receiver, ώστε να διασφαλιστεί η ασφαλής αποστολή του.

4. Δημιουργία JSON Δεδομένων

```
DynamicJsonDocument jsonDoc(1024);
jsonDoc["id"] = "esp32_node_1";
jsonDoc["temperature"] = temperature;
jsonDoc["humidity"] = humidity;
String jsonString;
serializeJson(jsonDoc, jsonString);
```

- **Λειτουργία**: Τα δεδομένα αισθητήρων (π.χ., θερμοκρασία, υγρασία) μετατρέπονται σε μορφή JSON για αποστολή.

5. Κρυπτογράφηση Δεδομένων με AES

```
void encryptData(String plainText, unsigned char* output, size_t& outputLen) {
  // Χρήση της mbedtls βιβλιοθήκης για κρυπτογράφηση AES
}
```

- **Λειτουργία**: Το JSON κείμενο κρυπτογραφείται με το AES κλειδί και το IV.

6. Αποστολή Δεδομένων στον Receiver

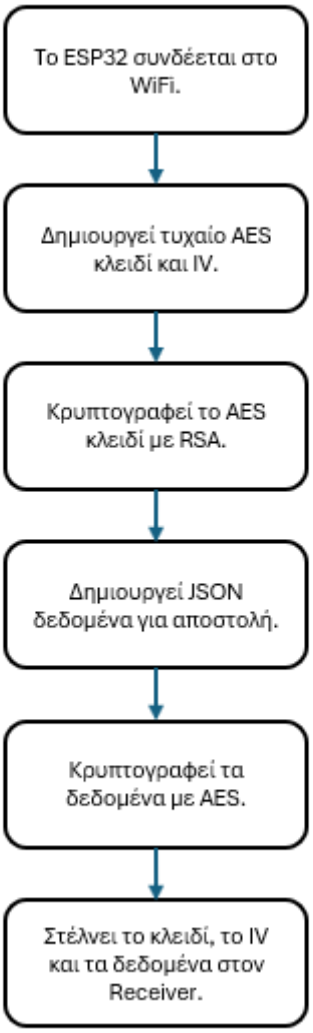
```
client.println(encryptedAESKey); // Αποστολή του κρυπτογραφημένου AES κλειδιού
client.write(iv, 16);            // Αποστολή του IV
client.write(encryptedData, encryptedDataLen); // Αποστολή των κρυπτογραφημένων δεδομένων
```

- **Λειτουργία**: Στέλνονται στον Receiver:
    1. Το κρυπτογραφημένο AES κλειδί.
    2. Το IV.
    3. Τα κρυπτογραφημένα δεδομένα αισθητήρων.

7. Αποσφαλμάτωση

```
Serial.println("Data sent!");
```

- **Λειτουργία**: Ενημερώνει για την επιτυχή αποστολή των δεδομένων.



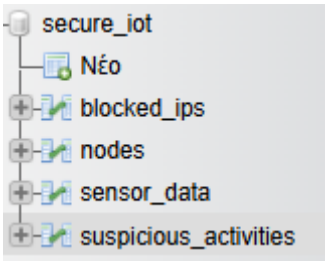
Εικόνα 4.9: Διάγραμμα Ροής Λειτουργιών ESP32 - Ασφαλής Επικοινωνία με Receiver

Το ESP32:

1. Συνδέεται στο WiFi δίκτυο για να αποκτήσει πρόσβαση στον Receiver.
2. Δημιουργεί τυχαίο AES κλειδί και IV, τα οποία θα χρησιμοποιηθούν για την κρυπτογράφηση των δεδομένων.
3. Κρυπτογραφεί το AES κλειδί με RSA, διασφαλίζοντας ότι μόνο ο Receiver μπορεί να το αποκρυπτογραφήσει.
4. Δημιουργεί δεδομένα σε μορφή JSON, τα οποία περιλαμβάνουν τις μετρήσεις των αισθητήρων (π.χ., θερμοκρασία, υγρασία).
5. Κρυπτογραφεί τα δεδομένα JSON με το AES κλειδί, εξασφαλίζοντας την ασφάλεια των πληροφοριών.

6. Στέλνει στον Receiver το κρυπτογραφημένο AES κλειδί, το IV και τα κρυπτογραφημένα δεδομένα.

4.4 Η βάση δεδομένων



Εικόνα 4.10: Η βάση με τους πίνακες που χρησιμοποιήθηκαν για το έργο

Η βάση δεδομένων αποτελεί την καρδιά του συστήματος IoT που υλοποιήθηκε, καθώς σε αυτήν αποθηκεύονται τα δεδομένα που συλλέγονται από τους αισθητήρες, οι καταγραφές ύποπτων δραστηριοτήτων και άλλες σχετικές πληροφορίες. Η σωστή σχεδίαση και δομή της βάσης εξασφαλίζει τη λειτουργικότητα, την επεκτασιμότητα και την αξιοπιστία του συστήματος.

Η βάση δεδομένων εξυπηρετεί πολλαπλούς σκοπούς:

1. **Αποθήκευση Δεδομένων Αισθητήρων**: Καταγραφή μετρήσεων, όπως θερμοκρασία και υγρασία, που αποστέλλονται από τους Senders.
2. **Καταγραφή Υποπτων Δραστηριοτήτων**: Παρακολούθηση και αρχειοθέτηση ύποπτων IP διευθύνσεων ή μη εξουσιοδοτημένων ενεργειών.
3. **Επαλήθευση Ταυτότητας Κόμβων**: Διατήρηση λίστας εγκεκριμένων κόμβων που ανήκουν στο δίκτυο.
4. **Υποστήριξη Αναφορών**: Παροχή δεδομένων για τη δημιουργία αναφορών μέσω της ιστοσελίδας.

Δομή και Σχεδίαση της Βάσης

Η βάση δεδομένων έχει υλοποιηθεί σε MySQL και αποτελείται από τρεις κύριους πίνακες, καθένας από τους οποίους εξυπηρετεί διαφορετική λειτουργία.

Πίνακας nodes

Σκοπός: Αποθηκεύει πληροφορίες για τους κόμβους (Senders) που συμμετέχουν στο δίκτυο.

Δομή:

Στήλη	Τύπος	Περιγραφή
id	INT	Πρωτεύον Κλειδί, μοναδικό αναγνωριστικό κόμβου.
name	VARCHAR(255)	Όνομα του κόμβου.
description	TEXT	Περιγραφή του κόμβου (προαιρετικά).

Παραδείγματα Εγγραφών:

id	name	description
1	Node_1	Αισθητήρας θερμοκρασίας.
2	Node_2	Αισθητήρας υγρασίας.

Πίνακας sensor_data

Σκοπός: Καταγραφή των δεδομένων που συλλέγονται από τους κόμβους.

Δομή:

Στήλη	Τύπος	Περιγραφή
id	INT	Πρωτεύον Κλειδί, μοναδικό αναγνωριστικό εγγραφής.
node_id	INT	Ξένο Κλειδί προς τον πίνακα nodes.
temperature	FLOAT	Θερμοκρασία που συλλέχθηκε.
humidity	FLOAT	Υγρασία που συλλέχθηκε.
timestamp	DATETIME	Ημερομηνία και ώρα καταγραφής.

Παραδείγματα Εγγραφών:

id	node_id	temperature	humidity	timestamp
1	1	22.5	NULL	2025-01-18 10:15:00
2	2	NULL	60.2	2025-01-18 10:16:00

Πίνακας suspicious_activities

Σκοπός: Αρχειοθέτηση ύποπτων δραστηριοτήτων, όπως προσπάθειες σύνδεσης από μη εξουσιοδοτημένα IPs.

Δομή:

Στήλη	Τύπος	Περιγραφή
id	INT	Πρωτεύον Κλειδί, μοναδικό αναγνωριστικό εγγραφής.
activity_type	VARCHAR(255)	Τύπος δραστηριότητας (π.χ., μη εξουσιοδοτημένο Node ID).
details	TEXT	Λεπτομέρειες για τη δραστηριότητα.
ip_address	VARCHAR(45)	IP διεύθυνση που προκάλεσε τη δραστηριότητα.
timestamp	DATETIME	Χρονική στιγμή καταγραφής.

Παραδείγματα Εγγραφών:

id	activity_type	details	ip_address	timestamp
1	Unauthorized Node ID	Invalid node ID: Node_123	192.168.1.50	2025-01-18 10:20:00
2	Blocked IP Access	Blocked IP attempted to reconnect	192.168.1.51	2025-01-18 10:25:00

Ερωτήματα για τη Βάση

1. **Ανάκτηση όλων των δεδομένων αισθητήρων**:

```
SELECT * FROM sensor_data ORDER BY timestamp DESC;
```

2. **Εύρεση ύποπτων δραστηριοτήτων συγκεκριμένης IP**:

```
SELECT * FROM suspicious_activities WHERE ip_address = '192.168.1.50';
```

3. **Εισαγωγή νέου κόμβου**:

```
INSERT INTO nodes (name, description) VALUES ('Node_3', 'Αισθητήρας πίεσης.');
```

4. **Καταγραφή νέας ύποπτης δραστηριότητας**:

```
INSERT INTO suspicious_activities (activity_type, details, ip_address)
VALUES ('Unauthorized Access', 'Invalid node ID: Node_456', '192.168.1.100');
```

Επεκτασιμότητα

Η βάση έχει σχεδιαστεί με γνώμονα την επεκτασιμότητα, ώστε να μπορεί να υποστηρίξει:

- Προσθήκη περισσότερων κόμβων και αισθητήρων.
- Διαχείριση περισσότερων τύπων ύποπτων δραστηριοτήτων.
- Ενσωμάτωση νέων πινάκων για αποθήκευση επιπλέον δεδομένων (π.χ., ιστορικό εντολών προς Senders).

Η βάση δεδομένων παρέχει τη θεμελιώδη υποδομή για την ασφαλή αποθήκευση και ανάκτηση δεδομένων στο σύστημα IoT. Με τη σωστή διαχείριση και οργάνωση, εξασφαλίζεται η λειτουργικότητα του συστήματος, η ανάλυση των δεδομένων και η προστασία του από κακόβουλες δραστηριότητες.

4.5 Ασφάλεια στο σύστημα και στα δεδομένα

Η ασφάλεια κατά την επικοινωνία μεταξύ των κόμβων (Senders) και του κεντρικού server εξασφαλίζεται μέσω ενός υβριδικού μοντέλου κρυπτογράφησης. Το μοντέλο αυτό συνδυάζει την ασύμμετρη κρυπτογράφηση RSA με την συμμετρική AES, παρέχοντας τα εξής πλεονεκτήματα:

1. **Κρυπτογράφηση Δεδομένων με AES**:
    - Τα δεδομένα που συλλέγονται από τους αισθητήρες κρυπτογραφούνται με AES (Advanced Encryption Standard) χρησιμοποιώντας ένα μοναδικό κλειδί για κάθε επικοινωνία.
    - Ο AES είναι γνωστός για την ταχύτητα και την ασφάλειά του, καθιστώντας τον ιδανικό για κρυπτογράφηση μεγάλου όγκου δεδομένων.
2. **Ανταλλαγή Κλειδιών με RSA**:
    - Το AES κλειδί κρυπτογραφείται με τον αλγόριθμο RSA, διασφαλίζοντας ότι μόνο ο Receiver, που κατέχει το ιδιωτικό του κλειδί, μπορεί να το αποκρυπτογραφήσει.
    - Ο RSA χρησιμοποιείται για την ασφαλή μεταφορά του AES κλειδιού, αποτρέποντας την υποκλοπή του κατά τη μεταφορά.
3. **Επαλήθευση Ταυτότητας**:
    - Κάθε Sender περιέχει ένα μοναδικό αναγνωριστικό (ID) που αποστέλλεται μαζί με τα δεδομένα.
    - Ο Receiver επαληθεύει το ID, διασφαλίζοντας ότι μόνο εξουσιοδοτημένοι κόμβοι μπορούν να επικοινωνούν με το σύστημα.

4. **Ακεραιότητα Δεδομένων**:
    - Με τη χρήση κρυπτογράφησης, διασφαλίζεται ότι τα δεδομένα δεν μπορούν να τροποποιηθούν κατά τη μεταφορά, χωρίς να ανιχνευθεί η παρέμβαση.

Έλεγχος και Ανίχνευση Κακόβουλων Ενεργειών

Το σύστημα περιλαμβάνει μηχανισμούς ανίχνευσης και αντιμετώπισης κακόβουλων δραστηριοτήτων:

1. **Μη Εξουσιοδοτημένες Συνδέσεις**:
    - Εάν ένας κόμβος επιχειρήσει να συνδεθεί με το σύστημα χωρίς έγκυρο ID, η προσπάθεια καταγράφεται ως ύποπτη δραστηριότητα.
    - Η διεύθυνση IP της μη εξουσιοδοτημένης συσκευής καταχωρείται στον πίνακα suspicious_activities.
2. **Αποκλεισμός Κακόβουλων IPs**:
    - Οι IP διευθύνσεις που παρουσιάζουν κακόβουλη συμπεριφορά (π.χ., υπερβολικός αριθμός αιτημάτων) προστίθενται στη λίστα αποκλεισμένων IPs.
    - Αυτό αποτρέπει επαναλαμβανόμενες απόπειρες πρόσβασης από την ίδια πηγή.
3. **Καταγραφή και Αναφορά**:
    - Όλες οι ύποπτες δραστηριότητες καταγράφονται στη βάση δεδομένων, παρέχοντας στους διαχειριστές λεπτομερείς πληροφορίες για ανάλυση και αναφορά.

Ασφάλεια στην Αποθήκευση Δεδομένων

Η βάση δεδομένων είναι σχεδιασμένη να διαχειρίζεται με ασφάλεια όλα τα δεδομένα του συστήματος. Τα μέτρα που λαμβάνονται περιλαμβάνουν:

1. **Διασφάλιση Ακεραιότητας**:
    - Κάθε εγγραφή στη βάση δεδομένων συνοδεύεται από χρονική σήμανση (timestamp), παρέχοντας σαφή καταγραφή του πότε συλλέχθηκαν ή καταχωρήθηκαν τα δεδομένα.
2. **Περιορισμένη Πρόσβαση**:
    - Η πρόσβαση στη βάση δεδομένων είναι περιορισμένη μόνο σε εξουσιοδοτημένους χρήστες και εφαρμογές, μειώνοντας τον κίνδυνο μη εξουσιοδοτημένων αλλαγών.
3. **Κρυπτογράφηση Ευαίσθητων Δεδομένων**:

- Αν κριθεί απαραίτητο, ευαίσθητα δεδομένα μπορούν να κρυπτογραφηθούν μέσα στη βάση, παρέχοντας επιπλέον προστασία ακόμα και σε περίπτωση παραβίασης.

Η ασφάλεια στο σύστημα και στα δεδομένα επιτυγχάνεται μέσω της προσεκτικής σχεδίασης και της ενσωμάτωσης προηγμένων τεχνολογιών κρυπτογράφησης και ανίχνευσης. Το υβριδικό μοντέλο RSA/AES διασφαλίζει την ασφαλή επικοινωνία, ενώ οι μηχανισμοί ανίχνευσης και καταγραφής παρέχουν μια ολοκληρωμένη προσέγγιση για την προστασία του συστήματος. Με αυτές τις στρατηγικές, το σύστημα είναι ανθεκτικό στις κακόβουλες δραστηριότητες και εξασφαλίζει την ακεραιότητα και την εμπιστευτικότητα των δεδομένων.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Στην παρούσα εργασία αναλύθηκε η ανάπτυξη ενός ασφαλούς συστήματος επικοινωνίας συσκευών IoT, το οποίο βασίζεται στη χρήση κρυπτογράφησης, επαλήθευσης ταυτότητας και ανίχνευσης ύποπτων δραστηριοτήτων. Στόχος ήταν η ασφαλής μετάδοση δεδομένων από απομακρυσμένους κόμβους (Senders) στον κεντρικό κόμβο (Receiver) και η προβολή τους σε ένα εύχρηστο web περιβάλλον μέσω Flask.

Η μελέτη περιέγραψε τη χρήση τεχνολογιών όπως η Python, το Flask, η MySQL και η υβριδική κρυπτογράφηση RSA/AES για την υλοποίηση του συστήματος. Η βάση δεδομένων σχεδιάστηκε για να υποστηρίζει την αποθήκευση δεδομένων αισθητήρων, την καταγραφή ύποπτων ενεργειών και τη διαχείριση εγκεκριμένων κόμβων. Παράλληλα, δόθηκε έμφαση στην ανθεκτικότητα του συστήματος απέναντι σε μη εξουσιοδοτημένες συνδέσεις και κακόβουλες ενέργειες.

Η υλοποίηση αυτή αναδεικνύει τις δυνατότητες που προσφέρει το IoT σε εφαρμογές πραγματικού χρόνου, ενώ ταυτόχρονα εξασφαλίζει την εμπιστευτικότητα και την ακεραιότητα των δεδομένων. Η χρήση ενός υβριδικού μοντέλου κρυπτογράφησης προσφέρει ισχυρή προστασία τόσο στην ανταλλαγή δεδομένων όσο και στην ταυτοποίηση των κόμβων.

Το σύστημα που υλοποιήθηκε παρέχει ασφαλή επικοινωνία, συνδυάζοντας την αποτελεσματικότητα της συμμετρικής κρυπτογράφησης AES με την ισχυρή προστασία της ασύμμετρης κρυπτογράφησης RSA.

Η χρήση μοναδικών αναγνωριστικών (IDs) για τους κόμβους εξασφαλίζει την αυθεντικότητα των συσκευών που συμμετέχουν στο δίκτυο.

Ο μηχανισμός καταγραφής ύποπτων ενεργειών ενισχύει την ασφάλεια του συστήματος, προλαμβάνοντας πιθανές επιθέσεις.

Η ενσωμάτωση του Flask προσφέρει ένα φιλικό web περιβάλλον για την παρακολούθηση δεδομένων και τη διαχείριση εντολών.

Αξίζει να αναφερθεί ότι το chat gpt αξιοποιήθηκε σε διάφορα σημεία της εργασίας για τη δημιουργία και τη διόρθωση του συντακτικού και για τη σωστή μορφοποίηση του κειμένου.

Υπάρχουν περιθώρια βελτίωσης για την ενίσχυση της λειτουργικότητας και της ασφάλειας:

- Χρήση πιο προηγμένων αλγορίθμων κρυπτογράφησης ή μεγαλύτερων κλειδιών (π.χ., 4096-bit RSA) για την ενίσχυση της προστασίας.
- Υλοποίηση HTTPS για την επικοινωνία με το web interface.
- Δημιουργία δυναμικού μηχανισμού προσθήκης ή αφαίρεσης εγκεκριμένων κόμβων μέσω της διεπαφής Flask.

- Ενσωμάτωση ειδοποιήσεων για μη εξουσιοδοτημένες προσπάθειες σύνδεσης.
- Υλοποίηση εργαλείων ανάλυσης δεδομένων σε πραγματικό χρόνο για τη βελτίωση της παρακολούθησης και της λήψης αποφάσεων.
- Ενσωμάτωση γραφημάτων για την οπτικοποίηση των δεδομένων στη διεπαφή.
- Δυνατότητα προσθήκης περισσότερων κόμβων ή τύπων αισθητήρων.
- Εφαρμογή τεχνητής νοημοσύνης για την ανίχνευση μοτίβων που υποδεικνύουν κακόβουλες ενέργειες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] P. V. Dudhe, N. V. Kadam, R. M. Hushangabade and M. S. Deshmukh, "Internet of Things (IOT): An overview and its applications," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 2017, pp. 2650-2653, doi: 10.1109/ICECDS.2017.8389935.

[2] "An Overview of Enabling Technologies for the Internet of Things," in Internet of Things A to Z: Technologies and Applications , IEEE, 2018, pp.77-112, doi: 10.1002/9781119456735.ch3.

[3] I. Ud Din et al., "The Internet of Things: A Review of Enabled Technologies and Future Challenges," in IEEE Access, vol. 7, pp. 7606-7640, 2019, doi: 10.1109/ACCESS.2018.2886601.

[4] Dritsas, E., & Trigka, M. (2025). A Survey on Cybersecurity in IoT. Future Internet, 17(1), 30. https://doi.org/10.3390/fi17010030

[5] Mahmoud, R., Yousuf, T., Aloul, F., & Zualkernan, I. (2015, December). Internet of things (IoT) security: Current status, challenges and prospective measures. In 2015 10th international conference for internet technology and secured transactions (ICITST) (pp. 336-341). IEEE.

[6] Jurcut, A. D., Ranaweera, P., & Xu, L. (2020). Introduction to IoT security. IoT security: advances in authentication, 27-64.

[7] Rautmare, S., & Bhalerao, D. M. (2016, October). MySQL and NoSQL database comparison for IoT application. In 2016 IEEE international conference on advances in computer applications (ICACA) (pp. 235-238). IEEE.

[8] Ray, P. P. (2016). A survey of IoT cloud platforms. Future Computing and Informatics Journal, 1(1-2), 35-46.

[9] https://www.python.org/

[10] https://www.w3schools.com/python/

[11] https://en.wikipedia.org/wiki/Python_(programming_language)

[12] https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

[13] https://www.geeksforgeeks.org/python-language-advantages-applications/

[14] https://en.wikipedia.org/wiki/Flask_(web_framework)

[15] https://careerfoundry.com/en/blog/web-development/what-is-flask/

[16] https://www.mytaskpanel.com/what-is-flask/

[17] https://en.wikipedia.org/wiki/MySQL

[18] https://blueclawdb.com/mysql/advantages-disadvantages-mysql/

[19] https://www.oracle.com/mysql/what-is-mysql/

[20] https://www.precisely.com/blog/data-security/aes-vs-rsa-encryption-differences

[21] https://www.geeksforgeeks.org/difference-between-aes-and-rsa-encryption/

[22] https://phalanx.io/aes-vs-rsa-encryption/

[23] https://medium.com/@prateekbansalind/ssl-tls-3-aes-and-rsa-63d356b3ebf7

[24] https://www.mdpi.com/2673-4591/20/1/14

[25] https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[26] https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

ΠΑΡΑΡΤΗΜΑ Α

Κώδικας esp32

```
#include <WiFi.h>
#include <ArduinoJson.h>
#include <mbedtls/aes.h>
#include <mbedtls/rsa.h>

// WiFi credentials
const char* ssid = "Your_SSID";
const char* password = "Your_PASSWORD";

// Server details
const char* serverIP = "192.168.1.200"; // IP του Receiver
const int serverPort = 65432;

// Keys (Dummy values, αντικαταστήστε με τα πραγματικά κλειδιά RSA)
const char* publicKey = "-----BEGIN PUBLIC KEY-----\n...YOUR_PUBLIC_KEY...\n-----END PUBLIC KEY-----";

// Data to send
float temperature = 22.5;
float humidity = 60.2;

// AES key and IV
unsigned char aesKey[32] = {0};
unsigned char iv[16] = {0};

// WiFi client
WiFiClient client;

// Function to generate random AES key and IV
void generateAESKeyAndIV() {
  for (int i = 0; i < 32; i++) aesKey[i] = random(0, 256);
  for (int i = 0; i < 16; i++) iv[i] = random(0, 256);
}

// Function to encrypt AES key with RSA
String encryptAESKeyWithRSA() {
  mbedtls_rsa_context rsa;
  mbedtls_rsa_init(&rsa, MBEDTLS_RSA_PKCS_V15, 0);

  // Load public key
  mbedtls_pk_context pk;
  mbedtls_pk_init(&pk);
  mbedtls_pk_parse_public_key(&pk, (const unsigned char*)publicKey, strlen(publicKey) + 1);
  rsa = *mbedtls_pk_rsa(pk);
```

```
  // Encrypt AES key
  unsigned char encryptedKey[256];
  size_t encryptedKeyLen = 0;
  mbedtls_rsa_rsaes_pkcs1_v15_encrypt(&rsa, NULL, NULL, MBEDTLS_RSA_PUBLIC, 32, aesKey, encryptedKey);

  // Return as Base64 (για απλότητα)
  char base64Output[512];
  size_t outputLen;
  mbedtls_base64_encode((unsigned char*)base64Output, sizeof(base64Output), &outputLen, encryptedKey,
rsa.len);

  return String(base64Output);
}

// Function to encrypt data with AES
void encryptData(String plainText, unsigned char* output, size_t& outputLen) {
  mbedtls_aes_context aes;
  mbedtls_aes_init(&aes);
  mbedtls_aes_setkey_enc(&aes, aesKey, 256);
  mbedtls_aes_crypt_cfb128(&aes, MBEDTLS_AES_ENCRYPT, plainText.length(), iv, (unsigned
char*)plainText.c_str(), output);
  outputLen = plainText.length();
}

// Setup
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi.");

  if (!client.connect(serverIP, serverPort)) {
    Serial.println("Connection to server failed!");
    return;
  }
  Serial.println("Connected to server.");

  // Generate AES key and IV
  generateAESKeyAndIV();
  String encryptedAESKey = encryptAESKeyWithRSA();

  // Prepare JSON data
  DynamicJsonDocument jsonDoc(1024);
  jsonDoc["id"] = "esp32_node_1";
  jsonDoc["temperature"] = temperature;
```

```
  jsonDoc["humidity"] = humidity;

  String jsonString;
  serializeJson(jsonDoc, jsonString);

  // Encrypt JSON data
  unsigned char encryptedData[1024];
  size_t encryptedDataLen;
  encryptData(jsonString, encryptedData, encryptedDataLen);

  // Send encrypted AES key, IV, and data
  client.println(encryptedAESKey);
  client.write(iv, 16);
  client.write(encryptedData, encryptedDataLen);
  Serial.println("Data sent!");
}

// Loop
void loop() {
  // No actions in loop
}
```

Κώδικας Python Receiver για επικοινωνία με τον esp32

```
import socket
import json
from base64 import b64decode
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.primitives.asymmetric import rsa

# Server RSA private key (Dummy value, αντικαταστήστε με το πραγματικό)
private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)

# Socket setup
HOST = '192.168.1.200'
PORT = 65432
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((HOST, PORT))
server_socket.listen(1)
```

```
print("Waiting for connection...")
conn, addr = server_socket.accept()
print(f"Connected by {addr}")

# Receive encrypted AES key and IV
encrypted_aes_key = conn.recv(512).decode()
iv = conn.recv(16)

# Decrypt AES key
aes_key = private_key.decrypt(
    b64decode(encrypted_aes_key),
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)

# Receive and decrypt data
encrypted_data = conn.recv(1024)
cipher = Cipher(algorithms.AES(aes_key), modes.CFB(iv))
decryptor = cipher.decryptor()
plain_data = decryptor.update(encrypted_data).decode()

# Parse JSON
data = json.loads(plain_data)
print("Received Data:", data)
```

sensor_data.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Sensor Data</title>
    <link         href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
    <div class="container mt-5">
        <h1 class="mb-4">Sensor Data</h1>
        <table class="table table-bordered table-striped">
            <thead class="table-dark">
                <tr>
                    <th>ID</th>
                    <th>Node ID</th>
                    <th>Node Name</th>
                    <th>Temperature</th>
                    <th>Humidity</th>
                    <th>Timestamp</th>
                </tr>
            </thead>
            <tbody>
                {% for row in data %}
                <tr>
                    <td>{{ row.id }}</td>
                    <td>{{ row.node_id }}</td>
                    <td>{{ row.name }}</td>
                    <td>{{ row.temperature }}</td>
                    <td>{{ row.humidity }}</td>
                    <td>{{ row.timestamp }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
</body>
</html>
```

blocked_ips.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Blocked IPs</title>
    <link        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="bg-light">
    <div class="container mt-5">
        <h1 class="mb-4">Blocked IPs</h1>
        <table class="table table-bordered table-striped">
            <thead class="table-dark">
                <tr>
                    <th>ID</th>
                    <th>IP Address</th>
                    <th>Timestamp</th>
                </tr>
            </thead>
            <tbody>
                {% for ip in ips %}
                <tr>
                    <td>{{ ip.id }}</td>
                    <td>{{ ip.ip_address }}</td>
                    <td>{{ ip.timestamp }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        <a href="/" class="btn btn-primary mt-3">Back to Suspicious Activities</a>
    </div>
</body>
</html>
```

activities.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Suspicious Activities</title>
    <link        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
```

```
<body class="bg-light">
    <div class="container mt-5">
        <h1 class="mb-4">Suspicious Activities</h1>
        <table class="table table-bordered table-striped">
            <thead class="table-dark">
                <tr>
                    <th>ID</th>
                    <th>Timestamp</th>
                    <th>Activity Type</th>
                    <th>Details</th>
                    <th>IP Address</th>
                </tr>
            </thead>
            <tbody>
                {% for activity in activities %}
                <tr>
                    <td>{{ activity.id }}</td>
                    <td>{{ activity.timestamp }}</td>
                    <td>{{ activity.activity_type }}</td>
                    <td>{{ activity.details }}</td>
                    <td>{{ activity.ip_address }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
        <a href="/blocked_ips" class="btn btn-primary mt-3">View Blocked IPs</a>
    </div>
</body>
</html>
```