DIT172 | Report on Project-1 | itp22104 | 22/12/2023

Γενικά

Αντικειμενοστραφής Σχεδιασμός/Υλοποίηση

Ορίζουμε την κλάση `Person` που αντιπροσωπεύει κάθε ένα από τα άτομα που συμμετέχουν στην ανταλλαγή μηνυμάτων. Η κλάση έχει μεθόδους που αντιστοιχούν στα βήματα της διαδικασίας ανταλλαγής μηνυμάτων τα οποία θα πραγματοποιήσει η μία ή άλλη πλευρά.

Χρήση Βιβλιοθήκης

Ο κώδικας χρησιμοποιεί τη βιβλιοθήκη κρυπτογραφίας PyNaCl και ειδικότερα κλάσεις/μεθόδους όπως οι `PrivateKey, Box, SecretBox, blake2b` για τη δημιουργία κλειδιών, την κρυπτογράφηση/αποκρυπτογράφηση, έλεγχο ακεραιότητας κλπ.

Ειδικότερα

Σχεδιασμός: `Person` class

Αναπαριστά την εκάστοτε πλευρά η οποία συμμετέχει στη διαδικασία και ορίζει attributes τα οποία κρατούν τις απαιτούμενες πληροφορίες για τις ενέργειες που θα πραγματοποιήσει οι οποίες υλοποιούνται από τις μεθόδους της.

Συγκεκριμένα:

- *Attributes*
  - `self.name` → Το όνομα της εκάστοτε πλευράς
  - `self.private` → Το ιδιωτικό κλειδί της εκάστοτε πλευράς
  - `self.public, self.other_public` → Το δημόσιο κλειδί της μιας και της άλλης πλευράς
  - `self.common_secret` → Το κοινό μυστικό K
  - `self.derived_key` → Το παραγόμενο από την KDF κλειδί K΄
  - `self.encrypted_message` → Το κρυπτογραφημένο μήνυμα
  - `self.message` → Το αρχικό προς ανταλαγή μήνυμα
  - `self.tag, self.other_tag` → Η ετικέτα της μιας και της άλλης πλευράς
- *Methods*
  - `get_other_public()`
    Αποθηκεύει στο instance της `Person` το δημόσιο κλειδί της άλλης πλευράς
  - `make_key_pair()`
    Δημιουργεί και αποθηκεύει στο instance της `Person` ένα ζεύγος δημοσίου/ιδιωτικού κλειδιού με χρήση της `PrivateKey` της `PyNaCl`.
    Χρησιμοποιείται ο αλγόριθμός *Curve25519* που παρέχει επίπεδο ασφάλειας 128 bits και το σχήμα ελλειπτικής καμπύλης Diffie–Hellman
  - `make_common_secret()`
    Χρησιμοποιεί το ιδιωτικό κλειδί του instance της `Person` και το δημόδιο κλειδί της άλλης πλευράς για να παράξει και αποθηκεύσει το κοινό μυστικό K στο instance της `Person`.
    Γίνεται χρήση της `Box` της `PyNaCl`
  - `make_derived_key()`
    Με χρήση της *key derivation function* `blake2b` και είσοδο το κοινό μυστικό K παράγεται αποθηκεύεται στο instance της `Person` το κλειδί K΄ το οποίο γίνεται truncated σε 32 χαρακτήρες λόγω του απαιτούμενου μεγέθους εισόδου στην κλάση κρυπτογράφησης `SecretBox` (βλ. `encrypt_message()`)
  - `make_message()`

Παράγεται το μήνυμα προς ανταλλαγή
- `encrypt_message()`
  Χρησιμοποιεί το κλειδί Κ΄ του instance της `Person` για να κρυπτογραφήσει και να αποθηκεύσει στο instance το μήνυμα που παράχθηκε.
  Γίνεται χρήση της κλάσης `SecretBox` της `PyNaCl` η οποία δέχεται ως είσοδο κλειδί 32 bytes και για την κρυπτογράφηση χρησιμοποιεί τον αλγόριθμο *XSalsa20 stream cipher* (1, 2). Η κλάση εισάγει κι έναν αυθεντικοποιητή των 16 byte με τον αλγόριθμο Poly1305 MAC, ο οποίος ελέγχεται κατά την αποκρυπτογράφηση.
- `make_tag()`
  Δημιουργεί και αποθηκεύει την ετικέτα από το κρυπτογραφημένο μήνυμα για το instance της `Person` για τη διαδικασία ελέγχου ακεραιότητας με χρήση της `blake2b`
- `get_requirements()`
  Δέχεται ως είσοδο και αποθηκεύει τις πληροφορίες που χρειάζονται στο instance της `Person` που έχει το ρόλο του παραλήπτη για την αποκρυπτογράφηση και τον έλεγχο ακεραιότητας το μηνύματος
- `verify_integrity()`
  Πραγματοποιεί τον έλεγχο ακεραιότητας του μηνύματος από τη μεριά του παραλήπτη. Χρησιμοποιεί τη συνάρτηση `sodium_memcmp` της `PyNaCl` και τυπώνει το αντίστοιχο μήνυμα κατά τον έλεγχο τερματίζοντας το πρόγραμμα σε περίπτωση αποτυχίας
- `decrypt_message()`
  Χρησιμοποιεί την κλάση `SecretBox` της `PyNaCl` η οποία αρχικοποιείται με το παραγόμενο κλειδί Κ΄ του instance της `Person` και τη μέθοδό της `decrypt()` που δέχεται ως είσοδο το κρυπτογραφημένο μήνυμα για να κάνει την αποκρυπτογράφηση και τυπώνει το αντίστοιχο μήνυμα κατά τον έλεγχο τερματίζοντας το πρόγραμμα σε περίπτωση αποτυχίας

Υλοποίηση: Βήματα

1. Αρχικοποίηση της `Person` για τον *Bob*
2. Δημιουργία κλειδιών για τον *Bob* με την `make_key_pair()`
3. Αρχικοποίηση της `Person` για την *Alice*
4. Η *Alice* λαμβάνει το δημόσιο κλειδί του *Bob* με την `get_other_public()`
5. Δημιουργία κλειδιών για την *Alice* με την `make_key_pair()`
6. Η *Alice* δημιουργεί το κοινό μυστικό K με την `make_common_secret()`
7. Η *Alice* δημιουργεί το παραγόμενο κλειδί Κ΄ με την `make_derived_key()`
8. Η *Alice* δημιουργεί το μήνυμα με την `make_message()`
9. Η *Alice* κρυπτογραφεί το μήνυμα με την `encrypt_message()`
10. Η *Alice* δημιουργεί την ετικέτα ακεραιότητας με την `make_tag()`
11. Ο *Bob* λαμβάνει τις απαραίτητες πληροφορίες για αποκρυπτογράφηση/έλεγχο ακεραιότητας με την `get_requirements()`
12. Ο *Bob* δημιουργεί το κοινό μυστικό K από το ιδιωτικό κλειδί του και το δημόδιο κλειδί της *Alice* με την `make_common_secret()`
13. Ο *Bob* δημιουργεί το παραγόμενο κλειδί Κ΄ με την `make_derived_key()`
14. Ο *Bob* κάνει έλεγχο ακεραιότητας με την `verify_integrity()` όπου τυπώνεται το αντίστοιχο μύνημα και σε περίπτωση αποτυχίας τερματίζεται το πρόγραμμα
15. Ο *Bob* κάνει αποκρυπτογράφηση με την `decrypt_message()` όπου τυπώνεται το αντίστοιχο μύνημα και σε περίπτωση αποτυχίας τερματίζεται το πρόγραμμα

Μπορούμε να ελέγξουμε την αποτυχία του προγράμματος αλλάζοντας το κρυπτογραφημένο μήνυμα ή την ετικέτα της *Alice* πριν το βήμα 11.