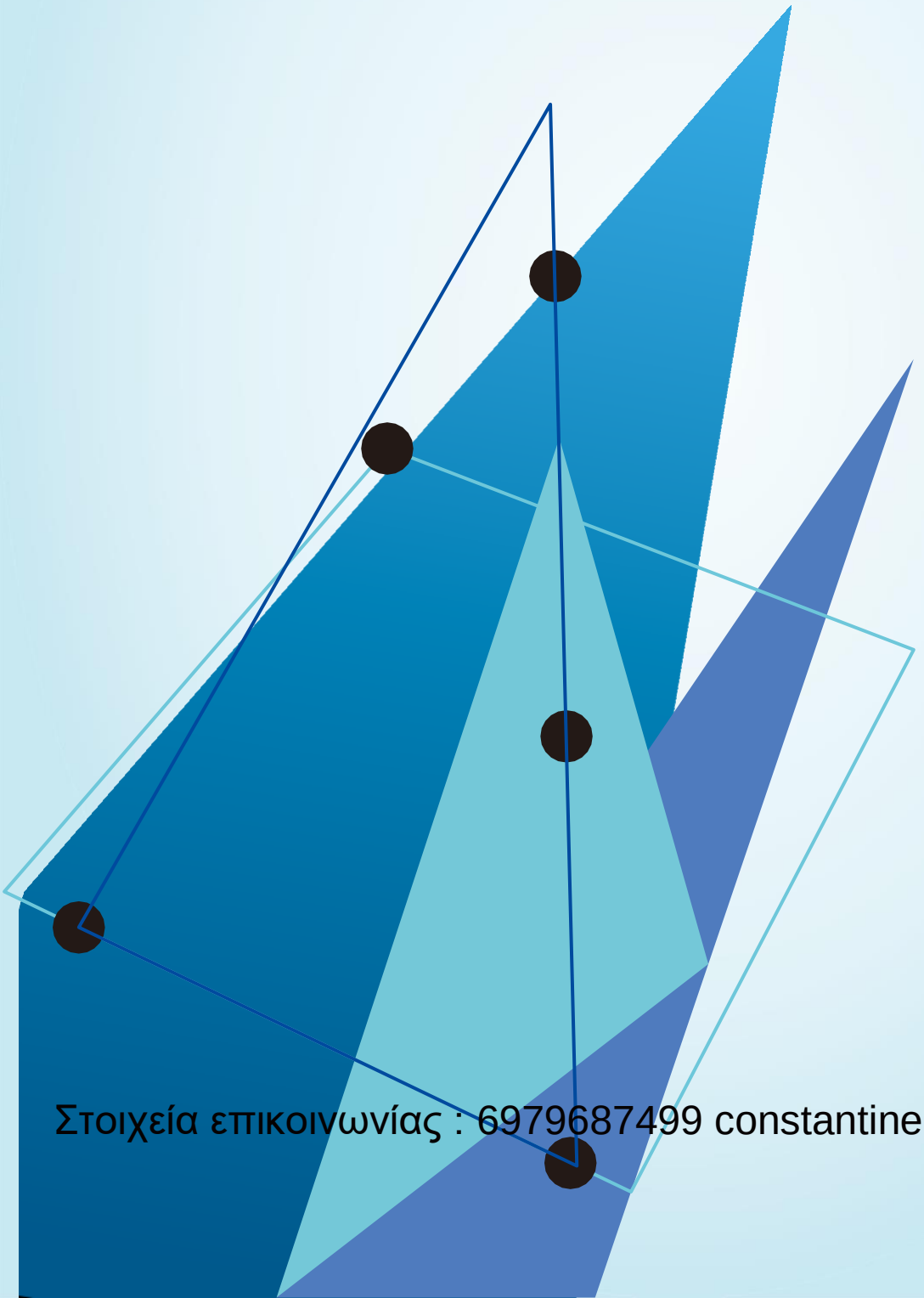
Τμήμα Πληροφορικής Πανεπιστήμιο Πειραιά

Πρόγραμμα Μεταπτυχιακών Σπουδών

Εργασία Λογικός Προγραμματισμός Μέρος Β : Κρυπτογραφία με Prolog

ΧΡΟΝΟΠΟΥΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

AM: MPPL 21081



Στοιχεία επικοινωνίας : 6979687499 constantine1212204@gmail.com

Κρυπτογραφία με Prolog

Η λέξη **κρυπτογραφία** (αγγλ.: *cryptography*) προέρχεται από τα συνθετικά «κρυπτός» + «γράφω» και είναι ένα διεπιστημονικό γνωστικό πεδίο που ασχολείται με τη μελέτη, την ανάπτυξη και τη χρήση τεχνικών *κρυπτογράφησης* και *αποκρυπτογράφησης* με σκοπό την απόκρυψη του περιεχομένου των μηνυμάτων.

Θα θέλαμε να βεβαιώσουμε :

- Ακεραιότητα των δεδομένων(τα δεδομένα δεν έχουν αλλοιωθεί).
- Αυθεντικότητα των δεδομένων (η πηγή των δεδομένων είναι επιβεβαιωμένη).
- Εχεμύθεια των δεδομένων (μόνο όσοι επιλέγουμε εμείς μπορούν να έχουν πρόσβαση στα δεδομένα).

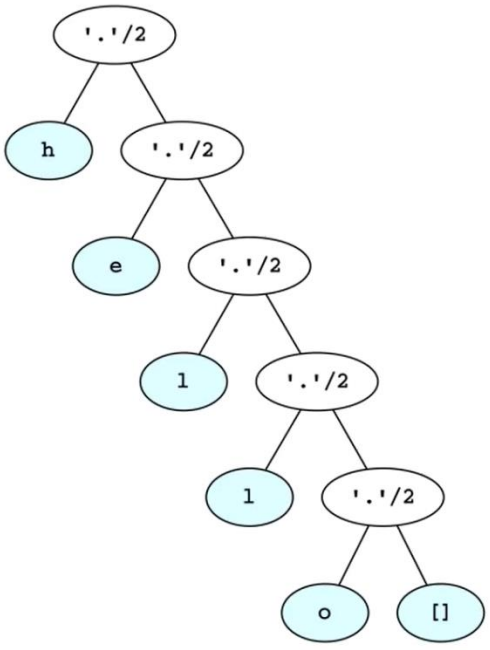
Εξασφαλίζουμε αυτές τοις ιδιότητες χρησιμοποιώντας :

- Κρυπτογραφικά hashes (cryptographic hashes)
- Ψηφιακές υπογραφές (digital signatures)
- Κρυπτογράφηση (encryption)

Το swi-prolog παραθέτει αυτές τοις ικανότητές στην βιβλιοθήκη Library(crypto).

Η Prolog αναπαριστά τα δεδομένα ως μια λίστα από χαρακτήρες Δηλ.

Το : ‘hello’ είναι μια λίστα από τους χαρακτήρες : ‘h’ , ’e’ . ’l’ , ‘l’ , ’o’.



Data encoding

Κάθε χαρακτήρας έχει ένα αντίστοιχο κωδικό χαρακτήρα

ΠΧ :

Ο χαρακτήρας ‘A’ έχει κωδικό χαρακτήρα ‘65’. (Hexadecimal : 41)

Ο χαρακτήρας ‘Ü’ έχει κωδικό χαρακτήρα ‘252’. (Hexadecimal : FC)

Ο χαρακτήρας ‘π’ έχει κωδικό χαρακτήρα ‘960’. (Hexadecimal : 3C0)

Θα μπορούσαμε να χρησιμοποιούμαι hexadecimal(δεκαεξαδικούς) για να αναφερόμαστε σε χαρακτήρες ΠΧ :

```
?- C = '\x41\'.
C = 'A'.


?- Cs = "\x41\\xfc\\x3c0\".
Cs = "Aüπ".
```

Η char_code/2 μας επιτρέπει να συνδέσουμε έναν χαρακτήρα με τον κωδικό χαρακτήρα του :

```
?- maplist(char_code, "Aüπ", Codes).
   Codes = [65,252,960].
```

Cryptographic hash functions

Μια cryptographic hash function(*συνάρτησης κατακερματισμού)* αντιστιχειζει τα δεδομενα σε ένα κωδικό (hash).

- Θα πρέπει να είναι γρήγορο στην εκτέλεση και αδύνατον να αντιστραφεί .
- Θα πρέπει να είναι δύσκολο να βρούμε δυο διαφορετικά μηνύματα που βγάζουν το ίδιο αποτέλεσμα .
- Μικρές αλλαγές στο μίνιά θα έχουν μεγάλες αλλαγές στο αποτέλεσμα .

Ένα cryptographic hash μπορεί να χρησιμοποιηθεί για να βεβαιώσει την ακεραιότητα των δεδομένων. Εάν τα δεδομένα έχουν αλλάξει το πηγαίο hash θα είναι διαφορετικό.

```
?- crypto_data_hash("Hello world!", Hash, []).
Hash = c0535e4be2b79ffd93291305436bf889314e4a3faec05ecffcbb7df31ad9e51a.

?- crypto_data_hash("Hello world!", Hash, [algorithm(blake2s256)]).
Hash = c63813a8f804abece06213a46acd04a2d738c8e7a58fbf94bfe066a9c7f89197.

?- crypto_data_hash("Hello world!", Hash, [algorithm(A)]).
Hash = c0535e4be2b79ffd93291305436bf889314e4a3faec05ecffcbb7df31ad9e51a,
A = sha256.
```

Εφαρμογή : Δημιουργία τυχαίων αριθμών

Επαναλαμβανομένη χρυσή hash function μας δίνει ψεφτο-τειχεα (pseudo-random) αριθμούς

```
random_chars(Sead,Cs)  :-
        crypto_data_hash(Seed, Hash, []).
        state_chars(Hash, Cs).

state_chars(Cs0, Cs) :- crypto_data_hash([r|Cs0], Cs, []).

state_chars(Cs0, Cs) :-
        crypto_data_hash (Cs0, Cs1, []),
        state_chars(Cs1, Cs).
```

```
?- random_chars("behold the purple raven scry", Cs).
   Cs = "88e22279122fdd5155f ..."
;  Cs = "676e9d7c443800c43f0 ..."
;  Cs = "b6a85ede23f6ff789fc ..."
;  Cs = "77aab4bfdabac317f7c ..."
;  Cs = "c882614364cb1a259fc ..."
```

Αλλά για έναν Κρυπτογραφικά «ασφαλή» τυχαίο αριθμό χρειαζόμαστε :

- Resilience (Ανθεκτικότητα)
  Όταν δεν γνωρίζουμε την ,κατάσταση το αποτέλεσμα φαίνεται τυχαίο.
- Forward security
  Αν η τωρινή κατάσταση αποκαλυφθεί ,τα προηγούμενα output πάλει φαίνονται τυχαία
- Break-in recovery
  Μελλοντικά output φαίνονται τυχαία .

Η crypto_n_random_bytes/2 παρέχει αυτές τοις διασφαλίσεις .

```
?- crypto_n_random_bytes(32, Bs).
Bs = [202, 89, 136, 199, 235, 200, 223, 170, 235|...].
```

Αν θέλουμε μπορούμε να μετατρέψουμε μια σειρά από bytes σε integer :

Ασφαλείς αποθήκευση κωδικού

Θα αποθηκεύσουμε μόνο ένα hash του κωδικού.

Το hash θα πρέπει να είναι διαφορετικό για δύο κωδικούς ακόμα και αν είναι ίδιοι

- Συνδικάζουμε τον κωδικό με ένα τυχαίο κομμάτι που λέγεται salt.

Το να μαντέψουμε τον κωδικό θα πρέπει να είναι δύσκολο.

- Ο υπολογισμός του hash θα πρέπει να είναι αργός, το salt θα πρέπει να είναι αρκετά μακρύ.

Η “crypto_password_hash/2” παρέχει όλα αυτά τα ζητούμενα και έχουμε :

```
?- crypto_password_hash("password1", Hash).
Hash = '$pbkdf2-sha512$t=131072$RU6YxN4px3Pq8dsqjedRYg$V0rsQneesApvMsqFCX1L
DljEftC+AhaN0mvlHD7/oYgv3sjnBRgnXchXq4j2ostMTNlEcFYuPsirK5er+0DXqw'.
```

Το υπολογισμένο hash μπορεί να αποθηκευτεί με ασφάλεια.

Χρησιμοποιούμαι το ίδιο predicate με το αντίστοιχο hash για να για να τσεκάρουμε αν ένα δοσμένος κωδικός είναι σωστός .

Ψηφιακή Υπογραφή (Digital Signature)

Η ψηφιακή υπογραφή μας επιτρέπει να βεβαιώσουμε την αυθεντικότητα των δεδομένων , τα δεδομένα είναι αυτά που ήταν και δεν έχουν αλλοιωθεί .

Χρησιμοποιούμαι το Ed25519 για να υπογράψουμε τα δεδομένα και να επιβεβαιώσουμε την υπογραφή. Το Ed25519 είναι μια περίσταση του Edwards-curve Difital Signature Algorithm (EdDSA).

Χρησιμοποιούμαι το ed25519_sign/4 για να υπογράψουμε τα δεδομένα και το ed25519_verify/4 για να επιβεβαιώσουμε την αυθεντικότητα της υπογραφής

```
?- ed25519_new_keypair(Pair).
   Pair = "0S\x2\\x1\\x1\0\x5\\x6\\x3\+ep\x4\\"\x4\ 3|\x9c\ ...".
   Pair = "0S\x2\\x1\\x1\0\x5\\x6\\x3\+ep\x4\\"\x4\ 2\x1\: ...".
   Pair = "0S\x2\\x1\\x1\0\x5\\x6\\x3\+ep\x4\\"\x4\ «¢¬ ...".
   Pair = "0S\x2\\x1\\x1\0\x5\\x6\\x3\+ep\x4\\"\x4\ ñ´¿ ...".
```

Το key pair περιέχει τον κωδικό και πρέπει να παραμένει κρυφό.

```
?- ed25519_new_keypair(Pair),
   Data = "Hello!",
   ed25519_sign(Pair, Data, Sig, []).
   Pair = "0S\x2\\x1\\x1\0\x5\\x6\\x3\+ep\x4\\"\x4\ F%\x80\ ...",
   Data = "Hello!",
   Sig = "40b4dd32e67a4a28108b438c7f9429608e1ca9f ...".
```

Το Sig χρησιμοποιείται για να αυθεντικοποίηση την υπογραφή και μπορεί να το μοιραστούμε ελεύθερα .

Συμμετρική Κρυπτογράφηση (Symmetric encryption)

Ερχόμαστε τώρα στο θέμα που μας ενδιαφέρει περισσότερο την συμμετρική κρυπτογράφηση. Αυτό σημαίνει ότι το ίδιο κλειδί χρησιμοποιείται για την κρυπτογράφηση και την αποκρυπτογράφηση των δεδομένων. Για παράδειγμα, μια περίπτωση χρήσης είναι η κρυπτογράφηση αρχείων με κωδικό πρόσβασης, ώστε να μπορεί να χρησιμοποιηθεί ο ίδιος κωδικός πρόσβασης για την αποκρυπτογράφηση τους.

Η Prolog μπορεί να εκτελέσει κρυπτογράφηση αυθαίρετων δεδομένων με ασφαλή τρόπο, χρησιμοποιώντας το κατηγόρημα crypto_data_encrypt/6.

Εκτός από τα δεδομένα που θέλουμε να κρυπτογραφήσουμε, πρέπει να παρέχουμε :

- τον αλγόριθμο που θέλουμε να χρησιμοποιήσουμε
- το κλειδί που χρησιμοποιείται για την κρυπτογράφηση
- το διάνυσμα αρχικοποίησης («initialization vector »).

Μπορούμε να καθορίσουμε κλειδιά και διάνυσμα ως λίστες byte. Μπορούμε να χρησιμοποιήσουμε το crypto_n_random_bytes/2 για να δημιουργήσουμε κρυπτογραφικά ισχυρά ψευδοτυχαία bytes. Με εξαιρετικά υψηλή πιθανότητα, κάθε επίκληση θα δημιουργήσει ένα εντελώς διαφορετικό κλειδί.

Το κλειδί πρέπει να διατηρείται απολύτως μυστικό για να διατηρείται η εμπιστευτικότητα του κρυπτογραφημένου κειμένου. Αντίθετα, το διάνυσμα μπορεί να αποθηκευτεί με ασφάλεια και να μεταδοθεί σε απλό κείμενο.

Ο αλγόριθμος ChaCha20 επεκτείνει ένα κλειδί 256-bit σε ένα 2^96 randomly accessible streams . Το κάθε ένα περιέχει 2^32 randomly accessible 64-byte blocks.

Για να κρυπτογραφήσουμε ένα n-byte μήνυμα Mn διαλέγουμε ένα από αυτά τα streams S, παίρνει τα πρώτα n bytes Sn , και υπολογίζει Sn ⊕ Mn .Για να αποκρυπτογραφήσουμε n bytes , ξανά χρησιμοποιούμαι xor με Sn .

Ένα block δημιουργείται από 3 inputs :

- Key
- Nonce
- Counter

Ο ChaCha20 λειτουργεί με βάσει λέξεις 1 λέξει = 4 bytes . Τα τρία inputs συνδικάζονται με τέσσερες σταθερές για να γεμίσει έναν 4*4 πίνακα .

Μετά ο πίνακας ανακατεύεται χρησιμοποιώντας 20 γύρους

Συγκεκριμένα ένα παράδειγμα με αλγόριθμο ChaCha20-Poly 1305 με τυχαίο key και nonce :

```
?- crypto_n_random_bytes(32, Ks),
   crypto_n_random_bytes(12, IV),
   crypto_data_encrypt("Top Secret Text", 'chacha20-poly1305', Ks,
IV, CipherText, [tag(Ts)]).
```

```
?- crypto_n_random_bytes(32, Ks),
   crypto_n_random_bytes(12, IV),
|        crypto_data_encrypt("Top Secret Text", 'chacha20-poly1305', Ks, IV, CipherText, [tag(Ts)]).
Ks = [13, 157, 251, 178, 23, 244, 213, 43, 107|...],
IV = [113, 135, 155, 104, 122, 193, 228, 120, 182|...],
CipherText = "\u001C\u0083dMp\u000FSw\u0016¶Ñ\aã®Y",
Ts = [30, 20, 124, 170, 72, 226, 27, 26, 190|...].
```

Και για αποκρυπτογράφηση :

```
?- crypto_data_decrypt($CipherText, 'chacha20-poly1305', $Ks, $IV,
PlainText, [tag($Ts)]).
```

```
?- crypto_data_decrypt($CipherText, 'chacha20-poly1305', $Ks, $IV, PlainText, [tag($Ts)]).
PlainText = "Top Secret Text",
CipherText = "\u001C\u0083dMp\u000FSw\u0016¶Ñ\aã®Y",
Ks = [13, 157, 251, 178, 23, 244, 213, 43, 107|...],
IV = [113, 135, 155, 104, 122, 193, 228, 120, 182|...],
Ts = [30, 20, 124, 170, 72, 226, 27, 26, 190|...].
```