

# Πρόβλημα Κρυπτογραφίας

- Ανταλλαγή μηνυμάτων μεταξύ δύο μερών ώστε το περιεχόμενό τους, σε περίπτωση υποκλοπής από τρίτο, να παραμένει ακατάληπτο

*Για παράδειγμα, οι ηλεκτρονικές αγορές στηρίζονται στην αποστολή αριθμού πιστωτικής κάρτας από τον Η/Υ μας στον Η/Υ της εταιρίας μέσω πολλαπλών ενδιάμεσων Η/Υ*

# Ορολογία

- Τα επικοινωνούντα μέρη επικράτησε να αποκαλούνται *Αλίκη* και *Μποπ* (*Alice, Bob*), ενώ ο υποκλοπέας *Εύα* (*Eve*)
- **Κρυπτογράφηση/Αποκρυπτογράφηση:**
  - Το κείμενο  $M$  (plaintext) κρυπτογραφείται (μετατρέπεται) στο κρυπτογράφημα (ciphertext)  $C$  από την Αλίκη και αποστέλλεται στον Μποπ
  - Αυτός πρέπει να είναι σε θέση να το αποκρυπτογραφήσει, δηλαδή να μετατρέψει το  $C$  σε  $M$
  - Ακόμη και αν η Εύα υποκλέψει το  $C$  δεν πρέπει να είναι σε θέση να το αποκαταστήσει στην αρχική του μορφή  $M$
- **Ψηφιακές Υπογραφές:**
  - Όταν γίνεται η αποστολή μηνυμάτων, η Αλίκη τα «υπογράφει» ώστε ο Μποπ να είναι σίγουρος ότι αποστολέας τους είναι η Αλίκη και όχι η Εύα υποκρινόμενη την Αλίκη

# Προσεγγίσεις

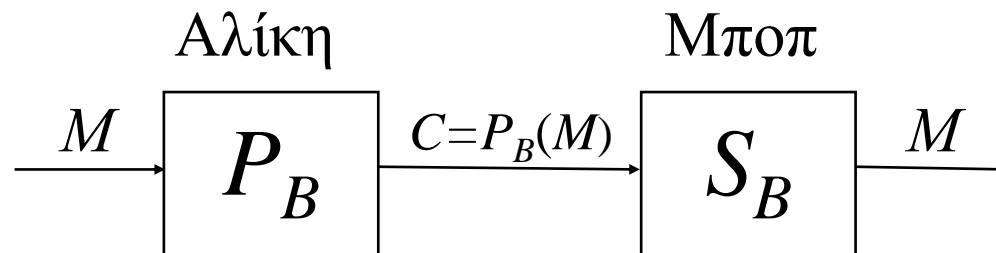
- **Μυστικό Κλειδί**

- Τα δύο μέρη ανταλλάσσουν ένα μυστικό κλειδί, το οποίο χρησιμοποιείται τόσο για την κρυπτογράφηση όσο και την αποκρυπτογράφηση
- Πρέπει να το ανταλλάξουν με ασφάλεια πριν αρχίσει η επικοινωνία
- **Παράδειγμα: Κρυπτογράφηση Ιουλίου Καίσαρα**
  - το  $i$ -στό γράμμα αντικαθίσταται με το  $(i+3)$ -στό mod 26
  - Το 3 αποτελεί το μυστικό κλειδί

# Προσεγγίσεις (συν.)

- **Δημόσιο Κλειδί**

- Η επικοινωνία γίνεται μέσω δύο κλειδιών, ενός δημόσιου  $P$  και ενός μυστικού  $S$
- Πχ, για την επικοινωνία του ο Μποπ διαλέγει δύο κλειδιά  $P_B, S_B$ :
  - Το πρώτο, το ανακοινώνει, και το δεύτερο, το κρατά για τον εαυτό του
  - Εάν η Αλίκη θέλει να επικοινωνήσει μαζί του, χρησιμοποιεί το  $P_B$  για την παραγωγή τού  $C$
  - Ο Μποπ αποκρυπτογραφεί το  $C$  μέσω του  $S_B$



# Προσεγγίσεις (συν.)

– Προδιαγραφές  $P, S$  :

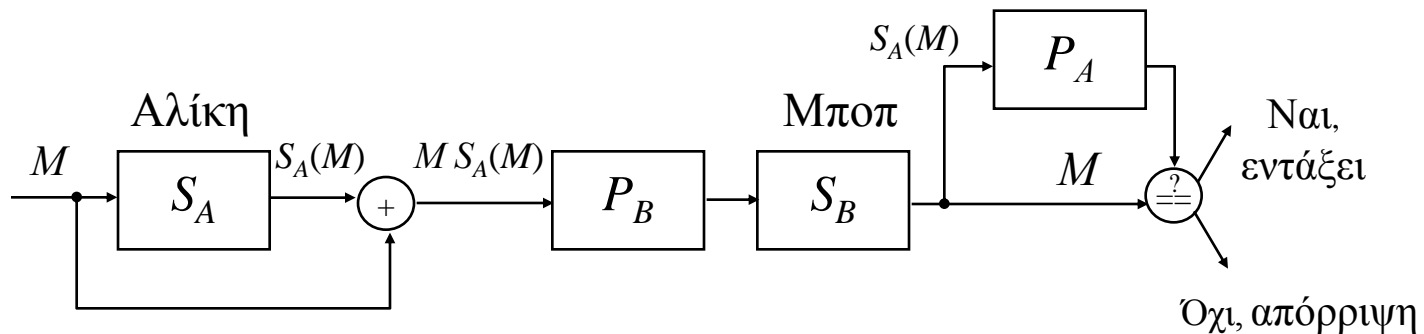
1.  $P(S(M)) = S(P(M))$
2. Να υπολογίζονται εύκολα
3. Η εξαγωγή του  $S$  από το  $P$  να είναι πρακτικώς αδύνατη

- Η ιδιότητα 1 εξασφαλίζει τον σχηματισμό ψηφιακής υπογραφής  $S(M)$ , καθώς:

$$M = P(S(M)),$$

συνεπάγεται πως ο Μποπ είναι ο συντάκτης του κειμένου  $M$

- Η πλήρης επικοινωνία, με υπογραφές, έχει ως εξής:



# Βασικές Μαθηματικές Έννοιες

- Πρώτος Ακέραιος:
  - Κάθε ακέραιος με διαιρέτες τον εαυτό του και την μονάδα
  - Υπάρχουν άπειροι πρώτοι αριθμοί
  - Κάθε ακέραιος γράφεται με μοναδικό τρόπο ως γινόμενο πρώτων
  - Η πυκνότητα των πρώτων στο διάστημα  $[1..n]$  είναι ανάλογη του  $n / \log n$  (**Θεώρημα πρώτων αριθμών**)

# Βασικές Μαθηματικές Έννοιες (συν.)

- Μέγιστος κοινός διαιρέτης (gcd)  
 $\gcd(x,y) = \min \{xa+yb \mid x,y \in \mathbb{Z}\} = \gcd(y, x \bmod y)$
- Σχετικώς πρώτοι αριθμοί  $x,y$   
 $\gcd(x,y) = 1$
- $\mathbb{Z}_p = \{0, 1, 2, \dots, p-1\}$
- $\mathbb{Z}_p^* = \{a \in \mathbb{Z}_p \mid \gcd(a,p) = 1\}$  (με την πράξη επί modulo, συνιστά αβελιανή πολλαπλασιαστική ομάδα-κλειστότητα, προσεταιριστικότητα, αντιμεταθετικότητα, ουδέτερο/αντίστροφο στοιχείο-ενώ  $a^{\phi(p)} \equiv 1 \pmod p, a \in \mathbb{Z}_p^*$ )
- Για  $p$  πρώτο ισχύει  $\mathbb{Z}_p^* = \{1, 2, \dots, p-1\}$
- Μικρό Θεώρημα Φερμά  
Εάν  $p$  πρώτος, τότε για κάθε ακέραιο  $a \in \mathbb{Z}_p^*$ , ισχύει:  
 $a^p \equiv a \pmod p$  ή  $a^{p-1} \equiv 1 \pmod p$   
**Το αντίστροφο δεν ισχύει, πχ,  $234 = 11 \times 31$**

# Βασικές Μαθηματικές Έννοιες (συν.)

- Αντίστροφος  $a^{-1} \pmod n$ 
  - $aa^{-1} \equiv 1 \pmod n$
  - Υπάρχει, εάν  $\gcd(a,n) = 1$
- Θεώρημα κινέζικου υπολοίπου:

Το σύστημα

$$x \equiv a_1 \pmod{n_1}$$

$$x \equiv a_2 \pmod{n_2}$$

⋮

$$x \equiv a_k \pmod{n_k}$$

όπου οι  $n_1, n_2, \dots, n_k$  ανά δύο σχετικώς πρώτοι, έχει μοναδική λύση  $\pmod{n_1 n_2 \cdots n_k}$ . Δηλαδή,  $x \equiv y \equiv a_i \pmod{n_i}$  αν και μόνον αν  $x \equiv y \pmod{n_1 n_2 \cdots n_k}$



# Βασικές Μαθηματικές Έννοιες (συν.)

- Συνέπεια θεωρήματος κινέζικου υπολοίπου
  - Έστω οι  $n_1, n_2, \dots, n_k$  ανά δύο σχετικώς πρώτοι. Για κάθε ζεύγος ακεραίων  $x, a$  ισχύει

$$x \equiv a \pmod{n_1}$$

$$x \equiv a \pmod{n_2}$$

⋮

$$x \equiv a \pmod{n_k}$$

αν και μόνον αν

$$x \equiv a \pmod{n_1 n_2 \dots n_k}$$

# Κρυπτοσύστημα Δημοσίου Κλειδιού RSA (Riverest, Shamir, Adleman)

- Ο αλγόριθμος εξαγωγής δημόσιου και κρυφού κλειδιού είναι πολύ απλός:
  1. Επιλογή δύο *τυχαίων, μεγάλων, πρώτων ακεραίων*  $p, q$
  2. Υπολογισμός του γινομένου  $n = pq$
  3. Επιλογή ενός μικρού, περιττού ακεραίου  $e$ :  $\gcd(e, \varphi(n))=1$  ( $e, \varphi(n)$  σχετικώς πρώτοι), με  $\varphi(n)=(p-1)(q-1)$
  4. Υπολογισμός του  $d=e^{-1} \bmod \varphi(n)$  ( $de=1 \bmod \varphi(n)$ )
  5. Ανακήρυξη του  $P = (e, n)$  ως *δημόσιο κλειδί RSA*
  6. Ανακήρυξη του  $S = (d, n)$  ως *μυστικό κλειδί RSA*

Οπότε, για  $M < n$ , έχουμε

$$P(M)=M^e \bmod n \text{ και } S(C)=C^d \bmod n$$

# Απόδειξη Ιδιότητας Αντιστροφής

- Πρέπει να δείξουμε ότι:

$$P(S(M)) = S(P(M)) = M$$

- Έχουμε ότι:

$$P(S(M)) = S(P(M)) = M^{ed} \pmod n$$

$$ed = ee^{-1} \pmod{\varphi(n)} = 1 + k\varphi(n) = 1 + k(p-1)(q-1)$$

$$M^{1+k(p-1)(q-1)} \pmod p = M(M^{(p-1)})^{k(q-1)} \pmod p$$

- Όμως  $p$  πρώτος, άρα

$$M^{(p-1)} \pmod p = 1 \text{ (μικρό θεώρημα Φερμά)}$$

και συνεπώς

$$M(M^{(p-1)})^{k(q-1)} \pmod p = M \cdot 1 \pmod p = M \pmod p (*)$$

# Απόδειξη Ιδιότητας Αντιστροφής (συν.)

- Παρομοίως,  $q$  πρώτος, άρα:

$$M^{(q-1)} \bmod q = 1 \text{ (μικρό θεώρημα Φερμά)}$$

και συνεπώς:

$$M(M^{(q-1)})^{k(p-1)} \bmod q = M \cdot 1 \bmod q = M \bmod q \quad (\dagger)$$

(\*) και  $(\dagger) \Rightarrow M(M^{(q-1)})^{k(p-1)} \bmod pq = M \bmod pq$   
(συνέπεια θεωρήματος κινέζικου υπολοίπου)

# Παράδειγμα

- Έστω ότι θέλουμε να κρυπτογραφήσουμε το μήνυμα:  
Σύστημα RSA
- Σε ASCII η μορφή του είναι  
211**253**243**244**231**236**255**032**082**083**041
- Εάν επιλέξουμε  
 $p=19, q=31, e=53,$   
τότε  
 $n=19 \cdot 31=586, \varphi(n)=18 \cdot 30=540$  και  $d = 377.$
- Το κείμενο πρέπει να χωριστεί σε τμήματα των *δύο* ψηφίων, τα οποία κρυπτογραφούνται και μετά συνενώνονται:  
211253243244231236255032082083041  
↓  
**048084375517497205396084180377350497202267296157001**
- Η αποκωδικοποίηση γίνεται τμήμα-τμήμα. Π.χ.,  
 $048^{377} \bmod 586 = 21$

# Πράξεις που απαιτούνται

- Υπολογισμός γινομένου δύο ακεραίων-εύκολη
- Ύψωση σε δύναμη modulo  $n$  -εύκολη αναδρομική διαδικασία, αφού:
  - $a^b = a^{b/2} a^{b/2} = (a^2)^{b/2}$ ,  $b$  άρτιος, ή
  - $a^b = a a^{(b-1)/2} a^{(b-1)/2} = a(a^2)^{(b-1)/2}$ ,  $b$  περιττός

## Υπολογισμός αντίστροφου $e$ modulo $n$ -εύκολος

- Ισχύει  $\gcd(a,b)=xa+yb$ .
- Εμείς θέλουμε να βρούμε τον αριθμό  $x$ , τέτοιον ώστε  $ex \equiv 1 \pmod n$  ή  $ex = 1+cn, c \in \mathbb{N}$ .
- Όμως  $e$  αντιστρέψιμος σημαίνει  $\gcd(e,n) = 1$  ή  $xe+yn = 1$ .
- Άρα, αρκεί να βρούμε τον συντελεστή  $x$  του μέγιστου κοινού διαιρέτη. Αυτό γίνεται μέσω του αλγορίθμου του Ευκλείδη

**Algorithm** gcdExtended (int a, int b, coeff c) // απόδειξη με επαγωγή

1. **if** (b == 0){
2.   c.x = 1;
3.   c.y = 0;
4. }
5. **else**{
6.   gcdExtended(b, a % b, c); // Επαγ. Υπόθεση:  $\gcd(b,a\%b)=x1*b+y1*(a\%b)$
7.   x1 = c.x; y1 = c.y; //  $\gcd(a,b) = y1*a+(x1-(a/b)*y1)*b = y1*(a-(a/b)b)+x1*b$
8.   c.x = y1;                   //                   =  $y1*(a\%b)+ x1*b = \gcd(b,a\%b)$  ΟΕΔ
9.   c.y = x1-(a/b)\*y1;
10. }

## Εύρεση δύο πρώτων ακεραίων $p, q$ (λίγο δύσκολο)

- Υπάρχουν πιθανοτικά τεστ που αποφαινόνται εάν ένας ακέραιος είναι πρώτος ή όχι.
- Το πιο γνωστό είναι το τεστ Miller-Rabin, το οποίο βασίζεται στο μικρό θεώρημα του Φερμά:

$$p \text{ πρώτος και } a \in \mathbb{Z}_p^* \Rightarrow a^{p-1} = 1 \pmod{p}$$
$$a^{p-1} \neq 1 \pmod{p} \Rightarrow p \text{ σύνθετος}$$

**Algorithm** millerRabinTest(int n, int s)

1. **for** ( $i = 0; i < s; i++$ ) {
2.  $a = \text{random}(1, n-1)$ ;
3. **if** ( $(\text{power}(a, n-1, n) \neq 1)$  //  $a^{n-1} \neq 1 \pmod{n}$
4. **return** composite; // σίγουρα σύνθετος
5. }
6. **return** prime; // με επιφύλαξη

Για  $s = 51$  η πιθανότητα λάθους είναι μόλις  $4.44 \cdot 10^{-16}$



# Μειονεκτήματα RSA

- Αργοί οι υπολογισμοί εν σχέσει με τους αλγορίθμους μυστικών κλειδιών
- Δεν υπάρχει απόδειξη ότι δεν «σπάει»: Στηρίζεται στην *εικασία* ότι *μόνον* εάν βρεθεί αποτελεσματική διαδικασία παραγοντοποίησης ακεραίων, ο αλγόριθμος σπάει

# Ορισμοί

- ***On-line* Αλγόριθμος**

Κάθε αλγόριθμος, ο οποίος, δεχόμενος μία ακολουθία αιτήσεων, την επεξεργάζεται *σειριακά*, ώστε να περατώνει κάθε αίτηση *προτού* αναλάβει την εξυπηρέτηση της επόμενης

- Δεν ενδείκνυται κλασική ανάλυσή τους γιατί, ενδεχομένως, ένας κακόβουλος χρήστης μπορεί να εμφανίζει ακολουθίες που οδηγούν τον αλγόριθμο σε χειρίστη συμπεριφορά

# Ορισμοί (συν.)

- Αντ' αυτού, συγκρίνεται βάσει της επιδόσεως ενός *off-line* ιδεατού αλγορίθμου  $O$ , ο οποίος γνωρίζει όλη την ακολουθία εξ αρχής
- Η επίδοση του *off-line* αλγορίθμου  $O$  εκφράζει την δυσκολία (πολυπλοκότητα) της εισόδου
- Κατά συνέπεια, μιλούμε για  $c$ -ανταγωνιστικό αλγόριθμο ελαχιστοποιήσεως  $A$ , όταν, για οποιαδήποτε ακολουθία εισόδου  $\sigma$ , ισχύει:

$$f_A(\sigma) - cf_O(\sigma) \leq b, \quad b \text{ σταθερά, ίσως } f(c)$$

# Ορισμοί (συν.)

- Ομοίως, μιλούμε για  $c$ -ανταγωνιστικό αλγόριθμο μεγιστοποιήσεως  $A$ , όταν, για οποιαδήποτε ακολουθία εισόδου  $\sigma$ , ισχύει:  
 $f_A(\sigma) - c^{-1}f_O(\sigma) \geq b$ ,  $b$  σταθερά, ίσως  $f(c)$

# Είδη Αντιπάλων

- Συνήθως, ο *off-line* θεωρείται ως *αντίπαλος*, ο οποίος:
  - είτε ξέρει τα πάντα για τον ντετερμινιστικό αλγόριθμο  $A$  και, έτσι, διαλέγει την ακολουθία εισόδου, ώστε να τον παρασύρει σε χειρίστη συμπεριφορά
  - είτε, καθώς ο  $A$  είναι τυχαίος, αλλά πάντοτε αποκαλύπτει τις επιλογές του, παρατηρεί την συμπεριφορά του  $A$  και, βάσει των  $i-1$  επιλογών, διαλέγει την  $i$ -στή είσοδο (*προσαρμοζόμενος-adaptive*)
  - είτε, καθώς ο  $A$  είναι τυχαίος και δεν αποκαλύπτει τις επιλογές του, δεν μπορεί να κάνει τίποτε άλλο παρά να παρακολουθεί τον αλγόριθμο σε μία ακολουθία εισόδου (*επιλήσμων-oblivious*)

# Αναζήτηση Τιμών

- Δοθείσης μίας ακολουθίας τιμών

$$\mathbf{p} = \langle p_1, p_2, \dots, p_n \rangle, \quad p_i \in [m, M], \quad m, M > 0$$

αποκαλυπτόμενης τιμή-προς-τιμή, ο ‘παίκτης’ αναζητεί την μέγιστη (ή την ελάχιστη τιμή), με τους εξής κανόνες:

- Αποδέχεται την τρέχουσα τιμή ως την μέγιστη (ελάχιστη)
  - Απορρίπτει την τιμή, και ζητά να δει την επόμενη
  - Ο χρονικός ορίζοντας του παιχνιδιού είναι πεπερασμένος:
    - Εάν δεν επιλέξει καμμία τιμή, μένει με την τελευταία  $p_n$
- Εφαρμογές
    - Αναζήτηση του καλύτερου/ης κατά την πρόσληψη προσωπικού
    - Αναζήτηση χαμηλότερης τιμής αγοράς
    - Η-πλειστηριασμοί (e-auctions): αναζήτηση υψηλότερης τιμής προσφοράς

# Ντετερμινιστικός Αλγόριθμος- Πολιτική Τιμής Εξασφαλίσεως (RRP)

- Αποδοχή της πρώτης προσφοράς μεγαλύτερης ή ίσης με:

$$\sqrt{mM}$$

ονομαζόμενης ως **τιμή εξασφαλίσεως**

- Ο αλγόριθμος είναι  $\sqrt{\varphi}$ - ανταγωνιστικός ( $\varphi = M/m$ )

– Έστω ότι στον παίκτη προσφέρεται η τιμή εξασφαλίσεως:

- Μελλοντικά, θα του προσφερθεί  $M$  και, άρα, ο λόγος της επιδόσεως είναι:

$$M / \sqrt{mM} = \sqrt{\varphi}$$

– Του προσφέρεται τιμή *το πολύ*  $\sqrt{mM}-1$ , οπότε θα μείνει με την τελευταία τιμή, η οποία, στην χειρότερη περίπτωση, είναι  $m$ :

$$(\sqrt{mM}-1) / m < \sqrt{mM} / m = \sqrt{\varphi}$$

# Πολιτική Τιμής Εξασφαλίσεως (RRP) (συν.)

- Ο λόγος είναι ο καλύτερος δυνατός
  - Θεωρήστε την ακολουθία  $\langle p_1, p_2 \rangle$ , με  $p_1 = x < M$
  - Εάν ο εκάστοτε αλγόριθμος επιλέξει  $x$ , τότε ο αντίπαλος προσφέρει  $p_2 = M$ , επιτυγχάνοντας λόγο  $M/x$
  - Διαφορετικά, ο αντίπαλος προσφέρει  $p_2 = m$ , επιτυγχάνοντας λόγο  $x/m$
  - Ισοσκελίζοντας τους δύο λόγους
$$M/x = x/m,$$
προκύπτει το αιτούμενο



# Πιθανοτικός Αλγόριθμος

- Το όριο  $\sqrt{\phi}$  δύναται να ξεπεραστεί με χρήση τυχειότητας
- Έστω  $RRP(i)$  ο ντετερμινιστικός αλγόριθμος με τιμή εξασφαλίσεως  $m2^i$ . Τότε:

Επιλέγουμε, ισοπίθανα (με πιθανότητα  $1/k$ ), μία εκ των πολιτικών

$$RRP(1), RRP(2), RRP(3), \dots, RRP(k)$$

- Θα δείξουμε ότι ο αλγόριθμος είναι  $(l(\phi)\log\phi)$ -ανταγωνιστικός, με

$$\lim_{\phi \rightarrow \infty} l(\phi) = 1$$

όπου, δίχως βλάβη της γενικότητας,  $\phi = 2^k$ ,  $k \in \mathbb{N}$

# Πιθανοτικός Αλγόριθμος (συν.)

- Έστω ότι η μέγιστη τιμή που ο αντίπαλος επιλέγει είναι:

$$m2^{i+1}-\varepsilon \in [m2^i, m2^{i+1}), \quad 0 < \varepsilon < 1$$

- Τότε

$$\mathbf{E}[f] = \sum_{j=1}^i (1/k)m2^j + \sum_{j>i} (1/k)m = m/k(2^{i+1} - 2 + k - i)$$

καθώς:

- για  $j \leq i$ , η στρατηγική θα διαλέξει τουλάχιστον την τιμή εξασφαλίσεως  $m2^j$
- για  $j > i$ , θα πετύχει τουλάχιστον  $m$  (την ελάχιστη)

# Πιθανοτικός Αλγόριθμος (συν.)

- Οπότε, ο λόγος είναι:

$$(m2^{i+1}-\varepsilon)/m/k(2^{i+1}-2+k-i)$$

$$< m2^{i+1}/m/k(2^{i+1}-2+k-i)$$

$$= k 2^{i+1}/(2^{i+1}-2+k-i)$$

$$= (2^{i+1}/(2^{i+1}-2+k-i))\log\varphi < l(\varphi)\log\varphi$$

$$\text{με } l(\varphi) \sim 1 \text{ για } i = (k-2)+1/\ln 2$$

# Κρυφή Μνήμη (Caching)

- Γρήγορη μνήμη  $k$  θέσεων (σελίδων) που παρεμβάλλεται μεταξύ κύριας μνήμης και εφαρμογής και αποθηκεύει τις τελευταίες προσπελάσεις
  - Εάν υπάρχει, hit! και αποφεύγουμε την προσπέλαση της μνήμης
  - Εάν δεν υπάρχει, miss! και την φέρνουμε από την κύρια μνήμη
    - Εάν υπάρχει χώρος στην cache, τότε την αποθηκεύουμε χωρίς πρόβλημα
    - Αλλιώς, πρέπει να δημιουργήσουμε ελεύθερο χώρο με την έξωση (*eviction*) κάποιας

# Κρυφή Μνήμη (Caching) (συν.)

- Το πρόβλημα εμφανίζεται στα λειτουργικά, αλλά είναι παρόμοιο με αυτό των browsers.
- Κυρίαρχο σημείο: η πολιτική εξώσεως
  - LRU: *η παλαιότερα ανακτημένη*
  - LFU: *με την μικρότερη συχνότητα ανακτήσεως*
  - FiFo: *κατά σειρά τοποθέτησεως στην μνήμη*
  - Random Selection: *τυχαία επιλογή*

# Δυσκολία caching

- Για οποιονδήποτε αλγόριθμο caching υπάρχουν αυθαίρετα μεγάλες ακολουθίες αιτήσεων σελίδων, τέτοιες ώστε ο λόγος ανταγωνιστικότητας να είναι τουλάχιστον όσο και το μέγεθος  $k$  της cache.
  - Έστω ότι οι cache του *on-line* και του *off-line* περιέχουν τις ίδιες  $k$  σελίδες, ενώ η πρώτη αίτηση είναι για σελίδα άγνωστη (miss)
  - Ονομάζουμε  $S$  το σύνολο αυτών των  $k+1$  σελίδων
  - Η ακολουθία αιτήσεων  $\sigma$ :
    - αιτείται εκείνη η σελίδα από το  $S$  που δεν υπάρχει στην cache (πάντοτε υπάρχει τέτοια)

# Δυσκολία caching (συν.)

- Ο *off-line* χωρίζει την  $\sigma$  σε μέγιστες υπακολουθίες (γύρους)  $k$  διακριτών σελίδων, και κάθε φορά διώχνει αυτήν που θα ζητηθεί πρώτη στον επόμενο γύρο. Πχ,
  - 4      2      3      /**1**      4      3      /**2**      4      1      /**3**...
  - {1,2,3},{1,4,3},{1,4,2}/ {3,4,2},{3,1,2},{4,1,2}|{4,1,3}, ... (*on-line*)
  - {1,2,3},{**4**,2,3},{4,2,3}|{4,**1**,3},{4,1,3},{4,1,3}| {4,1,**2**}, ... (*off-line*)
- Άρα:
  - Ο *on-line* έχει μία αποτυχία ανά αίτηση
  - Ο *off-line* έχει μία αποτυχία ανά γύρο (δηλ., μία ανά  $k$  αιτήσεις)

# LRU

- Προσπαθεί να εκμεταλλευτεί την χρονική «τοπικότητα» των σελίδων για να διαμορφώσει πολιτική εξώσεως
- Αιτεί μία ουρά προτεραιότητας, με αύξοντες αριθμούς κλειδιού-αντιπροσωπευτικούς της χρονικής στιγμής τελευταίας χρήσης (εύκολο)
- Έχει απόδοση:  $f_{LRU} \leq kf_O + k$ 
  - Η  $\sigma$  χωρίζεται σε υπακολουθίες  $\tau_0 \tau_1 \tau_2 \dots \tau_m$  ώστε η  $\tau_0$  να περιέχει το πολύ  $k$  αποτυχίες LRU και οι υπόλοιπες ακριβώς  $k$
  - Σε κάθε  $\tau_i$  η *off-line* πρέπει να σημειώνει τουλάχιστον μία αποτυχία, με εξαίρεση ίσως την  $\tau_0$ :
    - παρατηρήστε πως σε κάθε υπακολουθία  $\tau_i$ , λόγω του τρόπου λειτουργίας της LRU, θα υπάρχουν ανακτήσεις **τουλάχιστον**  $k+1$  διακεκριμένων σελίδων
    - Κατά την  $\tau_0$ , εάν παρατηρηθεί διπλή ανάκτηση, τότε υπάρχουν ανακτήσεις **τουλάχιστον**  $k+1$  διακεκριμένων σελίδων. Διαφορετικά, ίσως η *off-line* γλιτώσει την αποτυχία
  - Εάν οι δύο cache έχουν διαφορετικά περιεχόμενα, τότε η *off-line* πολιτική μπορεί να γλιτώσει το πολύ  $k$  ανακτήσεις



# FiFo

- Εύκολη υλοποίηση με μία λίστα
- Ίδια πολυπλοκότητα με LRU (παρόμοιο επιχείρημα)

# LFU

- Απαιτεί την δυσκολότερη υλοποίηση (ουρά μεγίστου)
- Δεν είναι  $k$ -ανταγωνιστική (!):

Έστω η ακολουθία

$$\tau_1 = \sigma_1 \sigma_1 \dots \sigma_1 \sigma_2 \sigma_2 \dots \sigma_2 \dots \sigma_{k-1} \sigma_{k-1} \dots \sigma_{k-1}$$

των  $l > k$  προσπελάσεων σε  $k-1$  διαφορετικές σελίδες και

$$\tau_2 = \sigma_k \sigma_{k+1} \sigma_k \sigma_{k+1} \dots \sigma_k \sigma_{k+1}$$

μία ακολουθία των  $2l-2$  προσπελάσεων σε δύο νέες σελίδες.

Τότε, η σύνθετη ακολουθία αιτήσεων

$$\Sigma = \tau_1 \tau_2$$

- οδηγεί την LFU, μετά τις πρώτες  $l(k-1)$  ανακτήσεις, να αποτυγχάνει σε κάθε αίτηση ( $2l-2$ , συνολικά)
- ο βέλτιστος να αποτυγχάνει μία φορά
- Άρα, κατά την  $\Sigma$ , ο λόγος LFU/off-line =  $2(l-1) > l > k$
- Έστω  $\Sigma' = \Sigma \Sigma \Sigma \dots \Sigma \Sigma$  αυθαίρετα μεγάλη ακολουθία. Σε αυτήν, ο λόγος LFU/off-line  $> l > k$

# LiFo

- Δεν είναι  $k$ -ανταγωνιστική (!):

Έστω η ακολουθία  $n$  ανακτήσεων

$$\tau = \sigma_1 \sigma_2 \dots \sigma_{k-1} \sigma_k \sigma_{k+1} \sigma_k \sigma_{k+1} \sigma_k \sigma_{k+1} \dots \sigma_k \sigma_{k+1} \dots$$

Τότε:

- Η LiFo, μετά την  $k$ -στή αίτηση, αποτυγχάνει συνεχώς
- Η *off-line* το πολύ  $k+1$  φορές
- $f_{\text{LiFo}}/f_o \geq (n-k)/k = n/k - 1 > k$

# Πολιτική Μαρκαδόρου (Marker)

- Τι ισχύει (χωρίς απόδειξη):

*Για κάθε τυχαίο αλγόριθμο caching, υπάρχει ακολουθία με λόγο ανταγωνιστικότητας:*

$$c \geq H_k = \sum_{1 \leq k \leq j} 1/j$$

- Πολιτική Μαρκαδόρου

- Προσπαθεί να εξομοιώσει την πολιτική LRU

- Κάθε σελίδα έχει ένα μπιτ, αρχικά θεμένο στην τιμή ‘0’
- Εάν ζητηθεί μία σελίδα, τότε το μπιτ της γίνεται ‘1’
- Εάν πρέπει να εκδιωχθεί μία σελίδα, τότε διαλέγουμε μία από αυτές με μπιτ 0. Εάν όλες έχουν μπιτ 1, τότε
  - Μηδενίζουμε όλα τα μπιτ
  - Διαλέγουμε μία στην τύχη

# Ανάλυση

- Η πολιτική μαρκαδόρου είναι  $(2\log k)$ -ανταγωνιστική
  - Η ανάλυση γίνεται σε γύρους
  - Κάθε γύρος ξεκινά με όλες τις σελίδες να έχουν μηδενικά μπιτ, και τελειώνει όταν όλες έχουν σημειωθεί με αναμμένα μπιτ
  - Μία σελίδα χαρακτηρίζεται *μπαγιατική* ή *έωλη* (*stale*) εάν προϋπήρχε στην cache προ του γύρου
  - Αλλιώς *νωπή* (*fresh*)

# Ανάλυση (συν.)

- Εάν κατά τον  $i$ -στό γύρο ο μαρκαδόρος σημειώσει  $f_i$  προσπελάσεις νωπών σελίδων ( $f_i$  εξώσεις, δηλαδή) και  $b_i$  είναι οι σελίδες που υπάρχουν μόνον στην cache του βέλτιστου αλγορίθμου, τότε ο *off-line* θα κάνει τουλάχιστον  $f_i - b_i$  εξώσεις
- Επίσης, ο *off-line* αλγόριθμος πρέπει να κάνει και  $b_{i+1}$  αντικαταστάσεις για να είναι έτοιμος για τον επόμενο γύρο
- Άρα, με το πέρας του  $i$ -στού γύρου, ο *off-line* έχει διενεργήσει τουλάχιστον:

$$\max \{ f_i - b_i, b_{i+1} \} \geq (f_i - b_i + b_{i+1}) / 2$$

εξώσεις

# Ανάλυση (συν.)

- Συνεπώς, σε όλους τους γύρους ο *off-line* έχει διενεργήσει **τουλάχιστον**

$$\sum((f_i - b_i + b_{i+1})/2) = (b_{\kappa+1} - b_1)/2 + \sum f_i/2$$

εξώσεις

- Ο μαρκαδόρος χρειάζεται  $f_i + s_i$  εξώσεις, δηλ.,
  - Αυτές που δεν υπήρχαν προ του γύρου (φρέσκιες)
  - Συν το μ.ο.  $s_i$  των μπαγιάτικων που τις ξαναέφερε πίσω, αφότου εκδιώχτηκαν
- Με κάθε μπαγιάτικη σελίδα  $\sigma$ , έστω  $X_\sigma$  η τυχαία μεταβλητή δείκτης που είναι 1 εάν η σελίδα αιτείται μετά την εκδιώξή της, και 0 εάν όχι. Τότε:

$$s_i = \sum \mathbf{Prob}[X_\sigma = 1]$$

- $\mathbf{Prob}[X_\sigma = 1] \leq \#$  νωπών που υπάρχουν μέχρι της αιτήσεως της  $\sigma$  /  $\#$  απροσπέλαστων έωλων (κάθε αίτηση φρέσκιας σελίδας, διώχνει και μία μπαγιάτικη)  $\leq f_i / \#$  απροσπέλαστων έωλων (μεγιστοποίηση όταν όλες έχουν μπει)
- Καθώς σημειώνονται  $k \cdot f_i$  προσπελάσεις σε μπαγιάτικες σελίδες, προκύπτει πως:
$$s_i \leq f_i/k + f_i/(k-1) + \dots + f_i/(f_i+1) \leq f_i \sum 1/j = f_i H_k$$
- Οπότε,  $f_m = \sum_i (f_i H_k + f_i)$  είναι το αναμενόμενο πλήθος αντικαταστάσεων του μαρκαδόρου σε όλους τους γύρους
- Σχηματίζοντας τον λόγο  $f_m / f_o = \sum_i (f_i H_k + f_i) / ((b_{\kappa+1} - b_1)/2 + \sum f_i/2)$ , μετά από πράξεις, βγαίνει μικρότερος ή ίσος του  $c \log n$ ,  $c$  σταθερά