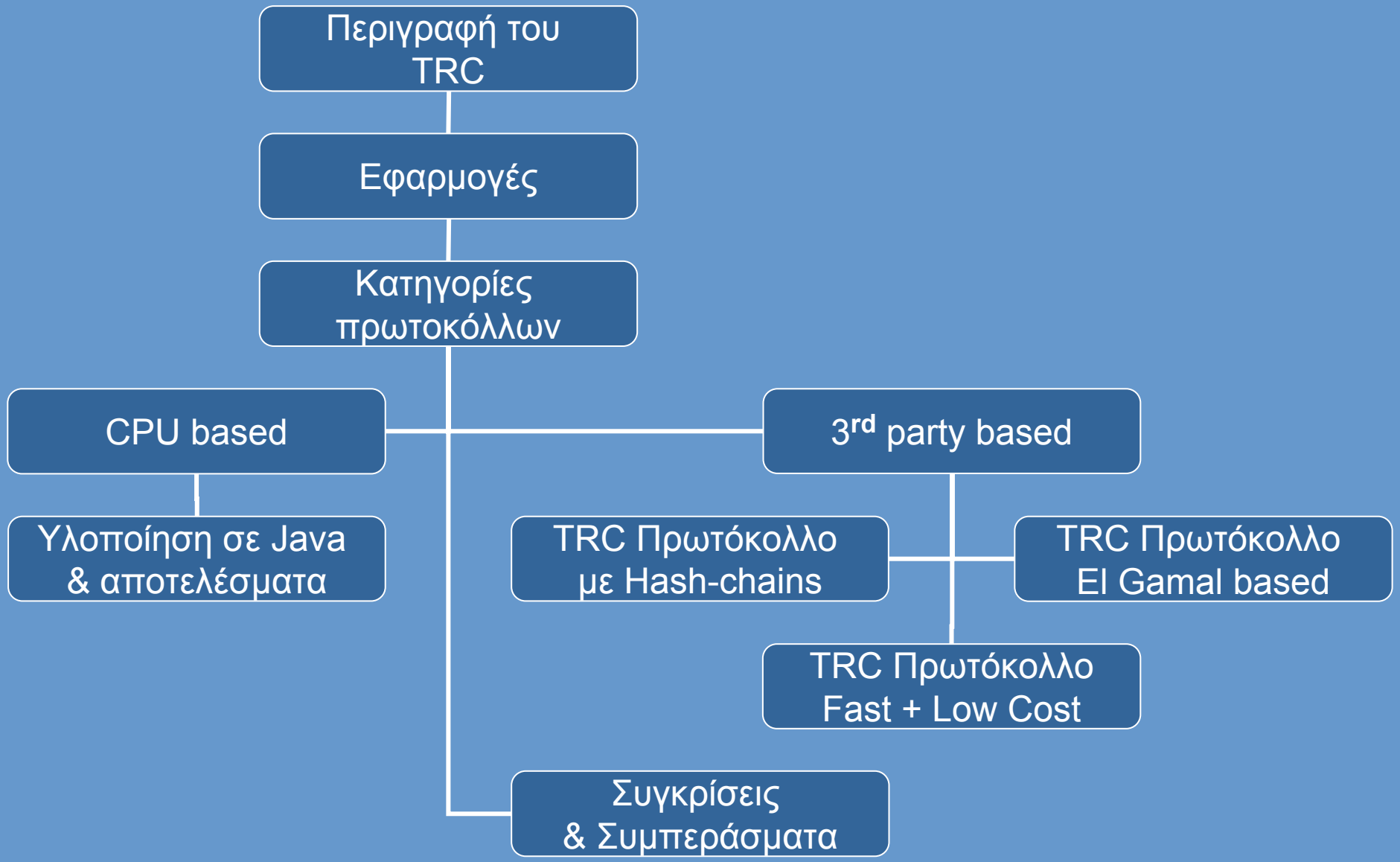
Διπλωματική εργασία

Σχεδίαση και ανάλυση κρυπτογραφικών πρωτοκόλλων με σκοπό την αποκρυπτογράφηση πληροφοριών σε συγκεκριμένες χρονικές στιγμές στο μέλλον. Timed-Release Cryptography (TRC)

Copyright © Κωνσταντίνος Γ. Χαλκιάς, Αύγουστος 2006

Υπεύθυνος Καθηγητής: Στεφανίδης Γεώργιος
Εξεταστές: Στεφανίδης Γεώργιος, Χατζηγεωργίου Αλέξανδρος
Μεταπτυχιακό τμήμα Εφαρμοσμένης Πληροφορικής
Πανεπιστήμιο Μακεδονίας

Οργάνωση παρουσίασης



Timed-Release Cryptography (TRC)

- Σκοπός → Κρυπτογράφηση ενός μηνύματος με στόχο την αποκρυπτογράφηση του σε μια ‘συγκεκριμένη’ χρονική στιγμή στο μέλλον.

  “sending information into the future”

- Γενικές Απαιτήσεις

→ Να μην επιτρέπεται η αποκρυπτογράφηση του κρυπτοκειμένου πριν από μια ‘αυστηρώς’ καθορισμένη χρονική στιγμή, συμπεριλαμβανομένου και του τελικού αποδέκτη.

→ Αν εμπλέκεται και τρίτη οντότητα (π.χ. agent, trusted server), τότε αυτή δεν πρέπει να έχει τη δυνατότητα εξόρυξης πληροφοριών που θα προδώσουν το περιεχόμενο του κρυπτοκειμένου.

Εφαρμογές που απαιτούν TRC (1)

- E-voting → Τα ψηφοδέλτια ανοίγουν μετά την λήξη της ψηφοφορίας.
- Κλειστές δημοπρασίες → Οι προσφορές παραμένουν κλειστές μέχρι τη λήξη της δημοπρασίας, ώστε κανείς από τους συμμετέχοντες να μη γνωρίζει τη προσφορά των ανταγωνιστών του.

Εφαρμογές που απαιτούν TRC (2)

- Διαγωνισμοί → π.χ. Internet programming contest όπου οι συμμετέχοντες **δεν** πρέπει να γνωρίζουν τις πληροφορίες του ζητούμενου προβλήματος πριν από την έναρξη του διαγωνισμού.

- Λοταρίες και online games → Οι παίχτες θα μπορούν να ελέγξουν την ορθότητα και την αμεροληψία του αποτελέσματος.

Εφαρμογές που απαιτούν TRC (3)

- Καθυστερημένη εμφάνιση και επαλήθευση ενός υπογεγραμμένου εγγράφου → π.χ. διαθήκες, κληρονομιές, συμβόλαια.
- Καθυστερημένη εμφάνιση μηνυμάτων ή και της ταυτότητας του αποστολέα → π.χ. για λόγους ψυχαγωγίας (ευχετήρια, φιλικά ή ακόμα και ερωτικά SMS / e-mails).
- Ενεργοποίηση / Απενεργοποίηση κάποιων features σε software εφαρμογές → π.χ. anti-piracy control.

Κατηγορίες TRC πρωτοκόλλων

1. CPU based :

- Πρωτόκολλα βασισμένα σε αλγορίθμους που δε μπορούν να παραλληλοποιηθούν.
- Στηρίζονται επομένως στην υπολογιστική ισχύ του αποδέκτη.
- Δεν απαιτούν συμμετοχή τρίτης οντότητας.
- Δεν παρέχουν ακριβή ημ/νια αποκρυπτογράφησης.

2. 3rd party based :

- Πρωτόκολλα βασισμένα κυρίως σε τεχνικές ασύμμετρης κρυπτογράφησης.
- Απαιτούν συμμετοχή τρίτης οντότητας που αναλαμβάνει το ρόλο ενός έμπιστου server.

CPU based Protocols

- Βασίζονται στο πρόβλημα εκθετοποίησης mod *n*.
- Πιο συγκεκριμένα αν *n* είναι ένας σύνθετος αριθμός : $n = p \times q$ (p , q πρώτοι) τότε αν δε γνωρίζει κάποιος τη παραγοντοποίηση του *n* είναι δύσκολο να υπολογίσει το $a^{2^t} \pmod{n}$ όπου $gcd(a,n)=1$.
- Μέχρι στιγμής αν $t \ll n$ (*π.χ.* $n>2^{1024}$, $t<2^{100}$) δεν υπάρχει αλγόριθμος που να λύνει το πρόβλημα σε λιγότερο από *t* squarings.
- Καθώς το αποτέλεσμα κάθε squaring είναι απαραίτητο για την εκτέλεση του επόμενου squaring δεν μπορεί να γίνει παραλληλοποίηση.

Υλοποίηση CPU based πρωτοκόλλων

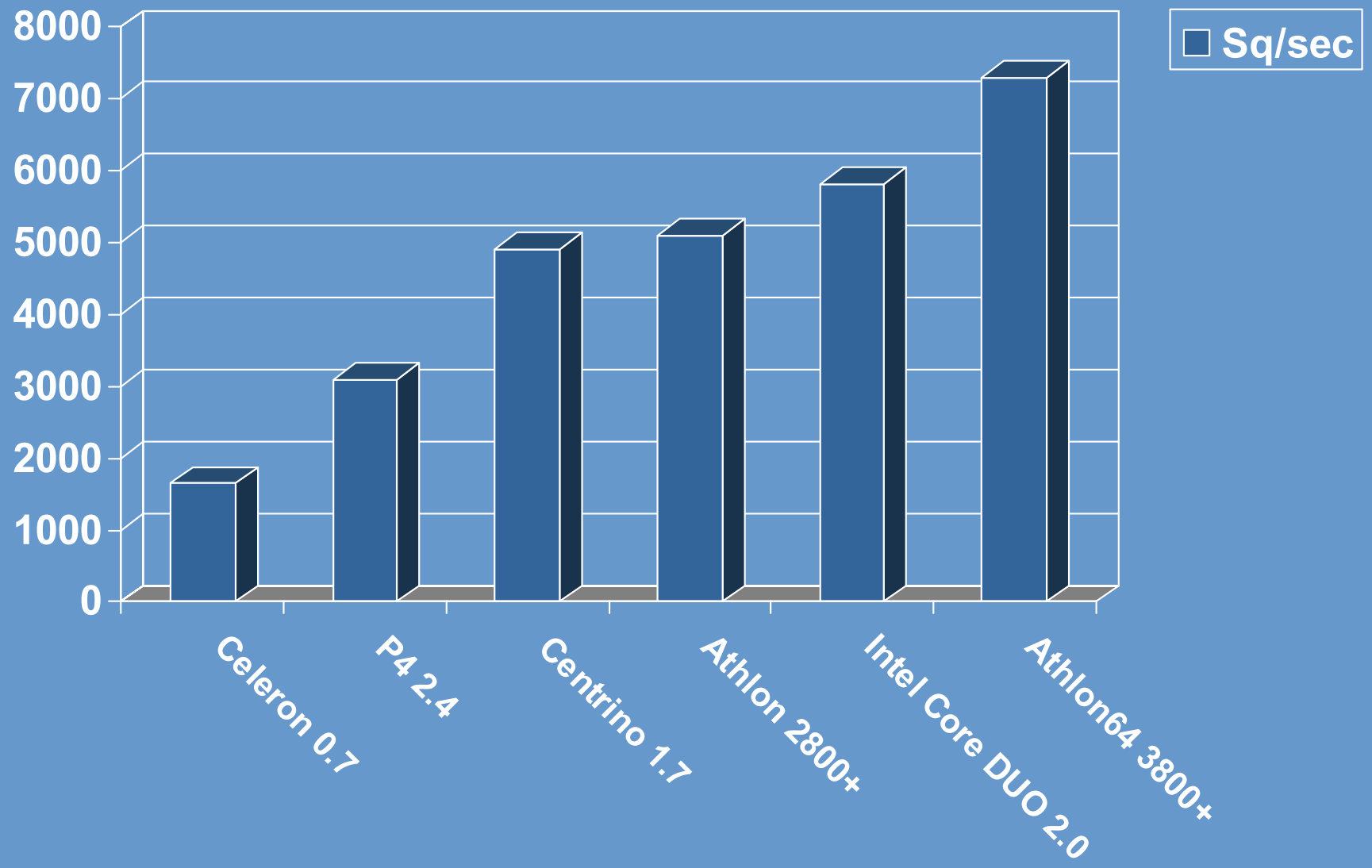
- Δημιουργία βιβλιοθήκης σε Java για τους ακόλουθους αλγορίθμους.

1. Time-Lock Puzzle των R. Rivest et. al
   (η πρώτη προσπάθεια για TRC)
2. Timed-Release crypto with Proven Membership του W. Mao
   (παρέχει διαβεβαίωση ότι το σωστό μήνυμα θα αποκαλυφθεί μετά από *t* squarings)
3. Timed-Release of an RSA signature του W. Mao

Συμπεράσματα εκτέλεσης (1)

- Καθώς ο χρόνος αποκρυπτογράφησης εξαρτάται από το μηχάνημα στο οποίο εκτελείται, ο αποστολέας πρέπει να γνωρίζει τα τεχνικά χαρακτηριστικά του hardware του παραλήπτη για να ορίσει έναν εκτιμώμενο αριθμό squarings.
- Για να είμαστε σίγουροι χρησιμοποιούμε ως βάση για τις μετρήσεις το ισχυρότερο μηχάνημα της εποχής, ικανοποιώντας παράλληλα το νόμο του Moore.
- Χρησιμοποιούν όλη την υπολογιστική ισχύ της CPU, δεσμεύοντας έτσι το μηχάνημα.

Συμπεράσματα εκτέλεσης (2)



3rd party or agent based protocols

- Τα πρώτα πρωτόκολλα απαιτούσαν τη συμμετοχή του server είτε στη διαδικασία κρυπτογράφησης, είτε στης αποκρυπτογράφησης.
- Πολλά από αυτά θυσίαζαν την ανωνυμία των αποστολέων.
- Σε μερικά IBE (Identity Based Encryption) συστήματα ο server μπορεί να εξάγει σημαντικές πληροφορίες για τα μηνύματα.
- Μια μεγάλη πλειοψηφία των TRC πρωτοκόλλων απαιτεί διαφορετική κατασκευή κλειδιών (κυρίως των δημόσιων κλειδιών), με αποτέλεσμα να μην είναι συμβατά με τις υπάρχουσες μεθόδους κρυπτογράφησης όπως με το πρωτόκολλο El Gamal και την ανταλλαγή κλειδιών Diffie-Hellman.
- Κανένα σύστημα δε λύνει ταυτόχρονα το πρόβλημα των προηγούμενων και επόμενων κλειδιών και ο χρήστης πρέπει να τα αναζητήσει στο server.

TRC based on Bilinear Pairings

- Υποθέτουμε ότι G_1 είναι μια προσθετική κυκλική ομάδα και P ο γεννήτορας αυτής. Η τάξη της ομάδας αυτής είναι q (prime number). Επιπλέον, G_2 είναι μια πολλαπλασιαστική κυκλική ομάδα ίδιας τάξης. Ορίζουμε ως bilinear mapping το $\hat{e}: G_1 \times G_1 \rightarrow G_2$ αν και μόνο αν ικανοποιούνται οι εξής ιδιότητες :

1. Bilinear : $\hat{e}(aP,bQ)= \hat{e}(abP,Q)= \hat{e}(P,Q)^{ab}$
2. *Non-Degenerate : Υπάρχουν P,Q όπου $\hat{e}(P,Q)\neq 1$*
3. *Efficient : Υπάρχει ευφυής αλγόριθμος που υπολογίζει το bilinear map*

TRC με Hash Chains

- Το προτεινόμενο πρωτόκολλο χρησιμοποιεί τη τεχνική του S/Key password system για τη κατασκευή των ιδιωτικών κλειδιών που αντιπροσωπεύουν μια χρονική στιγμή σε συνδυασμό με τα bilinear pairings που χρησιμοποιούνται κυρίως για authentication.
- Η καινοτομία έγκειται στο ότι :

1. Ο server δημοσιεύει μόνο μια μικρή πληροφορία (έναν ακέραιο π.χ. 160 bits) όταν φτάσει η συγκεκριμένη ημ/νια και όχι ένα πλήθος αριθμών και σημείων όπως στα περισσότερα ανταγωνιστικά πρωτόκολλα.
2. Λύνει το πρόβλημα των προηγούμενων time-private keys, καθώς με τη μέθοδο του συνεχόμενου hash μπορεί κάποιος να παράγει τα προηγούμενα time-keys.

TRC με Hash chains: Συμβολισμοί

H_1: $\{0,1\}^* \rightarrow G_1$ (message2point)

H_2: $\{0,1\}^* \rightarrow \{0,1\}^n$ (π.χ. MD5 product)

$H_2^n(x)$ είναι το αποτέλεσμα αν γίνει συνεχόμενο hash του *x* για *n* φορές

- Δημόσια και ιδιωτικά κλειδιά

Server : private → *s* (ακέραιος)

public → *G*, *sG*

Χρήστες : private → *a* (ακέραιος)

public → *aG*, *asG*

TRC με Hash chains: Time Keys (1)

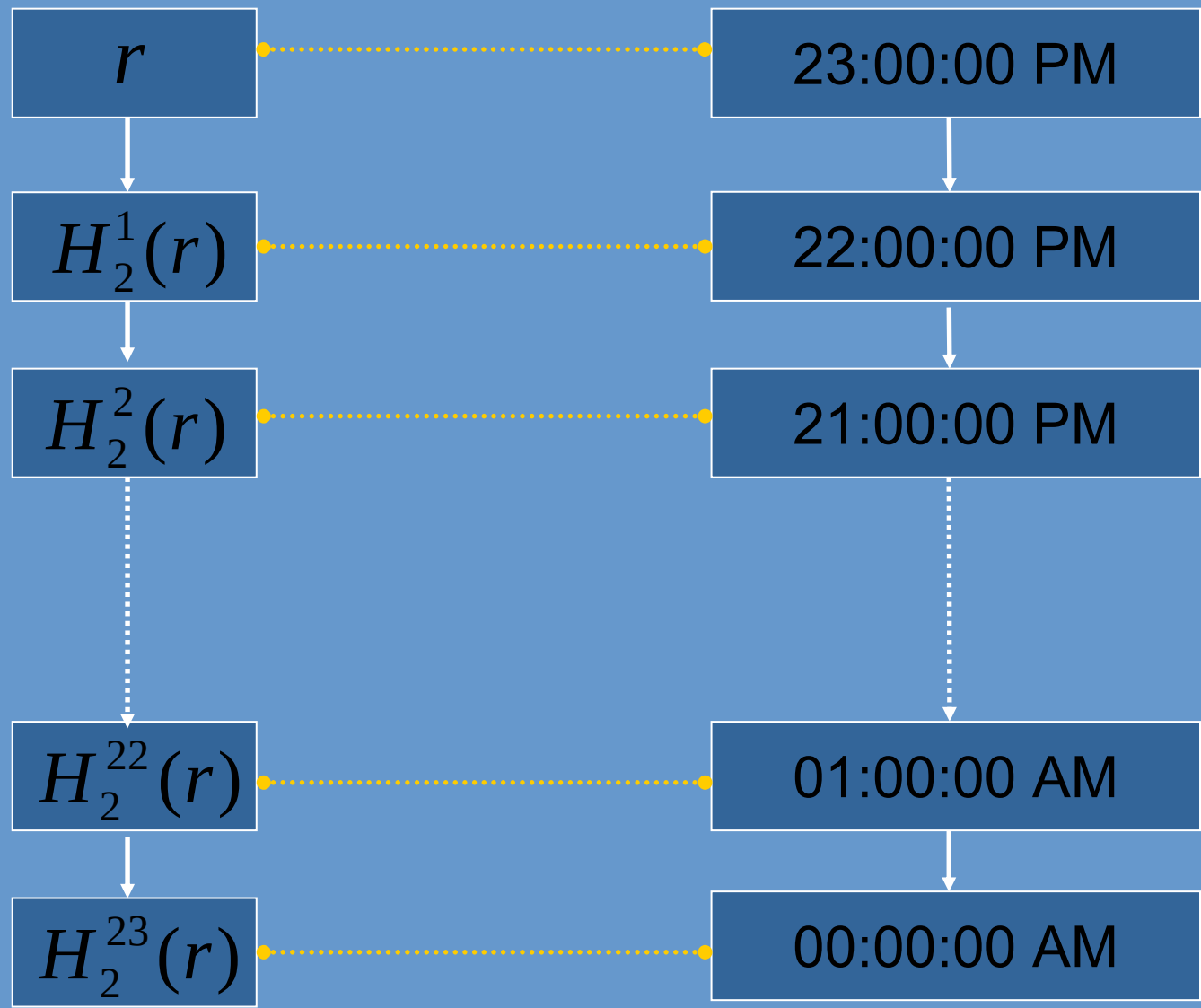
- Δημιουργία λίστας των time private keys.

Έστω ένας server που πρέπει να δημοσιεύσει κλειδιά ανά μία ώρα.

Αν θεωρήσουμε ότι ξεκινάμε από τις 00:00:00 AM Aug 30, 2006 GMT και χρειαζόμαστε κλειδιά μέχρι τις 23:00 AM Aug 30, 2006 GMT τότε ο server εκτελεί την εξής διαδικασία :

Επιλέγει ένα τυχαίο ακέραιο r 160 bits και εφαρμόζει την H_2 23 φορές. Τότε η λίστα που δημιουργείται περιέχει τα ιδιωτικά κλειδιά για κάθε ώρα, αλλά το σημαντικό είναι ότι **η αντιστοίχηση γίνεται αντίστροφα.**

TRC με Hash chains: Time Keys (2)



TRC με Hash chains: Time Keys (3)

- Δημιουργία λίστας των time public keys

Καταρχήν χρησιμοποιώντας την H_1 αντιστοιχούμε το string που αντιπροσωπεύει κάθε ώρα σε ένα σημείο της ελλειπτικής καμπύλης. Για παράδειγμα $T_1 = H_1$('00:00:00 AM Aug 30, 2006 GMT').

Έπειτα το time public key για αυτή τη μέρα είναι τα στοιχεία rT_1 και rsG *(sG είναι το public του server)*, όπου *r,* όπως έχουμε πει, είναι το ιδιωτικό κλειδί για αυτή την ώρα. Με αυτό τον τρόπο κατασκευάζονται και δημοσιεύονται από πριν όλα τα time public keys.

TRC με Hash chains: Encryption (1)

- Έστω ο χρήστης A θέλει να στείλει στο χρήστη B ένα μήνυμα *M* που θα ανοίξει την χρονική στιγμή *T*.

1. Παίρνει από το server το time public key για τη χρονική στιγμή *T* (*rT*, *rsG*) και ελέγχει την ορθότητα του :

$$\hat{e}(T,rsG)= \hat{e}(rT,sG) \Rightarrow \hat{e}(T,G)^{rs} = \hat{e}(T,G)^{rs}$$

2. Ελέγχει την ορθότητα του κλειδιού του παραλήπτη :

$$\hat{e}(bG,sG)= \hat{e}(G,bsG) \Rightarrow \hat{e}(G,G)^{sb} = \hat{e}(G,G)^{sb}$$

TRC με Hash chains: Encryption (2)

3. Επιλέγει έναν τυχαίο αριθμό *k.*
4. Κατασκευάζει το συμμετρικό κλειδί ως εξής : SK = $\hat{e}(rT,bG)^{ak} = \hat{e}(T,G)^{abkr}$
5. Αποστέλλει το κρυπτοκείμενο

$$C = < kT, M \oplus SK >.$$

TRC με Hash chains: Decryption

1. Ο παραλήπτης λαμβάνει από το server το ιδιωτικό κλειδί r που αντιστοιχεί στη χρονική στιγμή T.
2. Υπολογίζει το συμμετρικό κλειδί ως :

$$SK^* = \hat{e}(kT,aG)^{br} = \hat{e}(T,G)^{abkr}$$

3. Αποκρυπτογραφεί το M ως :

$$SK^* \oplus (M \oplus SK) = M$$

El Gamal based TRC Πρωτόκολλο (1)

- Στο πρωτόκολλο αυτό τα κλειδιά των χρηστών είναι αυτά που χρησιμοποιούνται και στον αλγόριθμο του El Gamal σε ελλειπτικές καμπύλες δηλ.
- Δημόσια και ιδιωτικά κλειδιά (γεννήτορας G)

Server : private $\rightarrow s$ (ακέραιος)

public $\rightarrow sG$

Χρήστες : private $\rightarrow a$ (ακέραιος)

public $\rightarrow aG$

El Gamal based TRC Πρωτόκολλο (2)

- Όσον αφορά όμως τα time keys η φιλοσοφία είναι εντελώς διαφορετική. Σε αυτή τη περίπτωση το time public key είναι το $T = H_1(\text{time})$ δηλαδή, ο καθένας μπορεί να το υπολογίσει και να κάνει encryption χωρίς να έρθει σε επαφή με το server.

- Το time private key είναι το sT που μόνο ο server μπορεί να υπολογίσει.

El Gamal based TRC: Encryption

1. Ο χρήστης A επιλέγει ένα τυχαίο αριθμό k
2. Υπολογίζει το $SK = \hat{e}(T,kG+sG)^a = \hat{e}(T,G)^{(k+s)a}$
3. Στέλνει 3 ποσότητες :

$$kG,\ kT+kbG \text{ και το } M \oplus SK$$

El Gamal based TRC: Decryption

1. Ο Β έχοντας λάβει το kG, υπολογίζει το bkG
2. Αφαιρεί αυτό που βρήκε από τη δεύτερη ποσότητα που έλαβε και παίρνει το kT δηλαδή: $(kT+rbG) - bkG = kT$
3. Μόλις λάβει από τον server το sT υπολογίζει το μυστικό κλειδί :

$$SK^* = \hat{e}(kT+sT,aG) = \hat{e}(T,G)^{(k+s)a} = SK$$

4. Αποκρυπτογραφεί το M όπως και πριν

FAST TRC Protocol

- Το πρωτόκολλο αυτό χρησιμοποιεί την ίδια λογική με το προηγούμενο και η κατασκευή των κλειδιών των χρηστών, αλλά και των time κλειδιών είναι ακριβώς η ίδια.
- Όσον αφορά το κόστος επικοινωνίας, αλλά και το υπολογιστικό κόστος, φαίνεται ότι υπερτερεί έναντι οποιουδήποτε άλλου υπάρχοντος TRC συστήματος, χωρίς να χάνει σε λειτουργικότητα και ασφάλεια.

FAST TRC Protocol: Encryption

1. Ο χρήστης A επιλέγει έναν τυχαίο αριθμό r
2. Υπολογίζει το συμμετρικό κλειδί ως:

$$SK = \hat{e}(T, rbG + sG)^{a} = \hat{e}(T, G)^{(rb+s)a}$$

3. Αποστέλλει τα r, $M \oplus SK$

FAST TRC Protocol: Decryption

1. Ο Β έχοντας λάβει το *r* υπολογίζει το *rbT*
2. Λαμβάνοντας από τον server το time private key *sT*, υπολογίζει το συμμετρικό κλειδί ως :

$$SK^* = \hat{e}(rbT+sT,aG)= \hat{e}(T,G)^{(rb+s)a}=SK$$

3. Αποκρυπτογραφεί το μήνυμα όπως και στις προηγούμενες περιπτώσεις.

TRC Protocols: Results

Protocol	Communication cost	Computational cost
Blake and Chan	1 Pt + 1 Msg (+2 Pt Public Keys)	Enc:1 pair + 2mul + 1H Dec:1 pair + 1 pow + 1H
Osipkov et. al	1 Pt + 2 Msg	Enc:1 pair + 2mul + 1H Dec:1 pair + 2 mul+ 1H
Proposed Hash Chains	1 Pt + 1 Msg (+ 2Pt Public keys + 2Pt time-keys)	Enc:1 pair + 1 mul + 1 pow + 1H Dec:1 pair + 1 pow +1H
Proposed El Gamal	2Pt + 1 Msg	Enc:1 pair + 2 mul + 1 pow + 1H Dec:1 pair + 1 mul + 1H
Proposed Fast TRC	1 Msg + 1num	Enc:1 pair + 1 mul + 1 pow + 1H Dec:1 pair + 1 mul + 1H

TRC … Τέλος

Q&A

Ευχαριστώ,

Κωνσταντίνος Γ. Χαλκιάς