Elliptic Curve Cryptography
Concepts, Implementation and Challenges

Dec 11, 2015

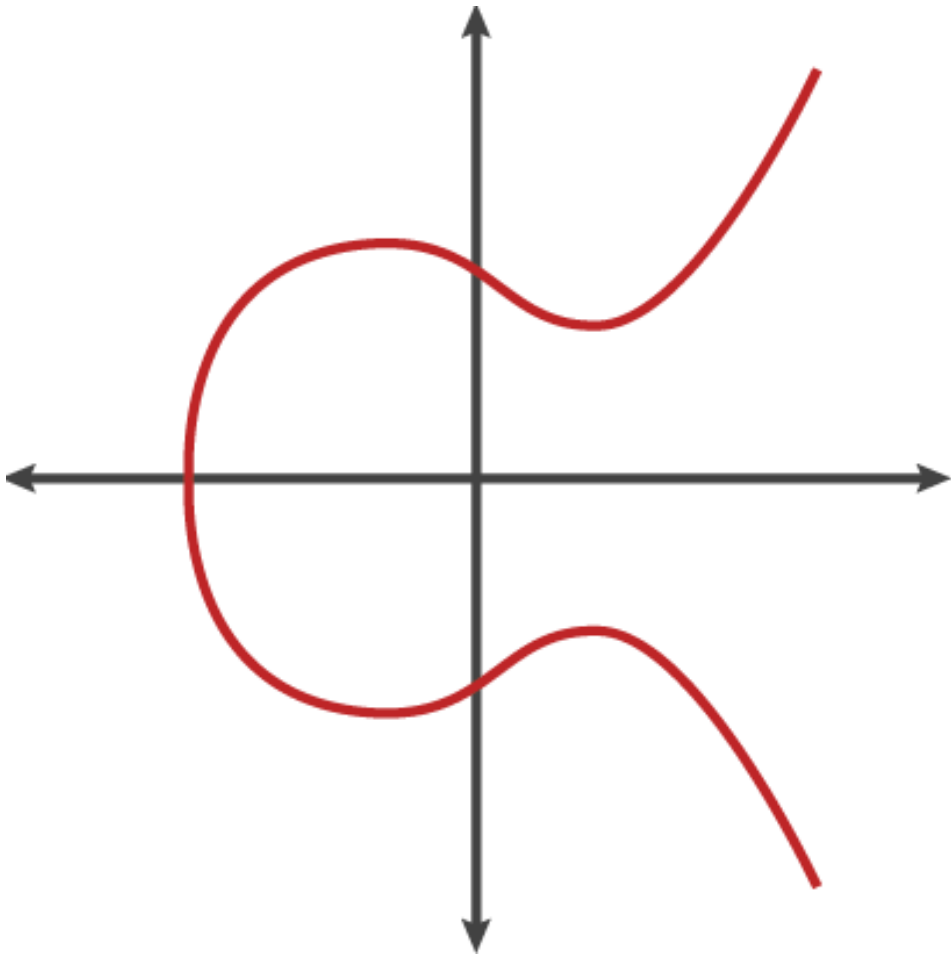
Rosy Sunuwar, Suraj Ketan Samal
rsunuwar@cse,unl.edu
ssamal@cse.unl.edu

Agenda

- Elliptic Curves and Visualization
- Elgamal encryption using *ECC*(E^3C^2) and comparison with other cryptosystems
- ECC Attacks
- Challenges and Future Work

Elliptic Curves

- Elliptic Curves has form:-
  - $y^2 = x^3 + ax + b$



- It has special properties
  - Symmetry in x-axis
  - Any non-vertical line intersects the curve in at most three places.

Elliptic Curve Arithmetic

- Point Addition:
  - draw a line between the two points(A & B) and find the third intersection point(C') and reflect it on X-axis to find C

    C = A + B
  - If A = B, then draw tangent at the point A and do same as above

    C = A + A
- Point Subtraction:
  - Defined by negative addition.

Elliptic Curve Arithmetic

- Point Multiplication:
  - repeated addition

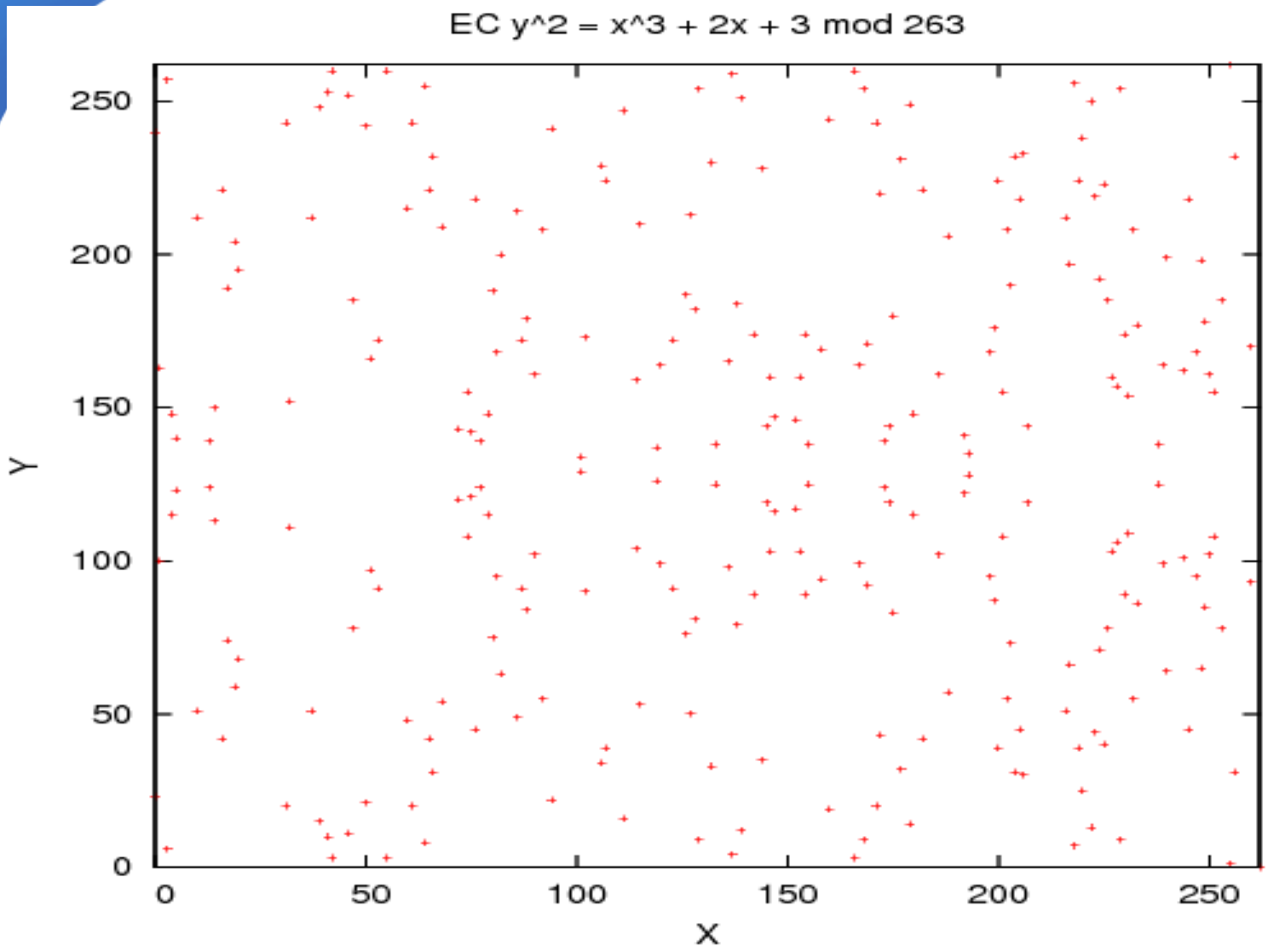
    $3A = 2A + A = A + A + A$

- Point Division (or inverse of multiplication):
  Discrete logarithm problem !

ECC Visualization using JavaPlot

- Elliptic curve used:-
  - $y^2 = x^3 - x + 1$

Elliptic Curve over Finite Field



Elliptic Curve Cryptography

- Uses Elliptic Curve over finite field
  - $y^2 \equiv x^3 + ax + b$
- Based on Elliptic Curve Discrete Logarithm Problem(ECDLP)
  - given points *P* and its multiple *kP* on an elliptic curve, it is computationally hard to find “k”
- Used as trapdoor function

Elgamal Encryption using ECC (E^3C^2)

- Based on the fact that it is hard to find *a* given *a.*P and P on elliptic curve C

  a – can act as hidden secret!!! (*private key*)

  α = *P* , β = *a.P* or a.α ,*C* publicly exposed.

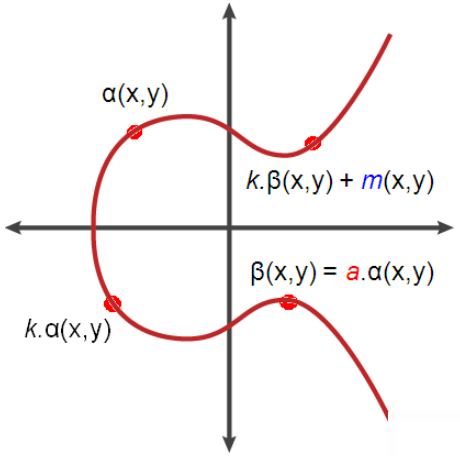
- (α ,β) can be used to hide a message *m* by choosing a random *k* to obtain (k.α ,k.β + *m*), calling *k* as a session secret.

- *m* can then be recovered by using

  (k.β + m) – a.(k.α) = (k.β + m) – k.(a.α) = m

[Trappe,Washington p363-364]

E^3C^2 – Cryptosystem



Public (α ,β)

m

Abbie(A)

(*k*α ,*k*β + *m*)

Private (*a*)

β = *a*α

Brooke(B)

m = (kβ + *m*) – a.(kα)

Ceaser (C)

DLP

(*k*α, α) => k = ?

(*a*α, α) => *a* = ?

E^3C^2 – Encryption using JECC

- *Abbie*(A) wants to send a set of secure messages M_j (j = 0,1,2…) to *Brooke(B)*. Each message M_j is padded to make it a multiple of 20 bytes and broken into small blocks of 20 bytes.

$$M_j = m_0 m_1 m_2 m_3 \ldots m_N \quad (m_i \text{ is 20 bytes})$$

- Using a random k_j , for each block m_i, use Brooke's public key (α ,β) to generate

$$e_i = H(k_j.\beta) \oplus m_i$$

where H(x) = SHA-1 function (160 bits)

and construct the encoded message

$$E_j = k_j.\alpha \,||\, e_0 e_1 e_2 \ldots e_N$$

http://jecc.sourceforge.net/

E^3C^2 – Decryption using JECC

- *Brooke(B)* breaks the encoded message into

$$k_j.\alpha \text{ and } e_i = H(k_j.\beta) \oplus m_i$$

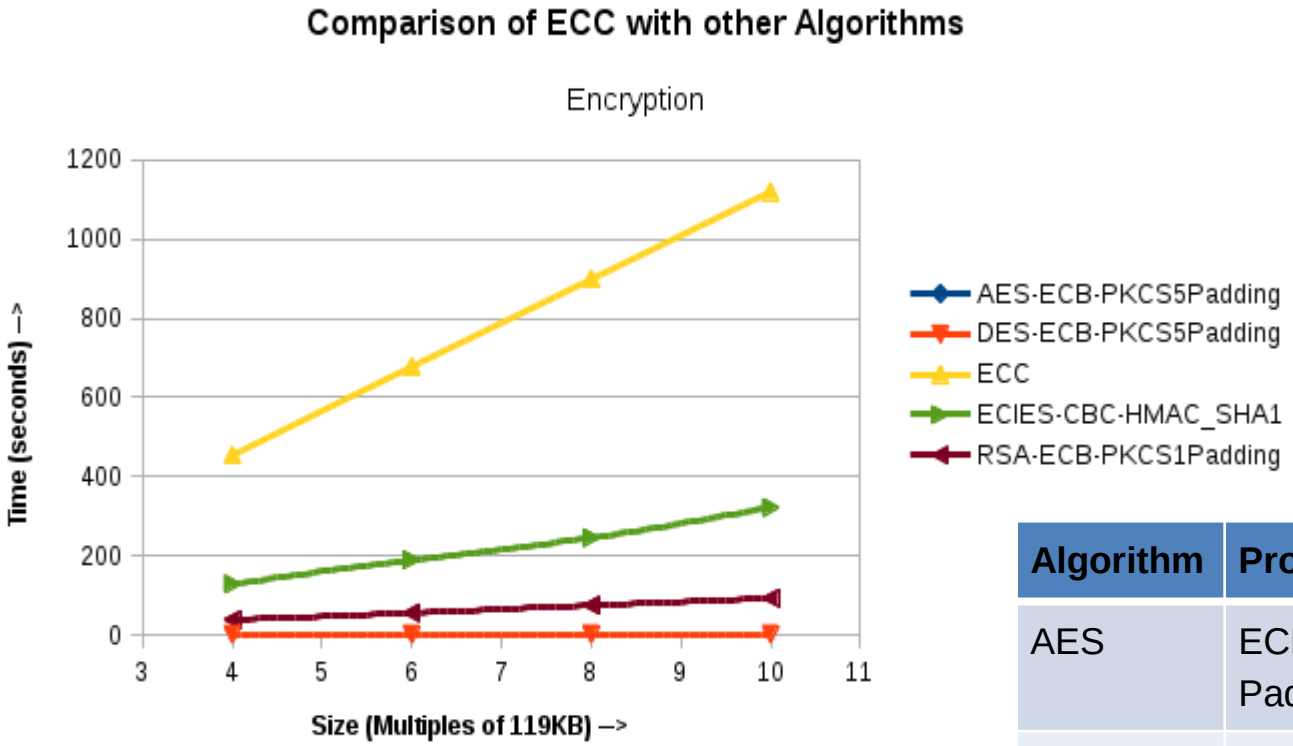
and uses its private key to compute each block using

$$H(a.\ k_j.\alpha) \oplus e_j$$
$$= H(k_j.\beta) \oplus H(k_j.\beta) \oplus m_i = m_i$$

and then constructs

$$M_j = m_0 m_1 .. m_N$$

E^3C^2 – Comparison with other Cryptosystems



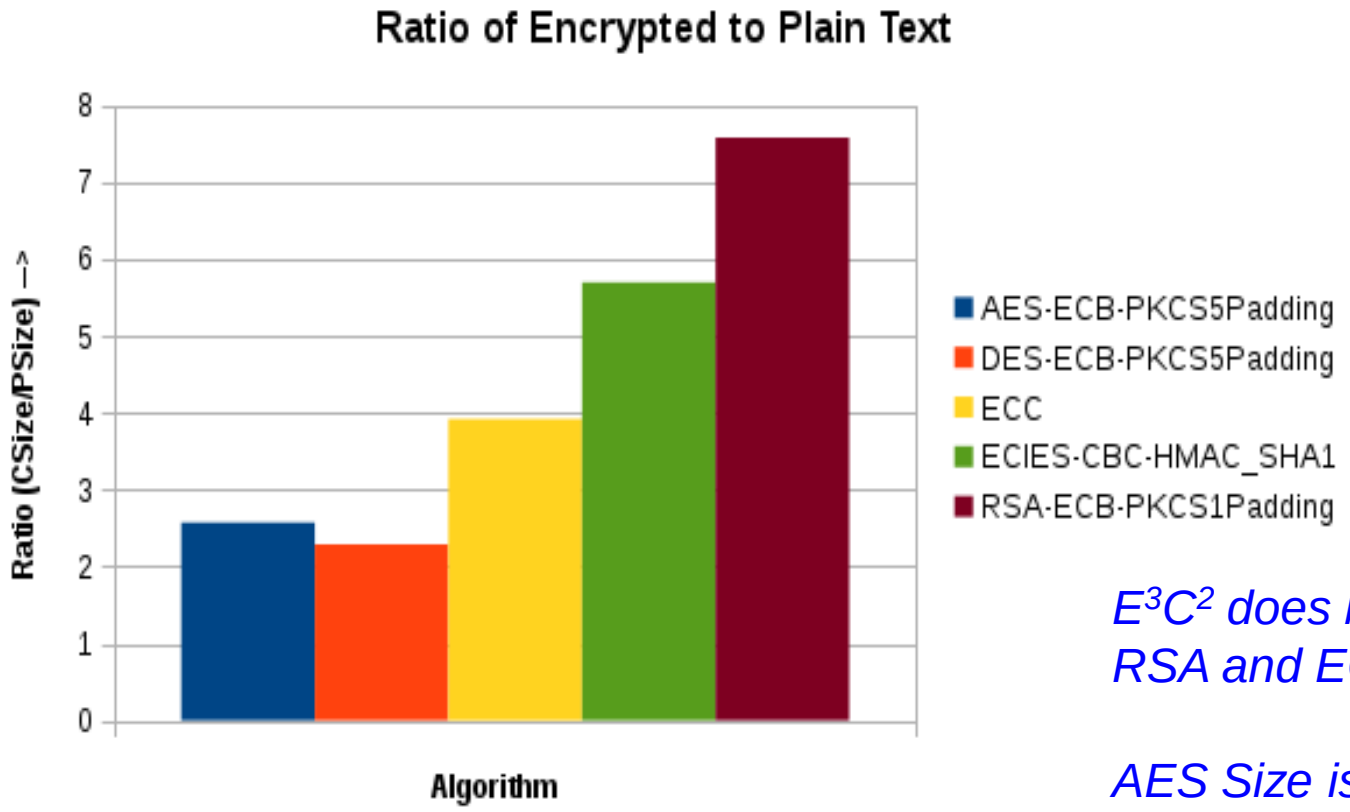
AES ~297 times faster than E^3C^2 on 1.13MB file (increases with data-size)

Even RSA came out ~11 times faster than E^3C^2

Algorithm	Properties	Key Size
AES	ECB/PKCS5 Padding	128 bit
DES	ECB/PKCS5 Padding	56 bit
RSA	ECB/PKCS1 Padding	1024 bit
ECIES	CBC/AES128_ HMAC_SHA1	256 bit
E3C2	ECC_SHA1	256 bit

E^3C^2 – Comparison with other Cryptosystems

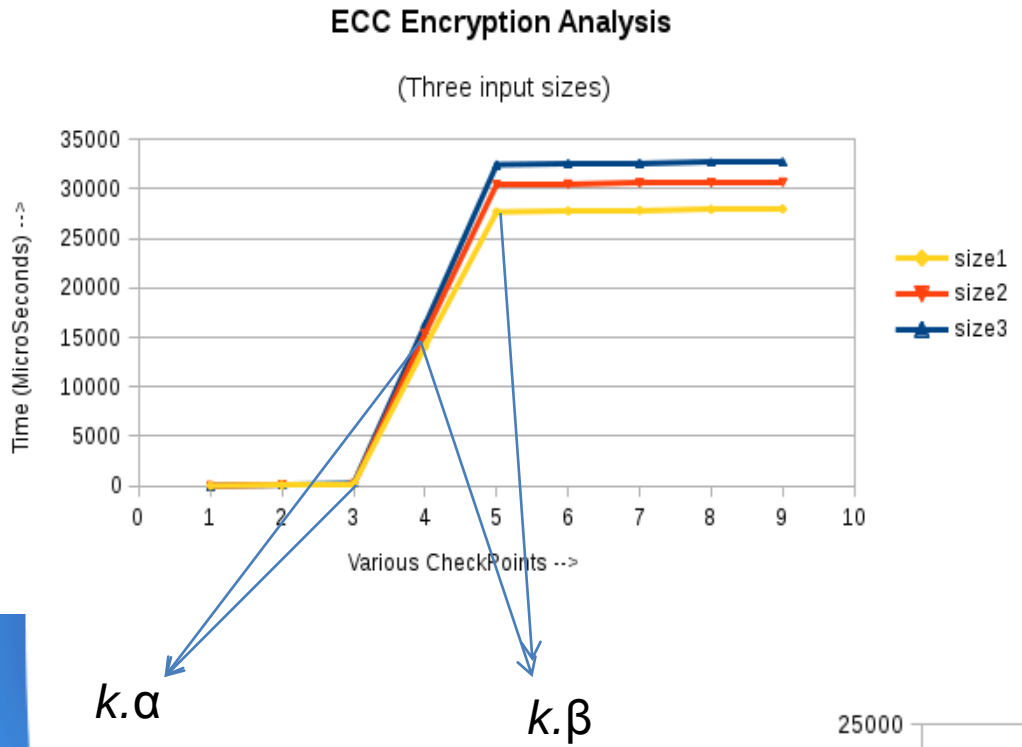
- Cipher Text to Plain Text Ratio:
  - Important as it's the encrypted text that is sent!!



E^3C^2 does better than RSA and ECIES!!

AES Size is still ~3 times smaller than E^3C^2

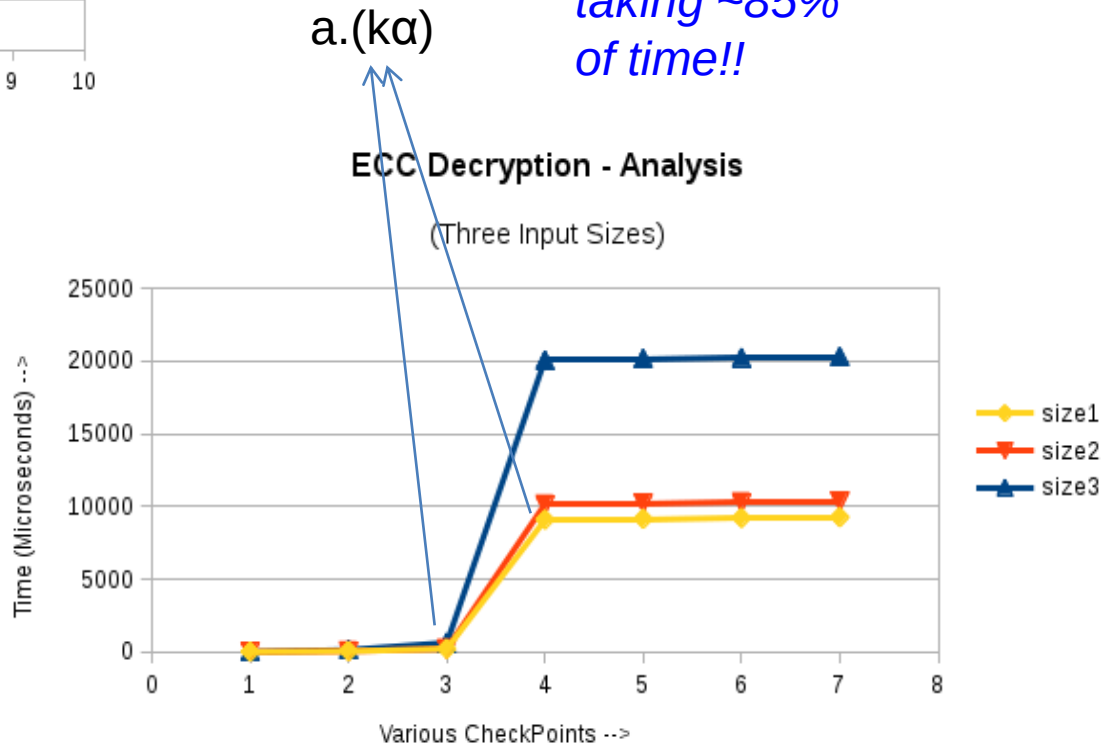
E^3C^2 – Encryption Analysis



Multiplications are expensive!!!!

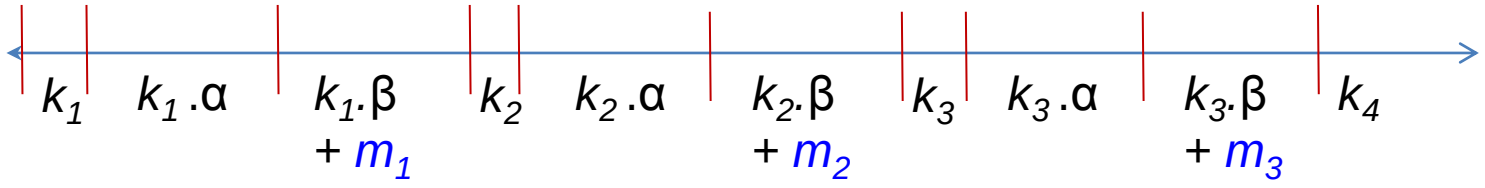
2 multiplications taking ~90% of total encryption time !!

1 multiplication taking ~85% of time!!



E^3C^2K – A modified version

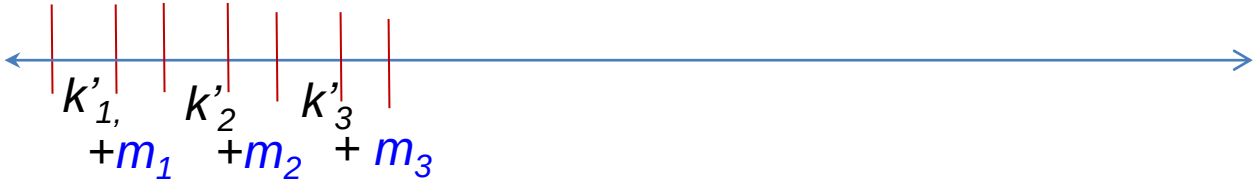
Public (α ,β)



Abbie(A)

k_2	$k_2.\alpha$	$k_2.\beta$
k_3	$k_3.\alpha$	$k_3.\beta$
k_4	$k_4.\alpha$	$k_4.\beta$
k_5	$k_5.\alpha$	$k_5.\beta$

k- tuple

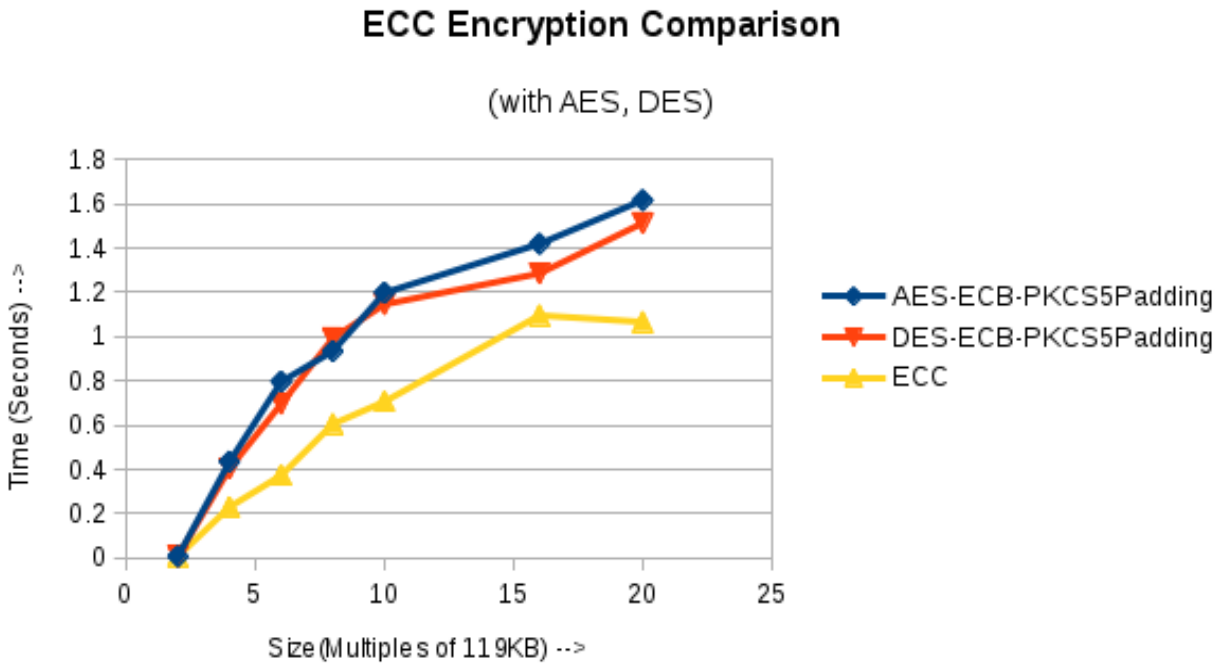


k’ - a random tuple from *k*-table

k-table

pre-computed

E^3C^2K – Comparison with others



E^3C^2 encryption turns out to be faster than even AES and DES

Caveats:

- Assumes k-table generation is done before-hand.
- Additional memory requirements. Needs to be sufficiently large for effective randomization. Even more for multiple *recievers* !!!

E^3C^2K – Fast Multiplication and *k*-tables

A closer look at calculation of *k.P* from *P*

cache

2^0	2^0P
2^1	2^1P
2^2	2^2P
2^t	2^tP

t additions

k-table

k_2	$k_2.\alpha$	$k_2.\beta$
k_3	$k_3.\alpha$	$k_3.\beta$
k_4	$k_4.\alpha$	$k_4.\beta$
k_5	$k_5.\alpha$	$k_5.\beta$

For S messages

Binary*(k)* = b*t*.....b*2*b*1*b*0*

or k= $2^t + 2^{t\text{-}x1} + \ldots + 2^{t\text{-}xn.}$
(for every non-zero bit b*i*)

Computing *k.P* =
2.t additions.

For *S* messages =
2.S.t additions.

Cache created only once for all *S* messages, hence *t* additions.

For S messages =
t + *S.t* = *(S+1).t* additions only.

E^3C^2 – Decryption Analysis

- Dominated by one multiplication

  $a.(k.\alpha) = a.r$ (lets say)

- *a* is private and known before-hand, *r* changes for every message. *i.e* we know the number of times to add *r*, but not *r.*

- Can this fact be somehow used to make decryption faster ? (*no luck yet!!!*).

- Choose *a* with special property so that *a.r* is easy to calculate. (*does this make ECC less secure ??*)

ECC Attacks

- ECC attacks are attacks on ECDLP problem
- Given (α, β) where β=aα in elliptic curve C, tries to calculate “a”
- Algorithms
  - ➢ Linear Search (O(n))
  - ➢ Baby-step giant-step (O(√n))
  - ➢ Pollard’s ρ (O(√n))

Linear Search (O(n))

- Naive brute force approach
- Calculate α, 2α, 3α,.....,xα until we get β
- If xα is equal to β, then secret key = x
- takes forever for large values of n

Baby-step giant-step (O(√n))

- Any integer can be written as x = im+j
- In ECC, β = kα, then
  - ➢ β = (im+j)α = imα + jα
  - ➢ or, β - amα = bα
- m=√n and 0 ≤ i,j < m
- Calculate the value of β by calculating jα (baby-steps)and imα(giant-steps).
- Improves upon linear search method
- But consumes more space for storing jα

Pollard's ρ ($O(\sqrt{n})$)

- For any value of α and β, find four integers a, b, A and B such that $a\alpha+b\beta = A\alpha+B\beta$.
- Then for $\beta = k\alpha$,
  - ➢ $a\alpha+b\beta = A\alpha+B\beta$
  - ➢ $a\alpha+bk\alpha = A+Bk\alpha$
  - ➢ $(a+bk)\alpha = (A+Bk)\alpha$
  - ➢ $(a-A)\alpha = (B-b)\alpha$
- Then the value of k can be computed as:-
  - ➢ $k = (a-A)(B-b)^{-1} \mod p$
- Reduces space complexity to O(1)

Challenges

- Increasing computing ability means easier to break cryptosystems (4096-bit RSA broken)
- ECC can be an alternative for future use
- But how to choose good curves with efficient arithmetic?
- General distrust of ECC curves promoted by NSA
- Lots of patents

Questions ?