universität freiburg

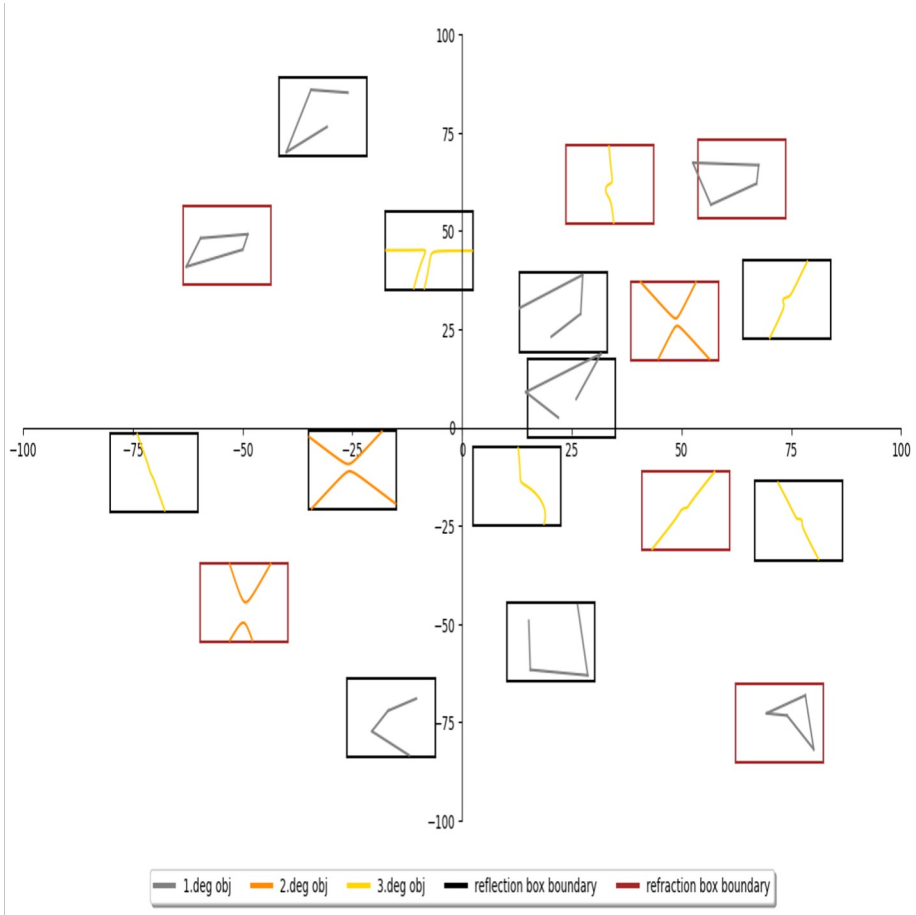
Cryptography based on 2D and 3D Ray Tracing

Sneha Mohanty

universität freiburg

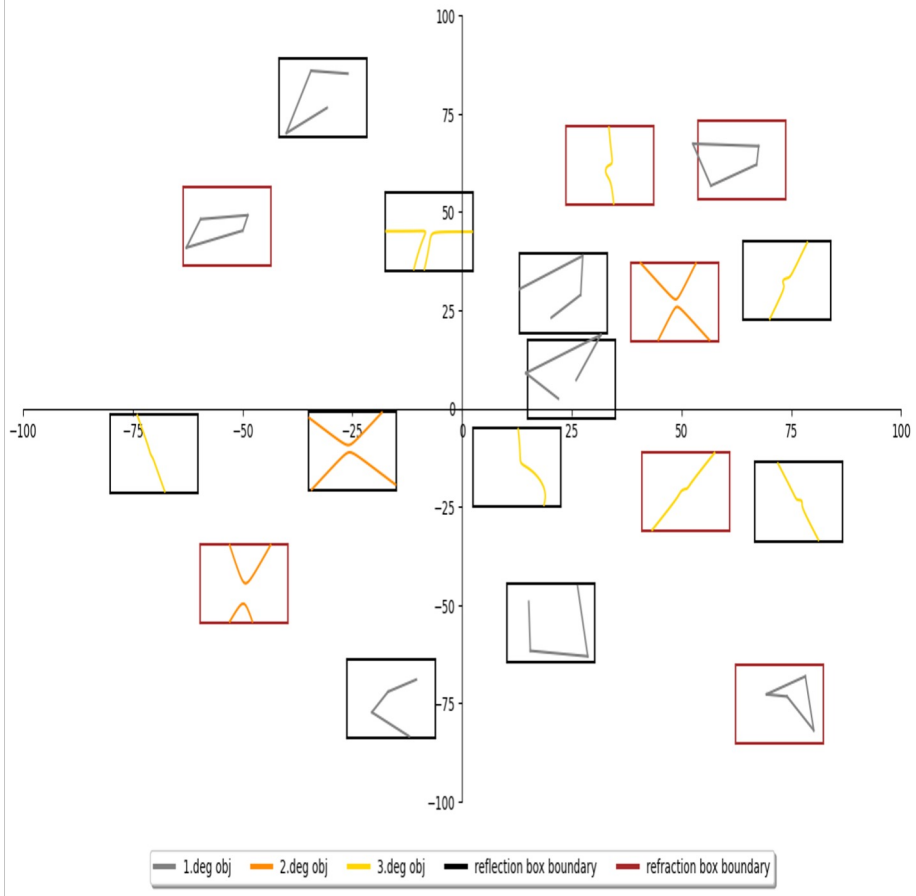
Introduction

- **Completely novel symmetric key cryptographic approach** involving 2D cartesian coordinate system and boolean gates
- The 2D setup consists of **Local bounding boxes (LBBs)** placed at random (x,y)-positions inside the 2D setup, which we call the **Global bounding box (GBB)**
- LBBs are of two types, **Reflection (black boxes)** as well as **Refraction (red boxes)**
- The curves inside the LBBs are **first, second or third degree polynomials**, which are rotated and translated from the origin
- Apart from these, we also use boolean logic gates, such as; **XOR, NOT-Shift** as well as **Permutation** in the cryptographic scheme, whose priority is decided pseudo-randomly for each Local Bounding Box in the given 2D setup

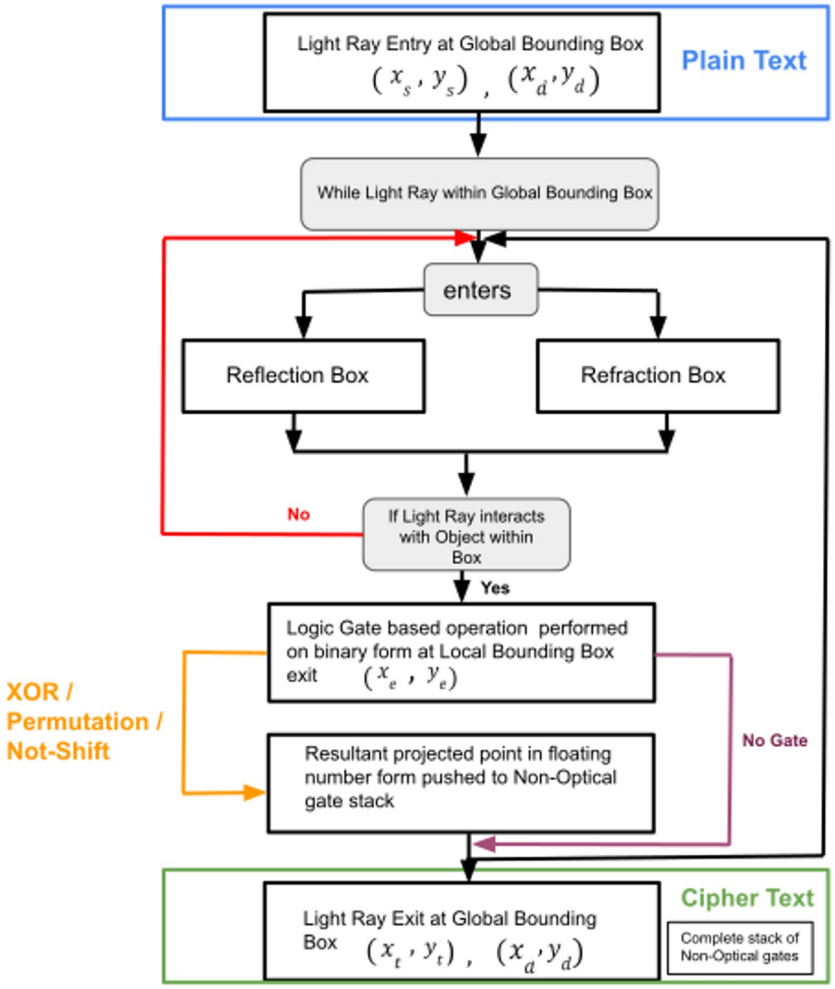


Components of the Cryptographic system

- **Plaintext** includes starting x and y coordinates of the Global bounding box (GBB) and the initial direction of light ray
- **Ciphertext** includes exit direction of light ray, its exit point coordinates as well as the stack of projection boxes, or Non-optical gate boxes.
- **Key** includes 3 elements: Global bounding box, Cryptographic element as well as the Local bounding box (LBB) parameters



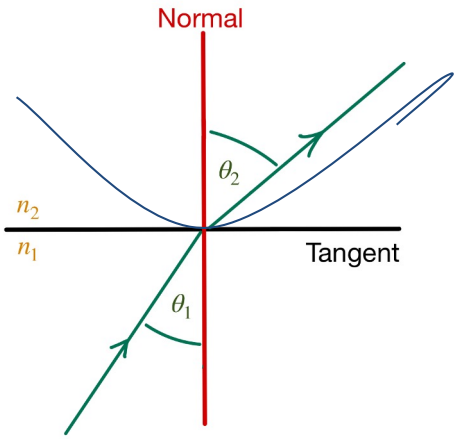
Overview

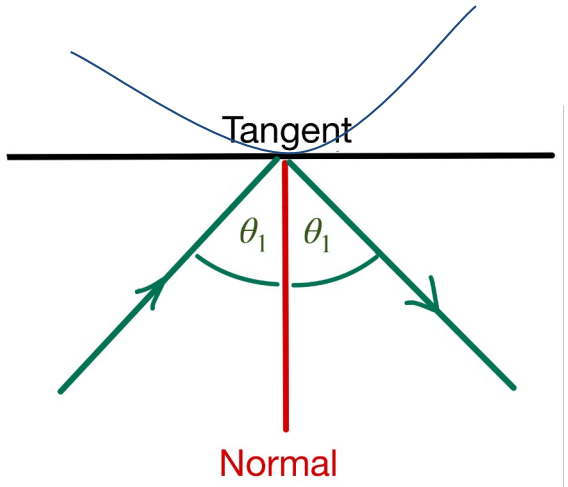

Light Ray Entry at Global Bounding Box
(x_s, y_s) , (x_d, y_d)
Plain Text
While Light Ray within Global Bounding Box
enters
Reflection Box
Refraction Box
No
If Light Ray interacts with Object within Box
Yes
Logic Gate based operation performed on binary form at Local Bounding Box exit (x_e, y_e)
XOR / Permutation / Not-Shift
Resultant projected point in floating number form pushed to Non-Optical gate stack
No Gate
Light Ray Exit at Global Bounding Box (x_t, y_t) , (x_d, y_d)
Cipher Text
Complete stack of Non-Optical gates

, ,

Optical gate components : Light ray, reflection and refraction

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} + \lambda \begin{pmatrix} x_d \\ y_d \end{pmatrix}$$





$$y = m_t \cdot x + c_t$$

$$y_p = m_t \cdot x_p + c_t$$

$$c_t = y_p - m_t \cdot x_p$$

$$y = m_n \cdot x + c_n$$

$$y_p = m_n \cdot x_p + c_n$$

$$c_n = y_p - m_n \cdot x_p$$

Snell's Law

$$n_1 \cdot \sin(\theta_1) = n_2 \cdot \sin(\theta_2)$$

Optical gates: 1st, 2nd and 3rd degree polynomials

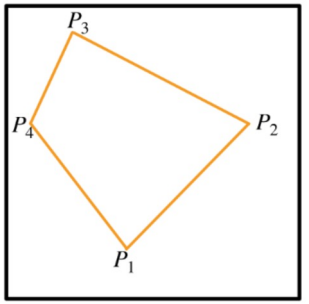
$$Dx + Ey + F = 0$$

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

$$Gx^3 + Hy^3 + Kx^2y + Lxy^2 + Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$$

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}$$



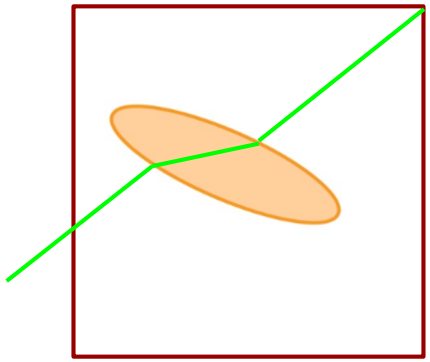
Optical gates: Intersection calculation with curves

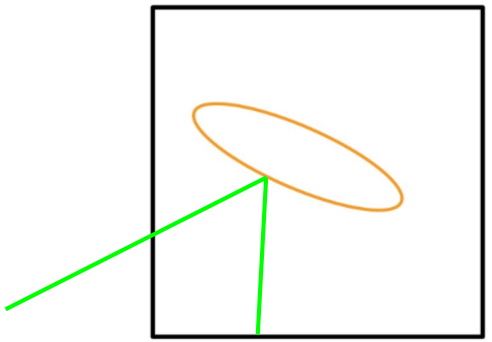
$$a\lambda^2 + b\lambda + c = 0$$

$$\lambda_{1/2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

discriminant	number of real roots
$\Delta > 0$	2
$\Delta = 0$	1
$\Delta < 0$	0

$$\Delta = b^2 - 4ac$$





Optical gates: Intersection calculation with curves

$$a\lambda^3 + b\lambda^2 + c\lambda + d = 0$$

$$a_3\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0 = 0$$

Cardano's equations

$$\mathcal{Q} \equiv \frac{3a_1 - a_2^2}{9}$$

$$\mathcal{R} \equiv \frac{9a_2a_1 - 27a_0 - 2a_2^3}{54}$$

$$\mathcal{D} \equiv \mathcal{Q}^3 + \mathcal{R}^2$$

$$\mathcal{S} \equiv \sqrt[3]{\mathcal{R} + \sqrt{\mathcal{D}}}$$

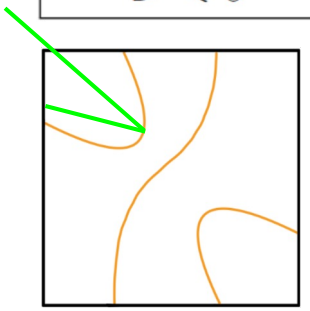
$$\mathcal{T} \equiv \sqrt[3]{\mathcal{R} - \sqrt{\mathcal{D}}}$$

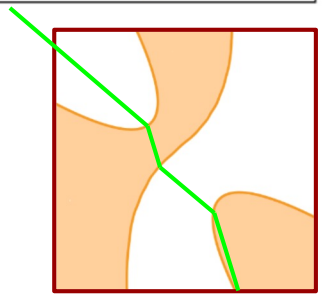
$$\lambda_1 = -\frac{1}{3}a_2 + (\mathcal{S} + \mathcal{T})$$

$$\lambda_2 = -\frac{1}{3}a_2 - \frac{1}{2}(\mathcal{S} + \mathcal{T}) + \frac{1}{2}i\sqrt{3}(\mathcal{S} - \mathcal{T})$$

$$\lambda_3 = -\frac{1}{3}a_2 - \frac{1}{2}(\mathcal{S} + \mathcal{T}) - \frac{1}{2}i\sqrt{3}(\mathcal{S} - \mathcal{T})$$

Discriminant	Roots
$\mathcal{D} > 0$	1 real, 2 complex
$\mathcal{D} = 0$	3 real, at least 2 are equal
$\mathcal{D} < 0$	3 real, all unequal



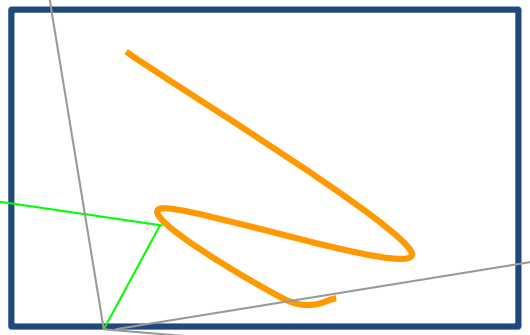


Non-Optical gates: NOT-Shift, XOR and Permutation

$(0110101.01101100110000001001, 1110110.11011001011111011 1)_2$

$= (53.424813270568847656 25,\ 118.849575042724609375)_{10}$

xor with 1110001.11100011100111001010



$= (1100000.10001111010111001100\ ,\ 10101.001110101110000011)_2$

$(96.56000900268554685,\ 21.229991912841796875)_{10}$

$(68.56, 25.23)_{10}$

$= (1000100.10001111010111000011, 11001.001110101110000101)_2$

$(1101110.00011100001010001111, 10001.101100010100011110)_2$

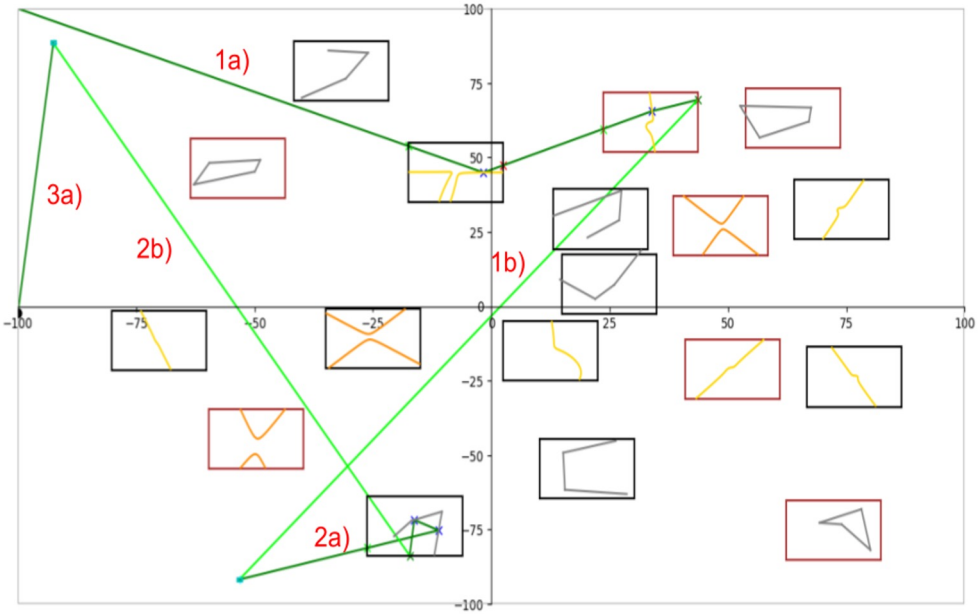
$= (110.10999965667724609375, 17.69249725341796875)_{10}$

Encryption

- Possible objects / curves:
  1st, 2nd and 3rd degree polynomials in 2D
  Polynomials rotated and translated from origin

- Boolean logic gates
  XOR, NOT-shift, Permutation

  We currently use 512 bits (in decimal) for encryption of the XOR gate (both sides of the decimal point), permutations as well as the NOT-shift gate. This can be customized to be increased/decreased, and int part is currently set to $\ln_2(\ size\ of\ \ GBB)$



Non-Optical gates: Validation of Projected point

- A projected point is considered invalid in the cases where :

  - It is outside the Global bounding box

  - It is potentially enclosed within another object (in current improvement, we don’t generate enclosed 2nd and 1st degree objects in the reflection case)

  - It is within the same Local bounding box from which it was projected

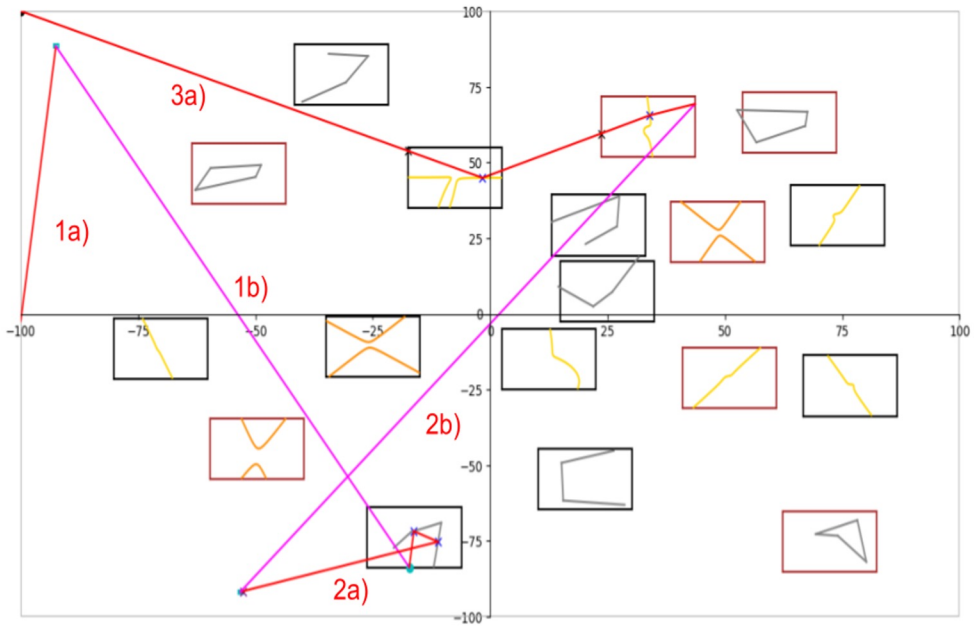
  - It is within another projected point

Key

```
[[-100, -100, 200.0, 200.0, \\ Global Bounding Box position and size
["00011010", \\ XOR constant for integer part
"0101001110101110100011011100001010110110101111100101010110101110001001110100110010110110100000000111000101010110011110010110100111101000111111110110111
0000001110011110001111011000011111101110101000111101000011100010101000001001000001011101001100111111111011100010011111011001101101000111000010100011000
0111001111111100001111111111111001011101000000011000001111101001011100110001011111011001000101000101010111110000010001011000000000101000111100000010010101
000011110110000110000100001101111011100111101100101001110001", \\ XOR constant for decimal part
 {"8": [7, 2, 4, 0, 6, 5, 3, 1], \\ Permutation mapping for integer part
 "512": [363, 258, 505,..., 312]}, \\ Permutation mapping for decimal part
{"8": [3, 7, 1, 6, 2, 5, 4, 0], \\ Inverse permutation mapping for integer part
"512": [223, 428, 416,..., 22]}, \\ Inverse permutation mapping for decimal part
 5, 1, "934dd5e3-bffb-418d-9afc-01304b6cbe9a"]], \\ Not-shift number of bit-shifts, direction of bit-shift, 0-> right shift, 1 -> left shift, Unique
identifier of the rendered Non-Optical gate position (if we want to have one at the beginning of the encryption, can also be disabled)
[197, 8, 512], \\ seed of the 2D setup, length of the integer part, length of the decimal part
[[2, "reflection", ["30", "30"], ["0", "-1.4368230289832744084321802802151069045066833496093 75", "-34.763256024470919669511204119771718978881835937 5",
"35.085941416030038908502319827675819396972656 25", "30", "30.568705665088884870783658698201179504394531 25",
"61.055673126880321888165781274437904357910156 25", "63.737198720821396591418306343257427215576171875",
"0.595213569368294503902916403603740036487579345703125", "1.000299999999999966959762787155341356992721557617 1875",
"1.520000000000000177635683940025046467781066894531 25", "01111011",
"1000011100100110110011111010100110001011001000111110000111101000110011011111110101110000100111110100101000000000101001000100100000110000111010001100 11
1110100100010110101010001110110000101001001010001111100010100000111100111011111101111000011000110010001011101011010011001101110111110010001111001100 10
1100111110000011001001111010101011001011011110001000110101010000100111001001100110000111110000101111100000000100101000100111100100101110111010101011011
011101111101111101100111010101001100010000100110101101100100", \\ 2nd degree reflection object, its local bounding box size, its object parameters
{"8": [1, 0, 7, 3, 4, 6, 2, 5],
 "512": [12, 315, 104,...,369]},
 {"8": [1, 0, 6, 3, 4, 7, 5, 2],
"512": [418, 445, 412,...,387]}, "d9aa7875-7c97-4a6c-b6c3-fe6112ea0005", 5, 1]], \\ similar parameters for the 2nd degree curve as in the case shown
above
[3, "reflection", ["30", "30"], ... ]]]] \\ continuation of the key with the next object and so on..
```

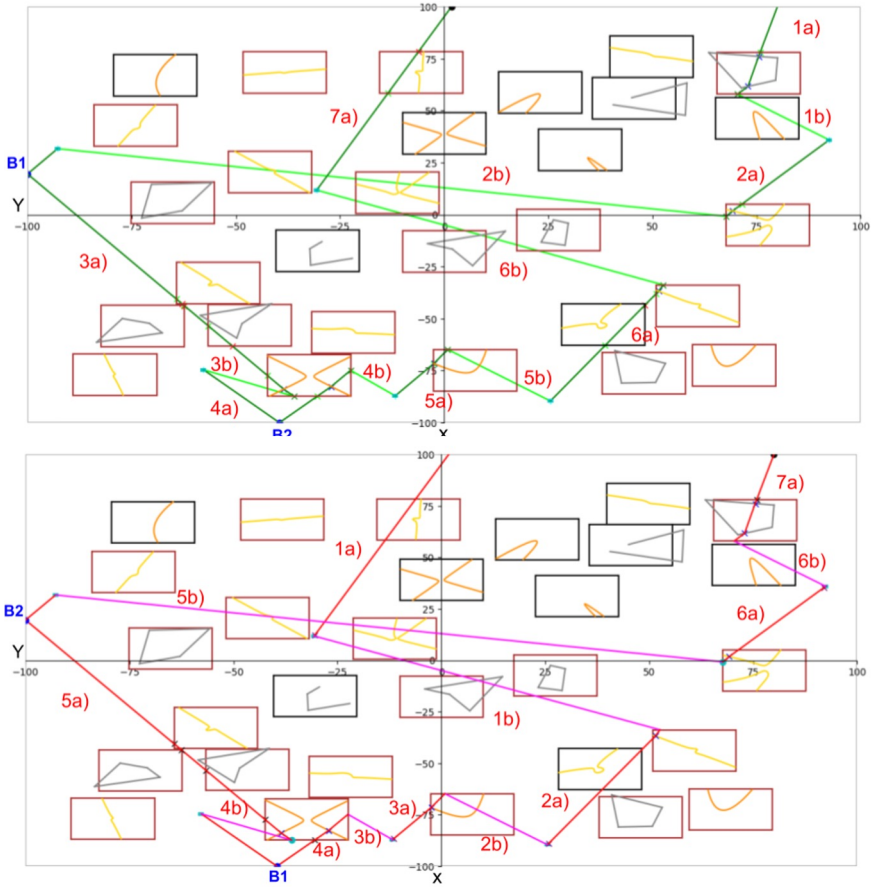
Decryption

- Reverse steps of encryption to obtain the initial point of entry into Global bounding box (x,y coordinates and thereby, the message Plaintext and the initial direction of light ray)



Further Improvements to Encryption-Decryption

- Changes to the key : add bouncing mechanism for the light ray, angle restrictions at point of entry and enforce density ranges (optimal global bounding box to local bounding box ratios)
- Ciphertext morphing - permutate the ciphertext stack, interweave its components and perform other complicated operations to it until we get an output of form (c1,c2) instead of returning the original stack of non-optical gate boxes
- Optimize the components of the key by removing explicit information about the rotation and translation operations, but by making these implicit
- Using mpmath to ensure arbitrary precision and creating helper crypto functions to add sufficient buffer to prevent any potential precision loss during encryption-decryption



Attacks on Cryptosystem

- The attacks on our cryptosystem are broken down into two parts :
  - Attacks on the Optical gates
  - Attacks on Non-optical gates

- If we start by the Overall attacks on the system, then we assume the Global bounding box to be a black box, inside which all the objects as well the subsequent projections of light rays are unknown

- If we start by the Box-by-Box attacks, then we assume that we know the positions of the Local bounding box inside the 2D setup but we don't know the polynomial curves as well as the boolean gates that are applied for the light ray projections

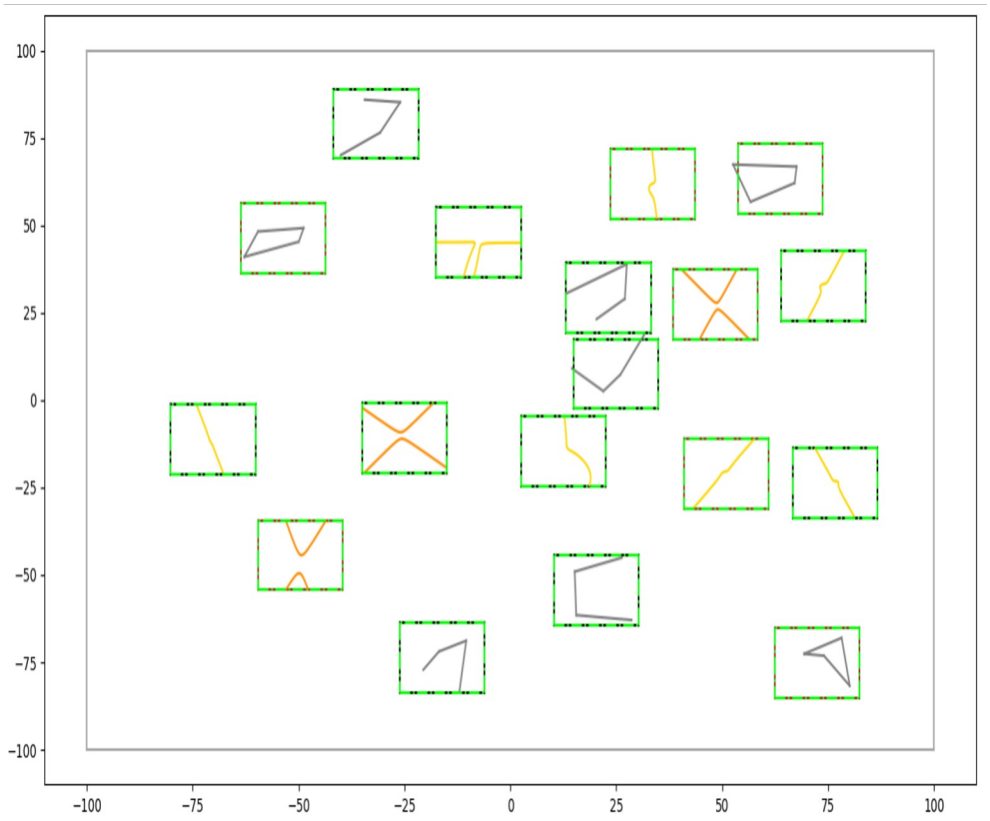
Overall attack

- The overall attack consists of first, launching a grid based search, to locate each curve that intersects a grid line (when it performs either reflection or refraction with it) and thereafter enclosing it with a Local bounding box

- After the positions of the Local bounding boxes are approximated, the Box-by-box attack is then launched on the ‘discovered’ Local bounding boxes, until all the polynomial curves as well as their corresponding boolean-gate related data are attempted to be found.

Overall attack - Grid search

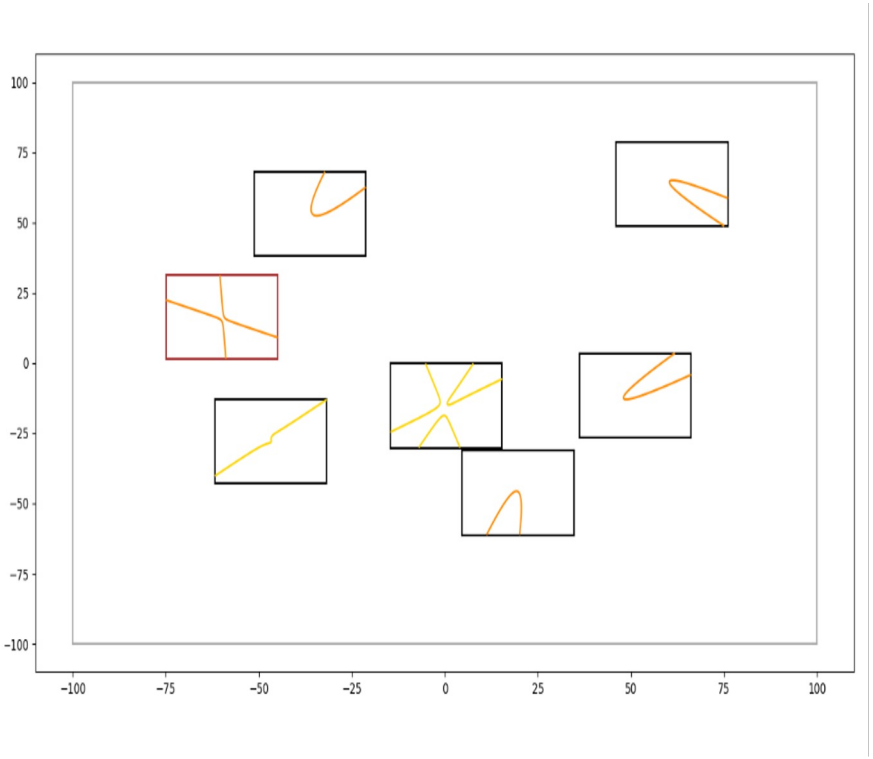
- A sample output for the grid search to locate the Local Bounding Boxes encapsulating objects in a 2D setup is shown here -



Overall attack - Grid search - sample 2

- A sample output for the grid search to locate the Local Bounding Boxes encapsulating objects in a different 2D setup is shown here

- As can be observed, in the case of this key, the Grid search followed by the Clustering algorithm could not compute the locations of the Local bounding boxes correctly

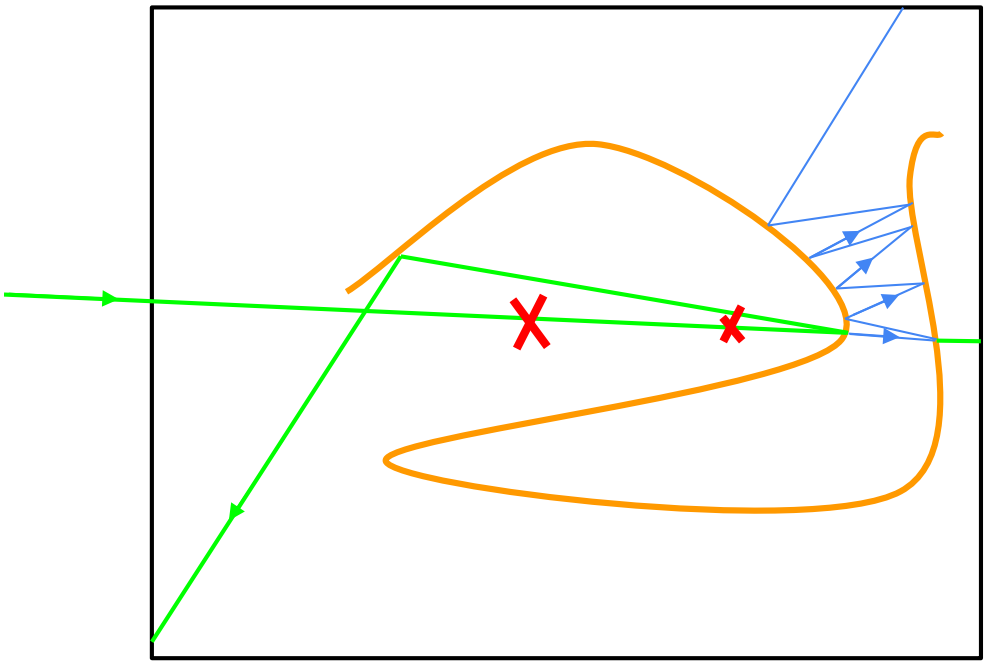


Box-by-Box attack

- The box-by-box attack is implemented by
  - data points gathering using Binary Search and then followed by the PLU technique to discover the polynomial curve inside the Local bounding box
  - followed by using multiple random samples to gather the bit-by-bit transformation information and therefore the XOR constant, Permutations as well as the NOT-shift positions and direction of bit shift for the specific Local bounding box

- Given are the unique box id, ground truth (position of exit of the light ray at Local bounding box), the numbers mapping to each Local bounding box for each of the boolean gates, such as 1,2 or 3 as well as the projected point after the transformation is performed

Box-by-Box attack - Binary Search



- To find data points on the polynomial curve, we compare light rays along the trajectory of the original light ray to find the position, where the output (direction) first changes.
- This indicates than an interaction with the curve has happened.
- We start the light rays according to a binary search along the original light ray by halving the interval that we search in each iteration.
- Each light ray gives us one data point on the curve. We use many such light rays, hitting the curve from many different directions to find more data points and then use approximation methods such as the PLU approximation method to determine the curve using these data points.

Box-by-Box attack - Approximation

Linear Objects :

- For approximating linear objects, we use the property that at least two data points are needed in order to draw a line through it. However, since our linear objects consist of multiple sides and we don't know the corresponding side of each data point, we use at least three data points per side to confirm that they belong to that particular side of the polygon/open-sided linear object.

- We start by gathering all data point pairs and their corresponding line equations. We narrow this search down to the line equations that cover more than two data points. All the remaining data point pairs then get assigned to their corresponding line equations.

- To find the corners, we first find the most distant data point from each of the other line equations to achieve better precision when finding the corners. We then calculate the intersection of all valid lines. By solving these, we retrieve a set of possible corner point coordinates which include the actual corners and false or redundant corners. Since we only allow simple polygons (with the sides not crossing over each other), we can group these lines as adjacent and opposite ones. The closer two opposing lines are to being parallel, the further away their intersection, in this case the more likely it will be a redundant corner. Therefore in some cases redundant corners can be discarded by introducing a bound. However, this is not trivial since it is not clear how large the bound could be, as parts of the polygon are allowed to be outside the Local bounding box.

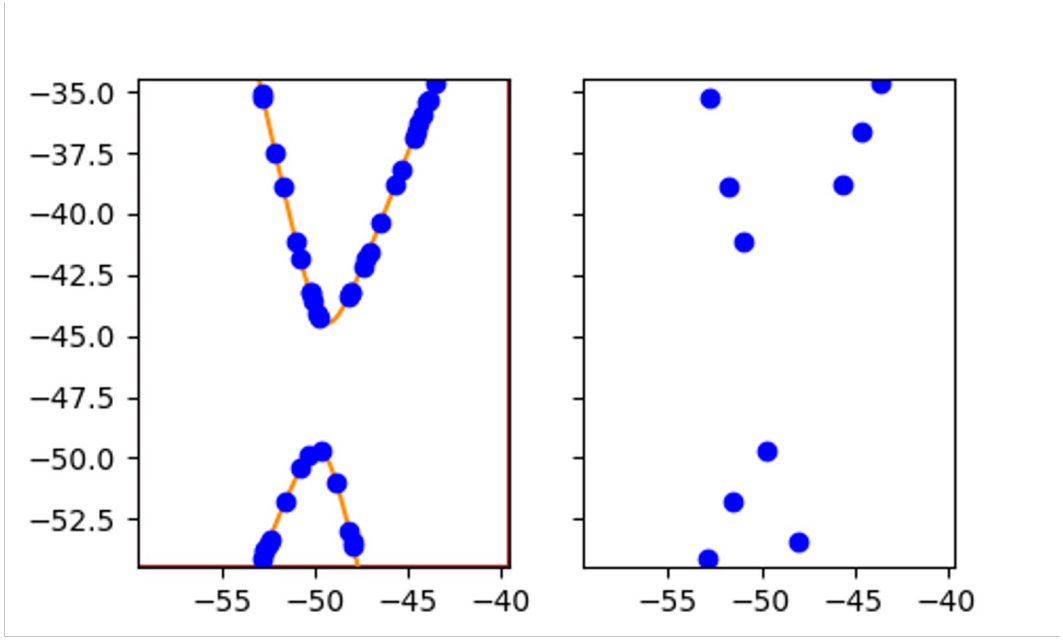
- Also, sometimes there is an open side to the polygon, we will not be able to find data points on the open side, since there can not be data points on the missing edge. Therefore we order the already found 'actual' corners in a chain to determine where the open side is. Sometimes, if one or more of the sides of the polygon does not get hit by sufficient number of light rays, because of its close proximity to another Local bounding box in the 2D setup, then we miss out on computing these edges of the polygon altogether.

Non-Linear Objects :

- For a system of Non-linear equations, we first compute the Jacobian matrix, J of the system, use PLU decomposition of the Jacobian to solve the linearized system and finally use the modified Newton method iteratively until convergence.

Box-by-Box attack - Sample Outputs

- Given 2D setup, attack on a sample box original vs approximated polynomial curve



Improved resilience against standard attacks

- The additional measures such as the bouncing mechanism as well as the ciphertext morphing add a much higher resilience to the key against standard attacks, such as; known Plaintext attacks, Ciphertext attacks etc. Covered in more detail in submitted Paper.

Conclusion and Future Work

- Novel symmetric-key cryptographic approach
- Extended also to 3D
- Could probably use the boolean gate operations on the dx, dy components rather than the x, y components on the point of exit of Local bounding boxes, thus bending the light ray rather than projecting it

References

1. Mohanty, S., Peairs, E., Schindelhauer, C.: Cryptography Based on 2D Ray Tracing (EasyChair, 2024),https://easychair.org/publications/preprint/h8fS ,Preprint no. 15449

1. Reif, J.H., Tygar, J.D., Yoshida, A.: Computability and complexity of ray tracing. Discrete & Computational Geometry 11, 265–288 (1994)

universität freiburg

Cryptography based on 2D and 3D Ray Tracing

Sneha Mohanty

universität freiburg