Guideline for Designing the GUI in JCrypTool Plug-ins

Version 1.0
23. May 2012

Contents

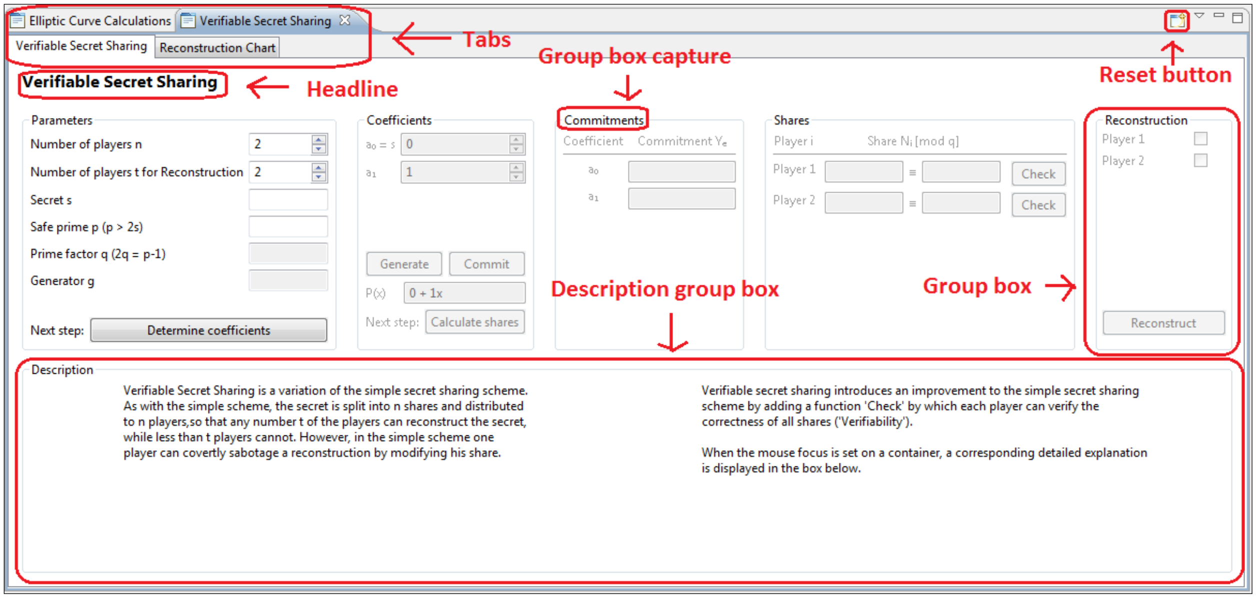
Goal of this Guideline

This guideline aims to support developers designing visualization, game and analysis plug-ins in JCrypTool. It contains rules and recommendations to improve JCrypTool's ease of handling and to standardize basic design principles.

Quick Start/Summary

This paragraph gives a complete summary of the design principles in a nutshell and to allow for a quick start.

The picture labels some elements in order to avoid confusions about the notations.



Verifiable Secret Sharing

Parameters

Number of players n: 2

Number of players t for Reconstruction: 2

Secret s

Safe prime p (p > 2s)

Prime factor q (2q = p-1)

Generator g

Next step: Determine coefficients

Coefficients

$a_0 = s$: 0

a_1: 1

P(x): 0 + 1x

Next step: Calculate shares

Commitments

Coefficient	Commitment Y_e
a_0	
a_1	

Shares

Player i	Share N_i [mod q]		
Player 1		≡	Check
Player 2		≡	Check

Reconstruction

Player 1 □

Player 2 □

Description

Verifiable Secret Sharing is a variation of the simple secret sharing scheme. As with the simple scheme, the secret is split into n shares and distributed to n players,so that any number t of the players can reconstruct the secret, while less than t players cannot. However, in the simple scheme one player can covertly sabotage a reconstruction by modifying his share.

Verifiable secret sharing introduces an improvement to the simple secret sharing scheme by adding a function 'Check' by which each player can verify the correctness of all shares ('Verifiability').

When the mouse focus is set on a container, a corresponding detailed explanation is displayed in the box below.

Figure 1: VSS overview of elements

1. **Required Objects**
   - A Headline
   - At least one group box for user input
   - One group box for descriptions
   - A reset button

2. **Layout Recommendations**
   - Don't use additional windows. Try to place the content in one window
   - Add descriptions and instructions in a separate group box for information. Use pop-ups for errors only
   - If tab controls are used the different tab windows should not depend on each other. Use tabs only if you implement independent parts or to show the final results

- Each explanation or instruction is within the description group box. The content may change, depending on the situations' context
- Optimize the layout to a 1680x1050 resolution or smaller
- Separate the user interaction in multiple steps
- Each step consists of a separate group box
- Sort the steps horizontally or vertically
- It must be obvious which user interaction is required in each situation
- The current step must not influence previous steps
- Important values (e.g. user input) have to be visible at each time
- Separate user input and computed output visually
- Insert the content of the description group box dynamically, depending on the position of the cursor

3. Mandatory Design Principles

- Buttons: Each button has an inscription which indicates what action will happen. The grammar of the inscription is like in a normal English sentence, not like in captions. Buttons don't have a label.
  Bad examples: "Result", "Step 4".
  Good examples: "Compute result", "Reset", "Generate prime numbers".

- Drop down lists: A drop down list has a pre-selection. Use drop down menus for a small amount of values only.

- Labels: Each input and output field has its own label. The label is in the left to the field. Don't use labels to insert whole sentences between control elements. Whole sentences belong to the description group box.

- Text fields: Text fields don't have a default text in it. Limit the length of input values. For data whose length can be longer than the regular length of the text field, use text areas.

- Radio buttons: Radio buttons are the right choice if its selection influences the whole layout (e.g. "Visual mode", "Text mode") or to separate basic settings (e.g. "Encryption", "Decryption").

- Group boxes: Each group box has a caption (title). The caption describes its content and not their order! You can create group boxes within a group box but only in the first iteration.

- Text: The fonts are defined in "import.org.jcryptool.core.util.fonts.FontService". Use the color red to highlight usage failures or mismatches and use color green to highlight successful verifications. Formulas should be written in a mathematical representation. But at least the notation of formulas has to be consistent in the whole plug-in. The "messages.properties" files support UTF-8 only. If you insert formulas, use pictures or don't use mathematical representations of formulas in the whole plug-in. The character encoding is UTF-8 and the language of the plug-in has to be German or English. If you are able to write the text in both languages, please do so.

- Pop-ups: Use pop-ups for errors only and if the plug-in cannot work without correcting the problem. All problems have to be located and described in detail. Implement different proceeding choices like automatic correction, setting the values back to default or manual correction. Don’t change the input values automatically without user confirmation. The curser has to jump to the first incorrect field. If there is a high amount of input fields, highlight the incorrect ones.

How to Start Designing the GUI

The first challenge for the designer is to draft the basic structure. This depends mostly on what kind your plug-in is. Do you want to implement a two-party protocol, a cipher or an algorithm? All those categories have different requirements for the GUI. But in most cases you don't have to start from scratch. Probably there are graphics or even animations of your or a similar subject. Those can be very helpful to get an idea how to design the basic structure. Maybe your plug-in is similar to an existing one. Then, you can probably adopt parts of its layout.

After you have chosen a basic layout, analyze your own requirements. Make a list of all input, output and control elements, estimate the required space for them and develop prototypes. Maybe you have to change your basic layout because the content doesn't fit in it. Especially content which size depends on user input is critical. Position the elements in the basic layout.

As for example: The programmer wants to implement a man-in-the-middle attack against a two-party protocol. How does a regular visualization of a cryptographic protocol looks like?

Usually cryptographic protocols are visualized in a diagram, as for example like this:

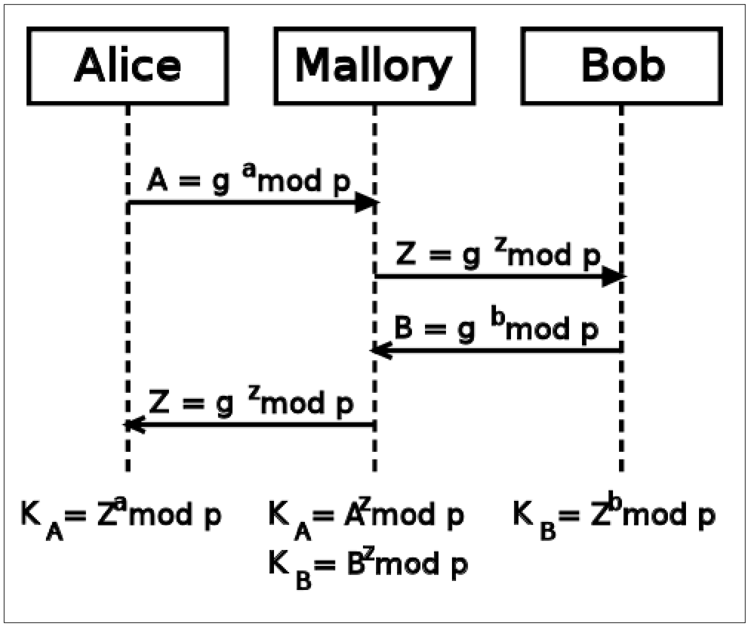


Figure 2: Man-in-the-middle attack against DHKE[1]

This structure can be used to design the basic elements.

For example: Implement three columns, one for each party. During the steps the user changes for which role he takes an action.

Arrows clarify the data transmission to the next party and thus lead the user to the next step.

* Source: http://upload.wikimedia.org/wikipedia/commons/5/50/Man-in-the-middle_attack_of_Diffie-Hellman_key_agreement.svg

Input values may need a lot of space. As the layout is based on small columns this may be a problem. Either restrict the input length or use a text area with scroll bars.

How can we position the elements in an appropriate way? At each step, first show the submitted values than implement the elements for the current step and at last a button for data transmission.

Unfortunately this would work for a protocol with only a few steps. A more complex protocol would cause a great amount of input and output fields. There is no way to keep track if the input and output fields are distributed in three columns. The designer has to outsource these fields. Maybe it is advisable to use a fourth column in which all input and output values are concentrated. The difficulty here is to create the logical connections with the other three columns.

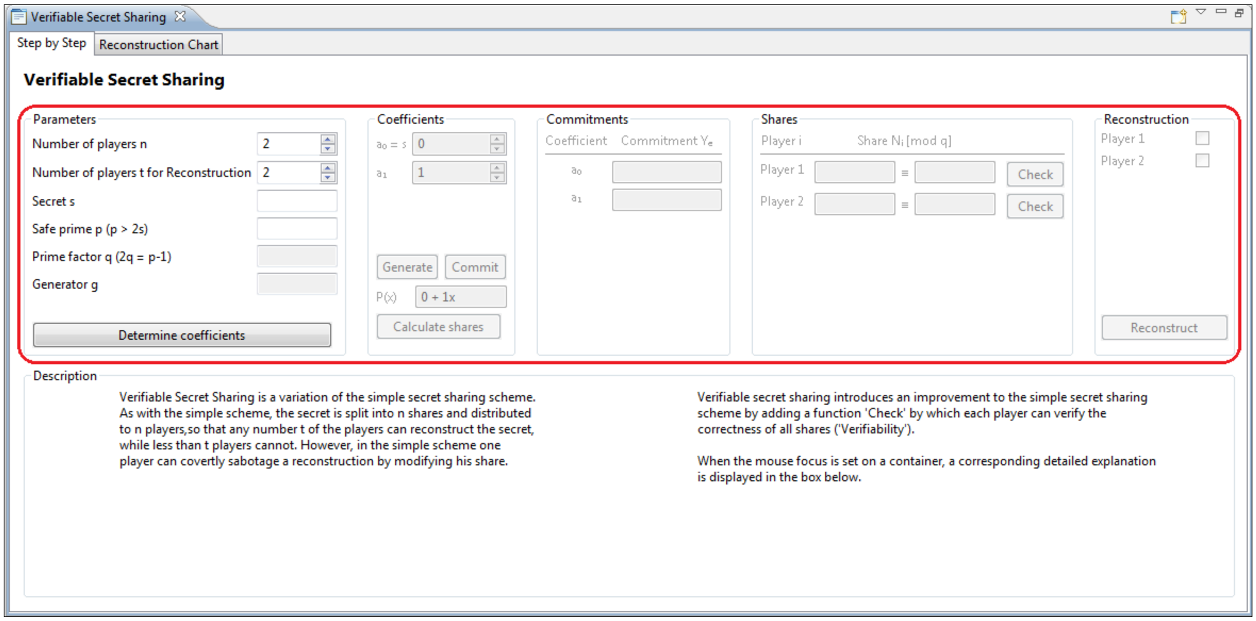
This example illustrates that there are many ways to design a GUI, but it is not easy to design a good one. Try to determine possible problems early and test the usability with people who are not involved in developing the plug-in.

Examples

In the following this document shows two examples for implementing the design. Finally it point out some examples how to implement single elements.

The Verifiable-Secret-Sharing (VSS) Plug-in

One example for a good design is the VSS plug-in. The VSS plug-in consists of multiple steps which lead the user from basic parameters to a final reconstruction graph. VSS is divided in two main areas, one area for the steps and one for the descriptions. The designer chose to use a horizontal layout which is advisable if one implements many steps and need a lot of space for variables.



Number of players n: 2

Number of players t for Reconstruction: 2

Secret s:

Safe prime p (p > 2s):

Prime factor q (2q = p-1):

Generator g:

$a_0 = s$: 0

a_1: 1

P(x): 0 + 1x

Verifiable Secret Sharing is a variation of the simple secret sharing scheme. As with the simple scheme, the secret is split into n shares and distributed to n players,so that any number t of the players can reconstruct the secret, while less than t players cannot. However, in the simple scheme one player can covertly sabotage a reconstruction by modifying his share.

Verifiable secret sharing introduces an improvement to the simple secret sharing scheme by adding a function 'Check' by which each player can verify the correctness of all shares ('Verifiability').

When the mouse focus is set on a container, a corresponding detailed explanation is displayed in the box below.

Figure 3: VSS complete view

The plug-in is divided in two tabs. The visualization is outsourced to a second tab because it can only be performed in the last step. This separates the final result from the previous steps.

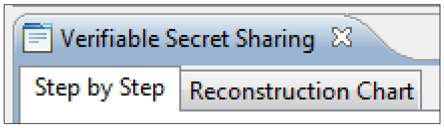
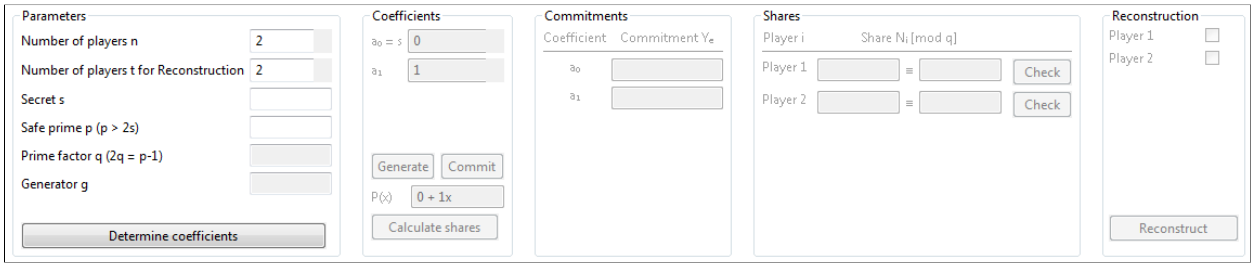


Figure 4: VSS tabs

The user interaction is divided in five separate steps. Each step is in a separate group box consisting of control objects. The “Commitments” group box (which only shows calculated output) and the “Shares” group box (which validates the shares) are optional so they don’t need a “Next Step Button”. The usage is straight forward: User interaction is limited to the actual step, starting from the upper left with basic parameters and ending on the reconstruction button in the lower right. To each time it is obvious how to interact with the GUI. The colors red and green are used to highlight a successful --- or a failed --- verification of values in the Shares group box, while clicking on the “Check” button.



Number of players n: 2

Number of players t for Reconstruction: 2

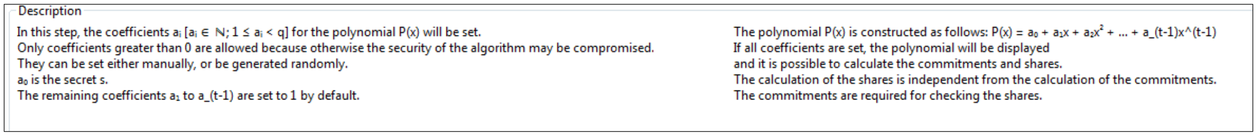
$a_0 = s$: 0

a_1: 1

P(x): 0 + 1x

Figure 5: VSS control objects

Underneath the main area there is a single description group box which concentrates all textual information. At the program start the description group box consists of an introduction of what the plug-in is about. Its content is context sensitive, so it changes during the steps and guides the user through them. Besides this, the description group box shows information about the variables and makes the computation transparent due to information about the computational background. The description group box is separated into two columns. This allows changing only parts of its content and offers to separate different kinds of information.



Description

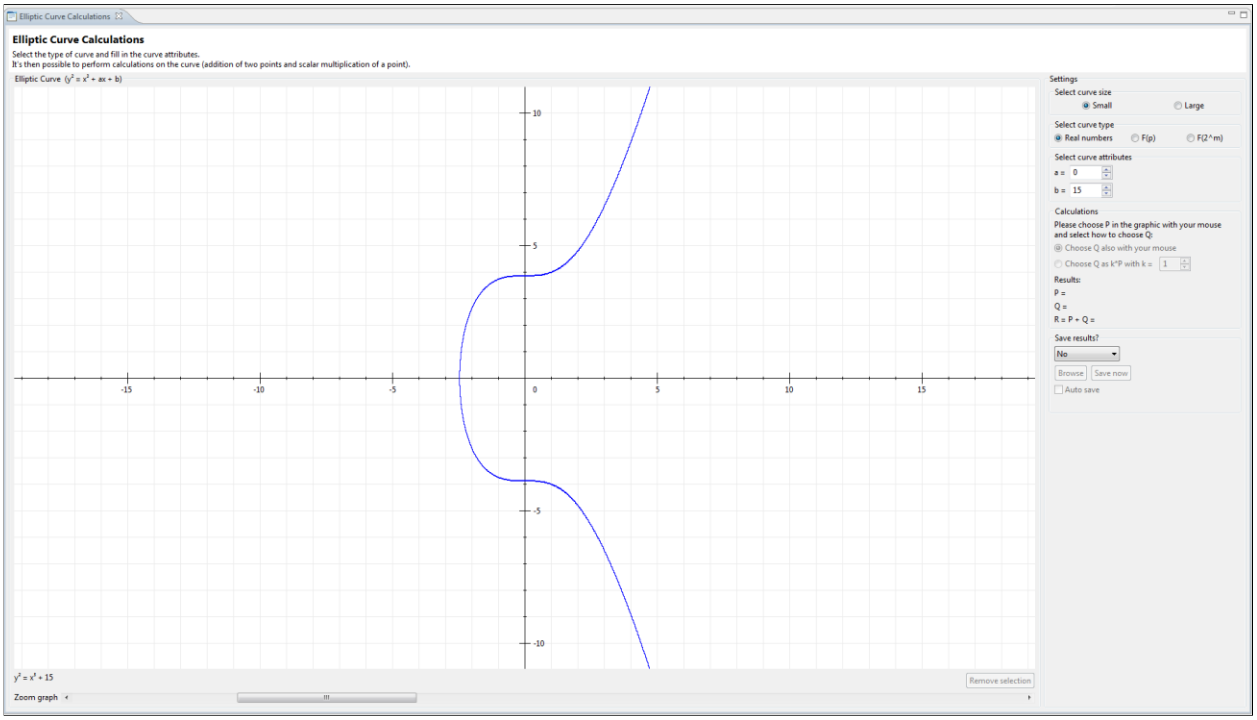
In this step, the coefficients a_i [$a_i \in \mathbb{N}$; $1 \leq a_i < q$] for the polynomial P(x) will be set.
Only coefficients greater than 0 are allowed because otherwise the security of the algorithm may be compromised.
They can be set either manually, or be generated randomly.
a_0 is the secret s.
The remaining coefficients a_1 to a_(t-1) are set to 1 by default.

The polynomial P(x) is constructed as follows: $P(x) = a_0 + a_1x + a_2x^2 + \dots +$ a_(t-1)x^(t-1)
If all coefficients are set, the polynomial will be displayed
and it is possible to calculate the commitments and shares.
The calculation of the shares is independent from the calculation of the commitments.
The commitments are required for checking the shares.

Figure 6: VSS description group box

This plug-in shows how to implement multi-step visualizations consisting of a high didactical aspiration. It may be improved by showing warnings in within the description group box instead of in pop-ups and by offering the possibility to use large parameters with being able to display them completely.

The Elliptic Curve (ECC) Plug-in



Elliptic Curve Calculations

Select the type of curve and fill in the curve attributes.
It's then possible to perform calculations on the curve (addition of two points and scalar multiplication of a point).

Elliptic Curve ($y^2 = x^3 + ax + b$)

Settings

Select curve size: ● Small ○ Large

Select curve type: ● Real numbers ○ F(p) ○ F(2^m)

Select curve attributes
a = 0
b = 15

Calculations
Please choose P in the graphic with your mouse and select how to choose Q:
● Choose Q also with your mouse
○ Choose Q as k*P with k = 1

Results:
P =
Q =
R = P + Q =

Save results? No
□ Auto save

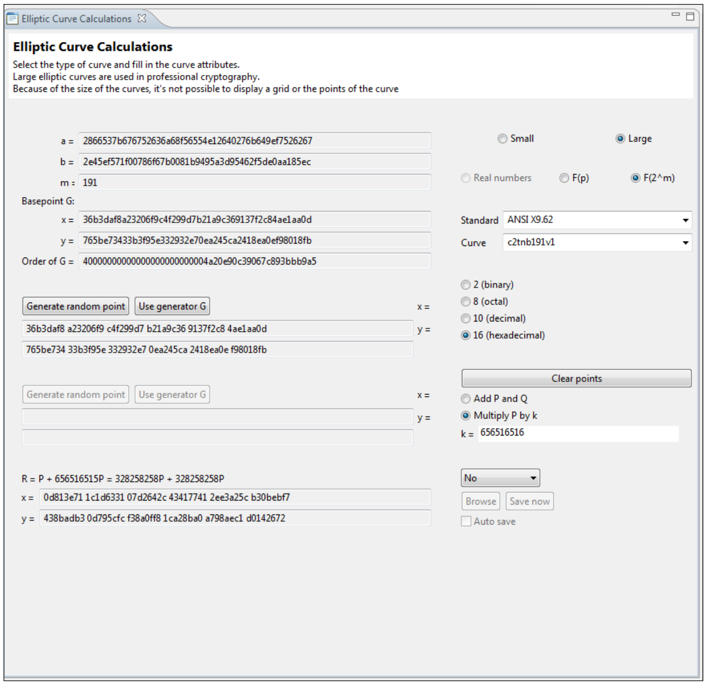
$y^2 = x^3 + 15$

Figure 7: ECC using small numbers

The elliptic curve plug-in has completely different demands as the VSS plug-in. The graphic is not only used for visualization but for user interaction, too. Therefore it is necessary to implement a huge graphic and it is not possible to outsource the graphic to a second tab as in the VSS plug-in. Because the graphic is almost quadratic and most monitors a wider than higher it makes sense to sort the input group boxes **vertically**.

The first settings in the upper right (group box "Settings") can completely change demands to further steps, so the layout has to change. Though during the steps the layout of the graphics and of the following steps may change, previous steps do not. Similar functionalities in different layouts are placed at the same position. This avoids the problem searching settings in different places.

Also there is no redundancy: Optional fields are only visible if they are useful. The plug-in supports a graphical mode for



Elliptic Curve Calculations

Select the type of curve and fill in the curve attributes.
Large elliptic curves are used in professional cryptography.
Because of the size of the curves, it's not possible to display a grid or the points of the curve

a = 2866537b676752636a68f56554e12640276b649ef7526267
b = 2e45ef571f00786f67b0081b9495a3d95462f5de0aa185ec
m = 191
Basepoint G:
x = 36b3daf8a23206f9c4f299d7b21a9c369137f2c84ae1aa0d
y = 765be73433b3f95e332932e70ea245ca2418ea0ef98018fb
Order of G = 40000000000000000000000004a20e90c39067c893bbb9a5

x = 36b3daf8 a23206f9 c4f299d7 b21a9c36 9137f2c8 4ae1aa0d
y = 765be734 33b3f95e 332932e7 0ea245ca 2418ea0e f98018fb

x =
y =

R = P + 656516515P = 328258258P + 328258258P
x = 0d813e71 1c1d6331 07d2642c 43417741 2ee3a25c b30bebf7
y = 438badb3 0d795cfc f38a0ff8 1ca28ba0 a798aec1 d0142672

○ Small ● Large

○ Real numbers ○ F(p) ● F(2^m)

Standard: ANSI X9.62
Curve: c2tnb191v1

○ 2 (binary)
○ 8 (octal)
○ 10 (decimal)
● 16 (hexadecimal)

○ Add P and Q
● Multiply P by k
k = 656516516

No
□ Auto save

Figure 8: ECC using large numbers

small numbers and a text mode for big numbers.

While the graphic mode consists of a huge visualization and small input fields, the text mode doesn't have visualization but huge input fields. This is realized by changing the layout instead of using additional tabs or windows. This increases the usability significantly.

This plug-in implementation shows how to implement a versatile GUI with powerful configuration opportunities using only one tab and no popup windows. This layout may be improved by adding a description group box in order to support curious users.

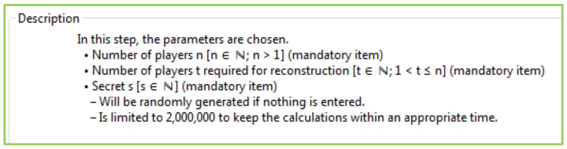
Examples for Single Elements

In the following section there are some examples for good visualization practices.

Pop-ups

The problem with pop-ups is that the usage of JCT is interrupted. This is the reason why pop-ups shouldn't be used for explanation. An additional click is required and the pop-up is blocking the whole window. An attempt was made to let the user configure via a checkbox, weather pop-ups should be shown – but this is still not user-friendly. Pop-ups which are used to show errors are OK because the program can't proceed without correcting the reason for the error. Errors have to be described in detail. It must be obvious which input failure causes the error. If more than one input is incorrect mention all affected fields. When closing the pop-up, the cursor has to jump to the first incorrect input field. Think about highlighting the incorrect values. If possible, implement selections. As for example instead of one "OK" button, use buttons "Generate", "Set to default" and "Correct manually".

Instead of explanation pop-ups please use an explanation text within the description group box as shown in the picture to the right.

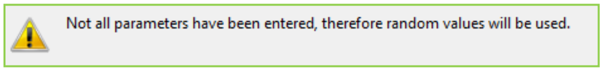


Description

In this step, the parameters are chosen.
- Number of players n [n ∈ ℕ; n > 1] (mandatory item)
- Number of players t required for reconstruction [t ∈ ℕ; 1 < t ≤ n] (mandatory item)
- Secret s [s ∈ ℕ] (mandatory item)
  - Will be randomly generated if nothing is entered.
  - Is limited to 2,000,000 to keep the calculations within an appropriate time.

Figure 9: VSS description group box snippet

A more sophisticated solution to the error pop-up in the right is to generate the values automatically and insert a warning in the description group box that the empty or incorrect field has been corrected.



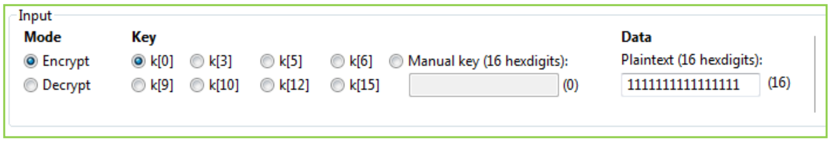
Not all parameters have been entered, therefore random values will be used.

Figure 10: VSS warning popup

Input Fields

Input fields should consist of a label. Additionally, add pre-selections of values if you are using drop-down lists.

The right group box shows a good example: Even without a description it is obvious what to do. Another advantage: Example inputs are given. A lot of users don't care about the input values and want an example they can use for a head-start.



Input

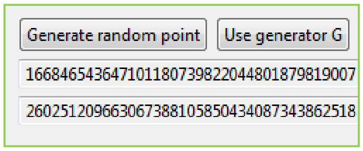
Mode
● Encrypt
○ Decrypt

Key
● k[0] ○ k[3] ○ k[5] ○ k[6] ○ Manual key (16 hexdigits):
○ k[9] ○ k[10] ○ k[12] ○ k[15] (0)

Data
Plaintext (16 hexdigits):
1111111111111111 (16)

Figure 11: ECC input group box

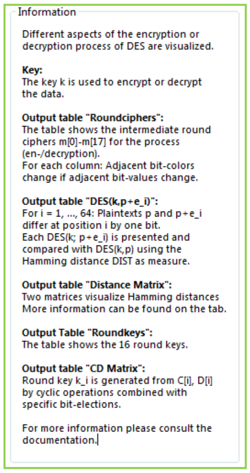
Another possibility to help the user entering valid data is to implement "Generate" buttons. Especially for big input values it is much easier to just click "Generate" than to insert the input manually.



166846543647101180739822044801879819007
260251209663067388105850434087343862518

Figure 2: ECC using large number snippet

The Description Group Box



Information

Different aspects of the encryption or decryption process of DES are visualized.

Key:
The key k is used to encrypt or decrypt the data.

Output table "Roundciphers":
The table shows the intermediate round ciphers m[0]-m[17] for the process (en-/decryption).
For each column: Adjacent bit-colors change if adjacent bit-values change.

Output table "DES(k,p+e_i)":
For i = 1, ..., 64: Plaintexts p and p+e_i differ at position i by one bit.
Each DES(k; p+e_i) is presented and compared with DES(k,p) using the Hamming distance DIST as measure.

Output table "Distance Matrix":
Two matrices visualize Hamming distances
More information can be found on the tab.

Output Table "Roundkeys":
The table shows the 16 round keys.

Output table "CD Matrix":
Round key k_i is generated from C[i], D[i] by cyclic operations combined with specific bit-elections.

For more information please consult the documentation.

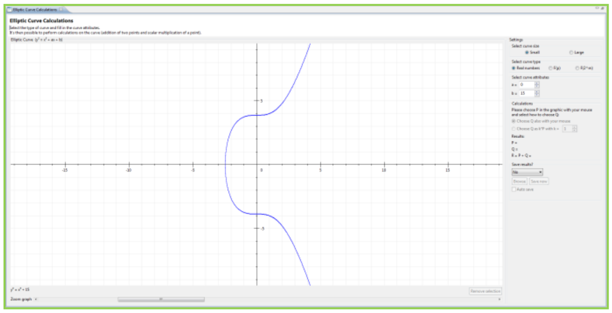
Figure 13: DPA parameters group box

The textual description of the plug-in is a very important part of a plug-in. Most users want to learn something if they are using the plug-in. Without a good explanation they don't know what they're actually doing. It is a good practice to change its content dynamically. After the start of the plug-in the description group box gives a short introduction to the subject and to the plug-in itself. While clicking on buttons or in input fields the description group box can explain the different input and output values or even explain background computations. Concentrate all descriptions in one group box; different description fields can distract the user.

Window Resizing

Because the windows size depends on user behavior, the GUI has to be flexible. It is recommended to adjust the GUI elements on its actual window size, to use docking and not to position absolutely. However one has to define a minimum width and height in order to use the plug-in in small windows. Also some elements don't look appropriate if they are stretched. The programmer has to find a compromise between usability and optic.

In these pictures you can see the Elliptic Curve plug-in again. The graph is the element which profits most from a huge window. Therefore it makes sense to resize the graph only and implement the other elements static. If the window is very deformed the graph stops resizing and scrollbars are added.



Elliptic Curve Calculations

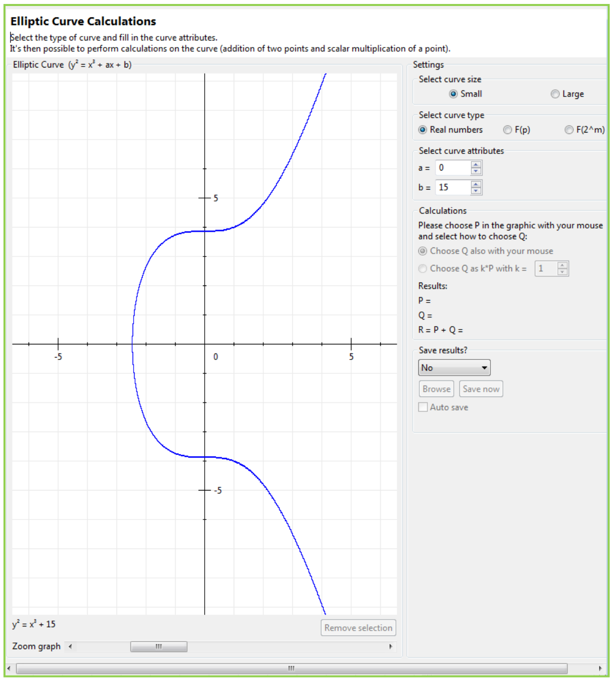
Select the type of curve and fill in the curve attributes.
It's then possible to perform calculations on the curve (addition of two points and scalar multiplication of a point).

Elliptic Curve ($y^2 = x^3 + ax + b$)

Settings
Select curve size: ● Small ○ Large
Select curve type: ● Real numbers ○ F(p) ○ F(2^m)
Select curve attributes: a = 0, b = 15
Calculations
Please choose P in the graphic with your mouse and select how to choose Q:
◉ Choose Q also with your mouse
○ Choose Q as k*P with k = 1
Results:
P =
Q =
R = P + Q =
Save results? No
☐ Auto save

$y^2 = x^3 + 15$

Figure 14: ECC resized using maximum size



Elliptic Curve Calculations

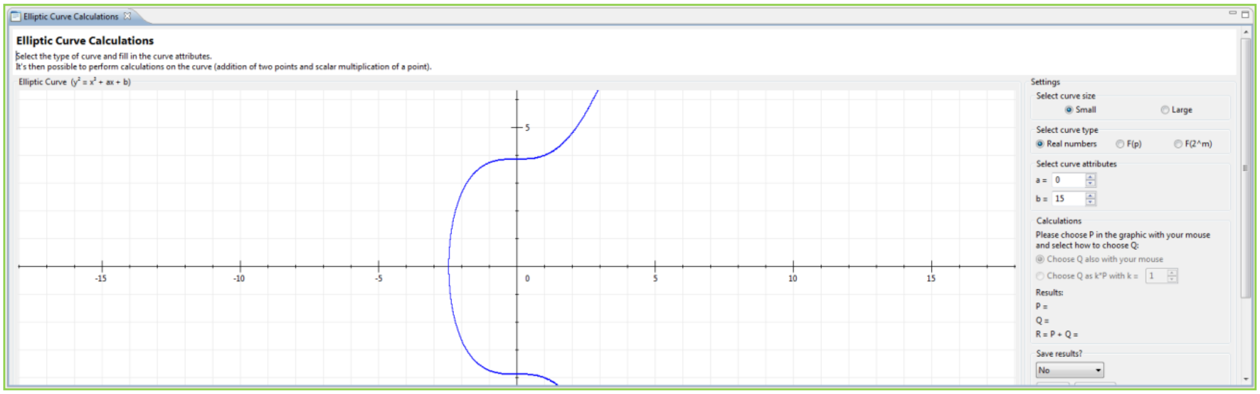
Select the type of curve and fill in the curve attributes.
It's then possible to perform calculations on the curve (addition of two points and scalar multiplication of a point).

Elliptic Curve ($y^2 = x^3 + ax + b$)

Settings
Select curve size: ● Small ○ Large
Select curve type: ● Real numbers ○ F(p) ○ F(2^m)
Select curve attributes: a = 0, b = 15
Calculations
Please choose P in the graphic with your mouse and select how to choose Q:
◉ Choose Q also with your mouse
○ Choose Q as k*P with k = 1
Results:
P =
Q =
R = P + Q =
Save results? No
☐ Auto save

$y^2 = x^3 + 15$

Figure 15: ECC resized slim



Elliptic Curve Calculations

Select the type of curve and fill in the curve attributes.
It's then possible to perform calculations on the curve (addition of two points and scalar multiplication of a point).

Elliptic Curve ($y^2 = x^3 + ax + b$)

Settings
Select curve size: ● Small ○ Large
Select curve type: ● Real numbers ○ F(p) ○ F(2^m)
Select curve attributes: a = 0, b = 15
Calculations
Please choose P in the graphic with your mouse and select how to choose Q:
◉ Choose Q also with your mouse
○ Choose Q as k*P with k = 1
Results:
P =
Q =
R = P + Q =
Save results? No

Figure 16: ECC resized broad

[1] Source: http://upload.wikimedia.org/wikipedia/commons/5/50/Man-in-the-middle_attack_of_Diffie-Hellman_key_agreement.svg