Getting started with JCrypTool development

May 2013 Edition

Getting started with JCrypTool development

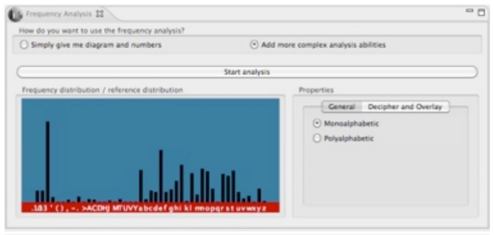
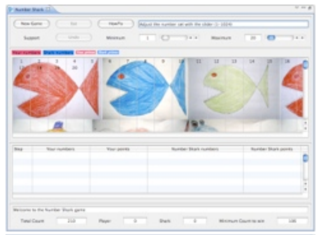
- JCrypTool – the cryptography e-learning platform
- Developing your own plug-in – extending JCrypTool
- Resources for a fast start – getting to know JCrypTool

Getting started with JCrypTool development

- JCrypTool – the cryptography e-learning platform
- Developing your own plug-in – extending JCrypTool
- Resources for a fast start – getting to know JCrypTool

Experience cryptography

- From classic to modern cryptography
  - Algorithms (classic, symmetric, asymmetric, hybrid, xml)
  - Analysis
  - Games
  - Visualizations
- Extensive help with user guide and cryptographic theory
- Record, replay, and share sequential crypto cascades
- Console to enter crypto commands directly inside JCrypTool
- Available in German and English

Develop cryptography

- Based on the Eclipse Rich Client Platform (RCP) 3.7
  - Modern user interface
  - Extremely extendable
  - Reusable plug-ins
- Open source
  - Licensed under the Eclipse Public License (EPL) 1.0
- Available for different platforms
  - Supports 32 and 64 bit operating systems

JCrypTool offers almost 20 special extension points which provide easy extensibility for new crypto plug-ins.

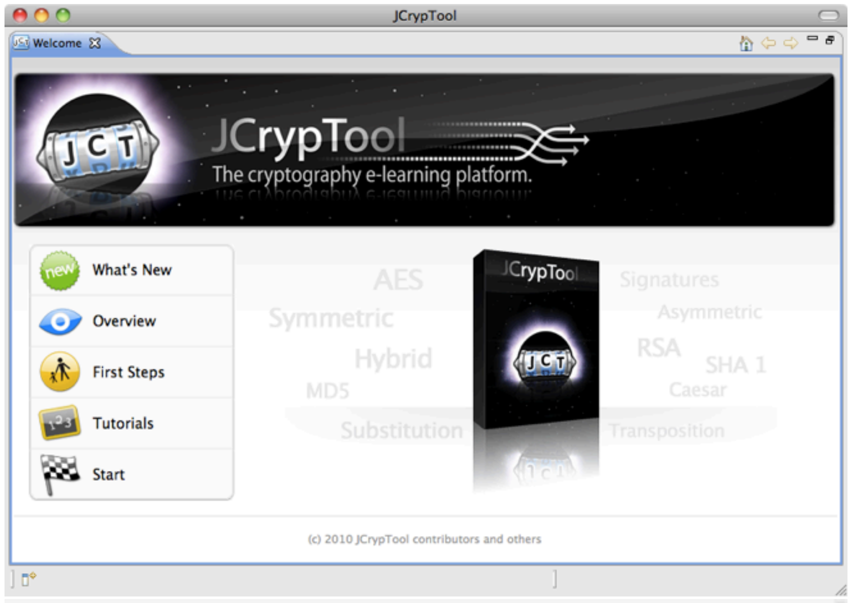
Obtaining JCrypTool

- Download
  - Visit http://www.cryptool.org and follow the **Download** link to get the zip archive or installer for your operating system
- Install
  - Extract/ install the downloaded file into an empty directory (the **jcryptool** directory is automatically created)
- Launch
  - Launch **JCrypTool**
    JCrypTool.exe JCrypTool.app JCrypTool.sh
  - JCrypTool uses English or German based on your regional settings (the language can be changed in the JCrypTool preferences)

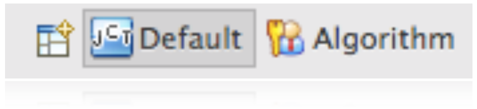
JCrypTool requires Java 1.6 or newer.

Get an overview on the Welcome Page

- After the first launch, the **Welcome Page** shows up
  - Provides an **Overview** and some **Tutorials** as an introduction
  - Click on **Start** to begin your JCrypTool experience
  - The **Welcome Page** can be reopened via the **Help** menu (entry **Welcome**)

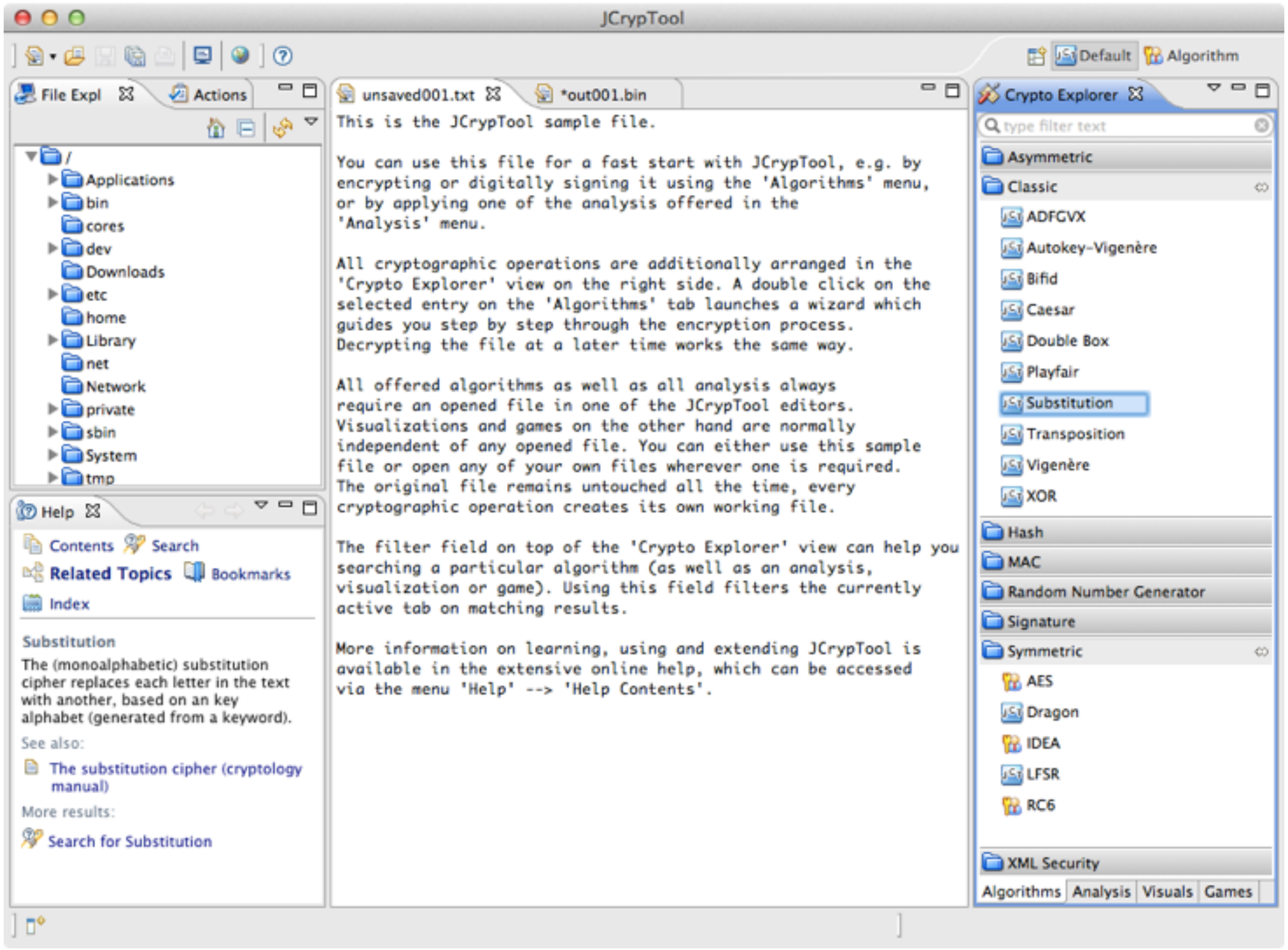


Perspectives, views and editors



- JCrypTool contains two perspectives
  - **Default**: document-centric
  - **Algorithm**: function-centric
  - Use the perspective switcher in the upper right to switch perspectives
- Each perspective offers its own set of views
  - Views are independent in each perspective
  - Views can be rearranged, resized, and closed
- Two editors, hex and text, are included
  - Use the **Edit – Open with** menu in order to switch to another editor
  - Right-click a file in the view **File Explorer** and select **Open with** to directly choose the desired editor

Default Perspective



```
This is the JCrypTool sample file.

You can use this file for a fast start with JCrypTool, e.g. by
encrypting or digitally signing it using the 'Algorithms' menu,
or by applying one of the analysis offered in the
'Analysis' menu.

All cryptographic operations are additionally arranged in the
'Crypto Explorer' view on the right side. A double click on the
selected entry on the 'Algorithms' tab launches a wizard which
guides you step by step through the encryption process.
Decrypting the file at a later time works the same way.

All offered algorithms as well as all analysis always
require an opened file in one of the JCrypTool editors.
Visualizations and games on the other hand are normally
independent of any opened file. You can either use this sample
file or open any of your own files wherever one is required.
The original file remains untouched all the time, every
cryptographic operation creates its own working file.

The filter field on top of the 'Crypto Explorer' view can help you
searching a particular algorithm (as well as an analysis,
visualization or game). Using this field filters the currently
active tab on matching results.

More information on learning, using and extending JCrypTool is
available in the extensive online help, which can be accessed
via the menu 'Help' --> 'Help Contents'.
```

Substitution

The (monoalphabetic) substitution cipher replaces each letter in the text with another, based on an key alphabet (generated from a keyword).

See also:

The substitution cipher (cryptology manual)

More results:

Search for Substitution

Document-centric access to JCrypTool

- The main document-centric perspective
  - Provides easy access to all JCrypTool functionality
  - Dynamic help (lower left) shows help for the active part (if available)
  - Open **Help – Help Contents** for the complete help
- JCrypTool is file-based
  - Open your file in one of the editors (hex or text)
  - Select your file in the File Explorer view and use the context menu to apply a cryptographic function to it
  - The original file remains untouched, every cryptographic operation creates a new file

Some views, like **Visualizations**, provide simulations and do not require any file input.

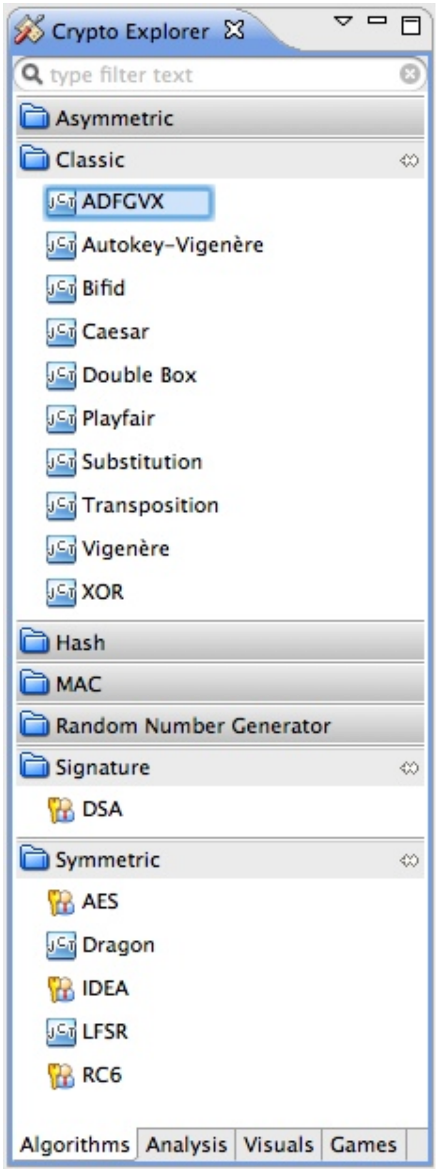
Cryptographic actions

1. Open your file or create a new one
2. Choose the cryptographic operation
    1. By double clicking on an entry in the **Crypto Explorer** view,
    2. By dragging and dropping an entry from the **Crypto Explorer** view onto an open editor, *or*
    3. By selecting an entry in the **main menu**
3. Provide the required information in the **wizard**
4. Click on **Finish** when done

Most wizards and views provide context sensitive help by clicking on the **help icon** or pressing **F1**.

Most crypto plug-ins provide background information in the help system.

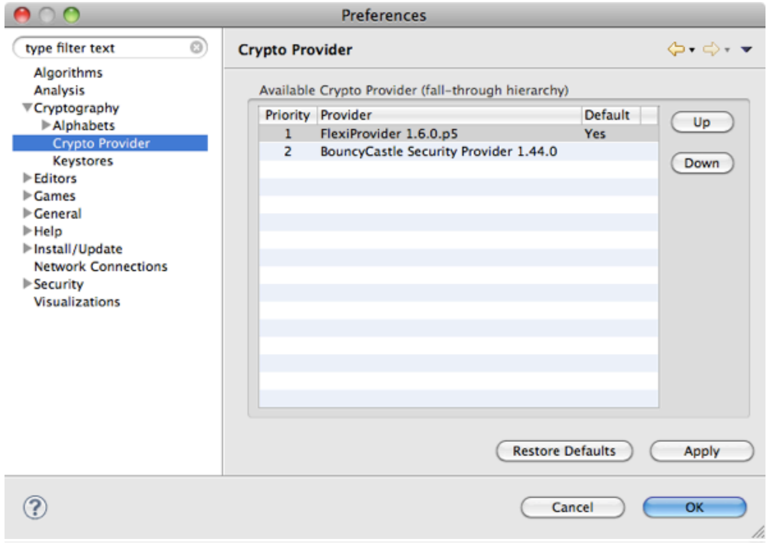
Accessing cryptographic plug-ins



- Four groups are available
  - Algorithms
  - Analysis
  - Visuals
  - Games
- Accessible via
  - The **main menu**
  - The **Crypto Explorer** view
    - Switch with the four tabs at the bottom
    - Search for an algorithm with the search field at the top
- Available algorithms
  - Menu and view entries are identical and depend on the installed crypto plug-ins

JCrypTool comes with two crypto providers

- FlexiProvider is the default crypto provider
  - BouncyCastle is also available
  - Other crypto providers can be installed as plug-ins
- Preferences
  - **Crypto Providers:** Select your default provider



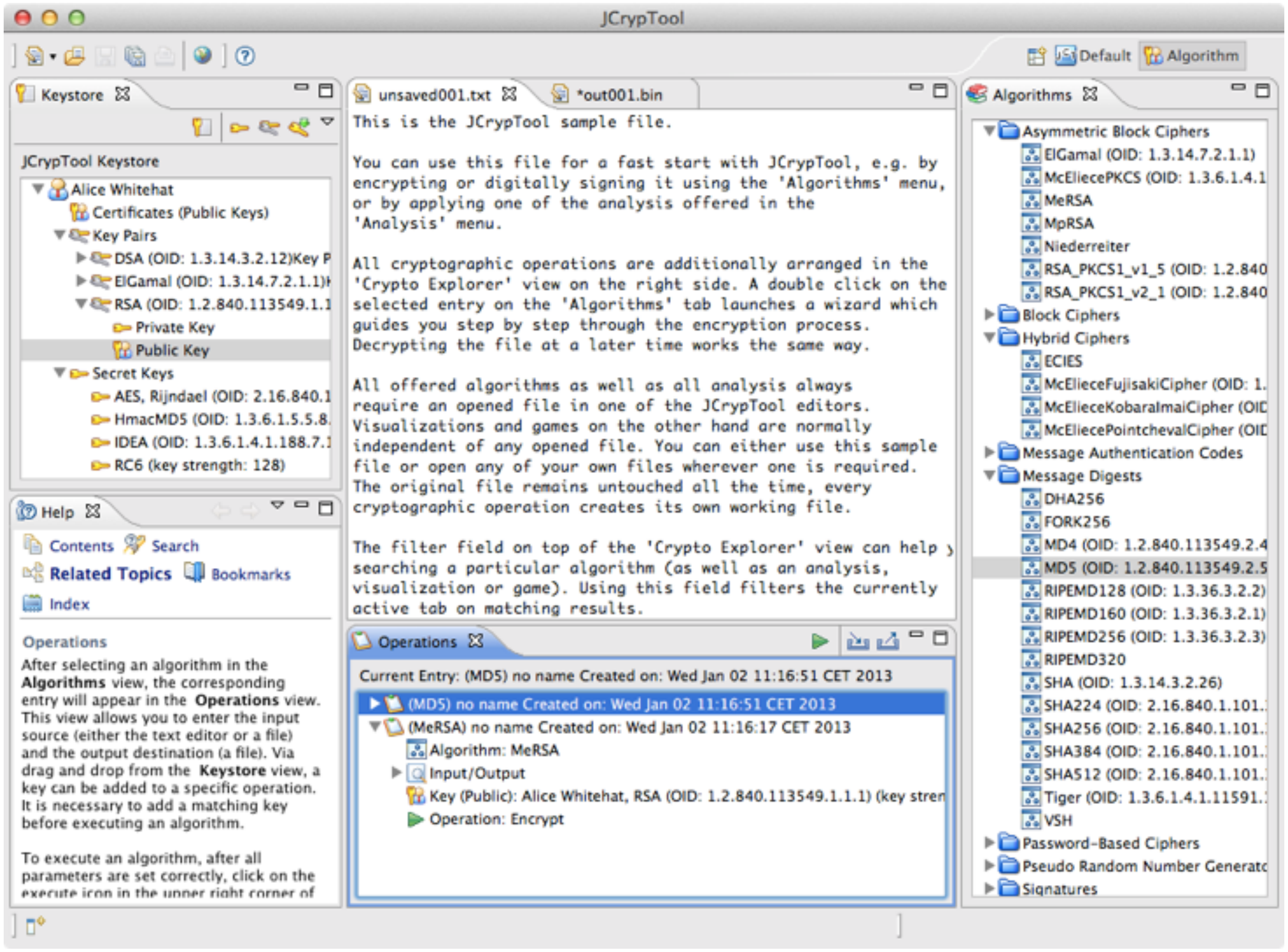
Crypto Provider

Available Crypto Provider (fall-through hierarchy)

Priority	Provider	Default
1	FlexiProvider 1.6.0.p5	Yes
2	BouncyCastle Security Provider 1.44.0	

Fall-through hierarchy: JCrypTool searches all installed providers in the defined order to find an implementation of the selected algorithm.

Algorithm Perspective


JCrypTool
Default
Algorithm
Keystore
JCrypTool Keystore
Alice Whitehat
Certificates (Public Keys)
Key Pairs
DSA (OID: 1.3.14.3.2.12)Key P
ElGamal (OID: 1.3.14.7.2.1.1)K
RSA (OID: 1.2.840.113549.1.1
Private Key
Public Key
Secret Keys
AES, Rijndael (OID: 2.16.840.1
HmacMD5 (OID: 1.3.6.1.5.5.8.
IDEA (OID: 1.3.6.1.4.1.188.7.1
RC6 (key strength: 128)
Help
Contents
Search
Related Topics
Bookmarks
Index
unsaved001.txt
*out001.bin
Operations
Algorithms
Asymmetric Block Ciphers
ElGamal (OID: 1.3.14.7.2.1.1)
McEliecePKCS (OID: 1.3.6.1.4.1
MeRSA
MpRSA
Niederreiter
RSA_PKCS1_v1_5 (OID: 1.2.840
RSA_PKCS1_v2_1 (OID: 1.2.840
Block Ciphers
Hybrid Ciphers
ECIES
McElieceFujisakiCipher (OID: 1.
McElieceKobaraImaiCipher (OID
McEliecePointchevalCipher (OID
Message Authentication Codes
Message Digests
DHA256
FORK256
MD4 (OID: 1.2.840.113549.2.4
MD5 (OID: 1.2.840.113549.2.5
RIPEMD128 (OID: 1.3.36.3.2.2)
RIPEMD160 (OID: 1.3.36.3.2.1)
RIPEMD256 (OID: 1.3.36.3.2.3)
RIPEMD320
SHA (OID: 1.3.14.3.2.26)
SHA224 (OID: 2.16.840.1.101.
SHA256 (OID: 2.16.840.1.101.
SHA384 (OID: 2.16.840.1.101.
SHA512 (OID: 2.16.840.1.101.
Tiger (OID: 1.3.6.1.4.1.11591.
VSH
Password-Based Ciphers
Pseudo Random Number Generat
Signatures

```
This is the JCrypTool sample file.

You can use this file for a fast start with JCrypTool, e.g. by
encrypting or digitally signing it using the 'Algorithms' menu,
or by applying one of the analysis offered in the
'Analysis' menu.

All cryptographic operations are additionally arranged in the
'Crypto Explorer' view on the right side. A double click on the
selected entry on the 'Algorithms' tab launches a wizard which
guides you step by step through the encryption process.
Decrypting the file at a later time works the same way.

All offered algorithms as well as all analysis always
require an opened file in one of the JCrypTool editors.
Visualizations and games on the other hand are normally
independent of any opened file. You can either use this sample
file or open any of your own files wherever one is required.
The original file remains untouched all the time, every
cryptographic operation creates its own working file.

The filter field on top of the 'Crypto Explorer' view can help y
searching a particular algorithm (as well as an analysis,
visualization or game). Using this field filters the currently
active tab on matching results.
```

Operations

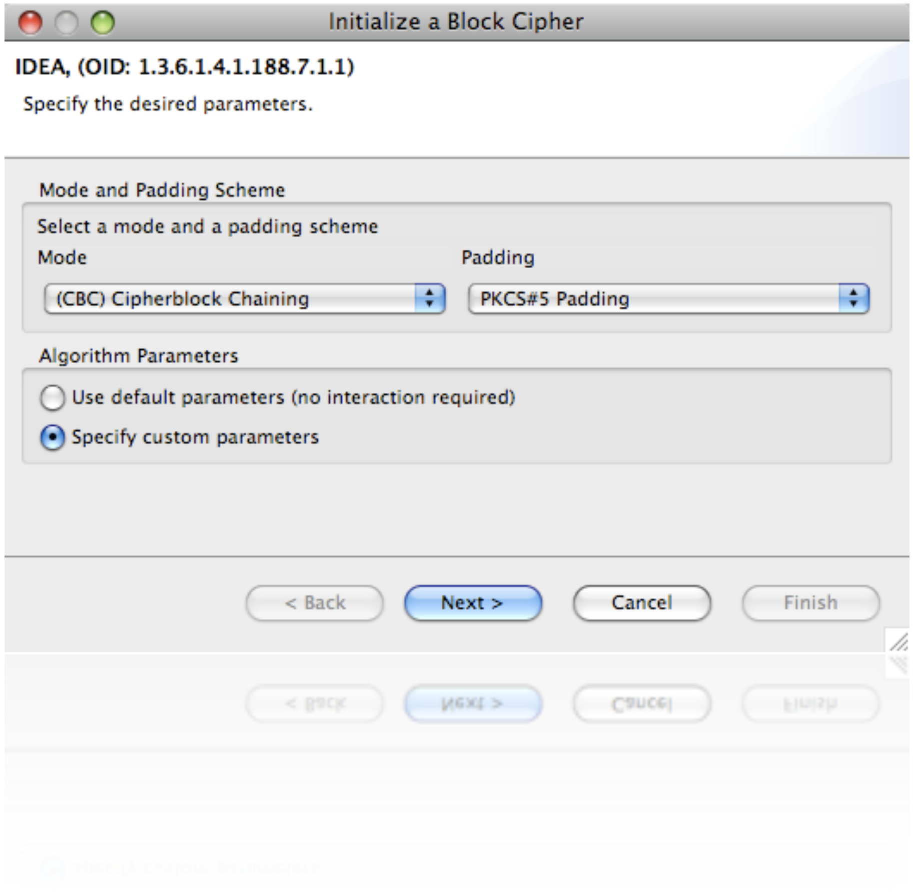
After selecting an algorithm in the **Algorithms** view, the corresponding entry will appear in the **Operations** view. This view allows you to enter the input source (either the text editor or a file) and the output destination (a file). Via drag and drop from the **Keystore** view, a key can be added to a specific operation. It is necessary to add a matching key before executing an algorithm.

To execute an algorithm, after all parameters are set correctly, click on the execute icon in the upper right corner of

```
Current Entry: (MD5) no name Created on: Wed Jan 02 11:16:51 CET 2013
▶ (MD5) no name Created on: Wed Jan 02 11:16:51 CET 2013
▼ (MeRSA) no name Created on: Wed Jan 02 11:16:17 CET 2013
    Algorithm: MeRSA
  ▶ Input/Output
    Key (Public): Alice Whitehat, RSA (OID: 1.2.840.113549.1.1.1) (key stren
    Operation: Encrypt
```

Function-centric access to the FlexiProvider

- Advanced operations with FlexiProvider
  - Dynamic wizards directly linked to the FlexiProvider library
    - Default parameters (no interaction required)
    - Custom parameters (choose every possible parameter by yourself)



Initialize a Block Cipher

IDEA, (OID: 1.3.6.1.4.1.188.7.1.1)

Specify the desired parameters.

Mode and Padding Scheme

Select a mode and a padding scheme

Mode: (CBC) Cipherblock Chaining

Padding: PKCS#5 Padding

Algorithm Parameters

Use default parameters (no interaction required) [✓ Specify custom parameters]

Getting started with JCrypTool development

JCrypTool – the cryptography e-learning platform

Developing your own plug-in – extending JCrypTool

Resources for a fast start – getting to know JCrypTool

JCrypTool is an Eclipse RCP application

- Eclipse Rich Client Platform (RCP)
  - Collection of plug-ins and a runtime
  - Provides basic functionality
    - Preferences
    - Help system
    - And plenty more
  - Integrates update functionality
  - Contains supporting views
    - Error log
    - Progress view

ⓘ There are differences between plug-in and *normal* Java application development. An existing Java application won't work without modifications as an Eclipse plug-in.

Everything is a plug-in

- A plug-in is the smallest deployable component
- A plug-in provides one kind of functionality, such as
  - An implementation of the AES algorithm
  - An implementation of a DES brute-force-attack
  - A visualization of a cryptographic process
  - …
- In fact, almost every project in our repositories is a plug-in
  - Exceptions
    - The feature projects ending on **.feature**
    - The build projects **org.jcryptool.product**, **org.jcryptool.releng** or **org.jcryptool.repository**

A feature bundles one or more plug-ins

- A feature is a simple container
  - Bundles plug-ins that belong together
  - Does not contain any code
  - Is a separate project that ends on the name **.feature**
- JCrypTool is completely feature based
  - Every plug-in must be included in one
  - Makes installations easier
  - Reduces or removes dependency problems

There a lots of features already available in JCrypTool. Before creating a new one, make sure that there is not already a suitable feature for your plug-in.

Requirements for your JCrypTool development

- **Java** (1.6 or newer) http://www.oracle.com/technetwork/java
  - **Java Runtime Environment** (JRE)
    or
  - **Java Development Kit** (JDK)
- **Eclipse** (3.7 or newer) http://www.eclipse.org
  - **Eclipse Classic**
    or
  - **Eclipse for RCP and RAP Developers**
- **Git** plug-in
  - **EGit** http://www.eclipse.org/egit

JCrypTool Core and Crypto repositories

- The **Core Repository** contains the main platform
  - Runtime
  - Logging, help, preferences
  - Crypto providers (FlexiProvider and BouncyCastle)
  - Editors (hex and text)
  - Views (Actions, Commands, Crypto Explorer, File Explorer, Web browser)
- The **Crypto Repository** contains the crypto plug-ins
  - Analysis
  - Classic, modern, hybrid and xml security algorithms
  - Games
  - Visualizations

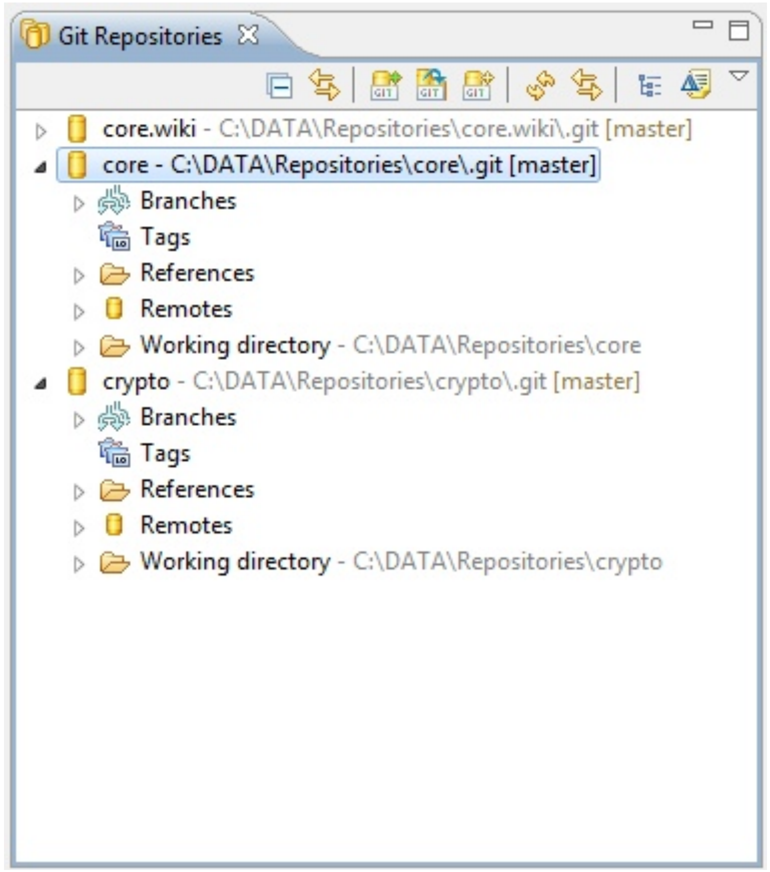
Obtaining the JCrypTool sources

- Two repositories
  - Both offer **anonymous read access**
  - A **GitHub user** (and registration with the JCrypTool project) is required **for write access**
  - Feel free to create a fork for the JCrypTool project and start right away
- **JCrypTool Core**
  https://github.com/jcryptool/core
- **JCrypTool Crypto**
  https://github.com/jcryptool/crypto

As a developer you must check out all core repository plug-ins. The crypto plug-ins are optional, but they make development easier since you can use them as samples.

Using EGit to check out the projects

- Add both repository locations and clone the repositories
- Right click on each repository and choose **Import Projects...**



```
▷ core.wiki - C:\DATA\Repositories\core.wiki\.git [master]
◢ core - C:\DATA\Repositories\core\.git [master]
    ▷ Branches
      Tags
    ▷ References
    ▷ Remotes
    ▷ Working directory - C:\DATA\Repositories\core
◢ crypto - C:\DATA\Repositories\crypto\.git [master]
    ▷ Branches
      Tags
    ▷ References
    ▷ Remotes
    ▷ Working directory - C:\DATA\Repositories\crypto
```

Visit our wiki for more information on our repositories: https://github.com/jcryptool/core/wiki/Getting-started-as-a-JCrypTool-Developer

Starting JCrypTool in your Eclipse IDE

- First launch
  - Open the file **jcryptool.product** located in the **org.jcryptool.repository** project
  - Click on the **Launch an Eclipse application** link in the lower left
- Two things happen now
  - JCrypTool is started
  - A new run configuration is created
- Later launches
  - Either use the **Launch an Eclipse application** link again
  - Or use the generated **run configuration**

ⓘ Visit our wiki for more information on how to get started: https://github.com/jcryptool/core/wiki/Getting-started-as-a-JCrypTool-Developer

JCrypTool divides plug-ins into related groups (1)

- Mainly view based plug-ins
  - **Analysis** plug-ins
    - Example plug-in: org.jcryptool.analysis.freqanalysis
    - Branding plug-in: org.jcryptool.analysis
    - Feature: org.jcryptool.analysis
  - **Visualization** plug-ins
    - Example plug-in: org.jcryptool.visual.ecc
    - Branding plug-in: org.jcryptool.visual
    - Feature: org.jcryptool.visual
  - **Game** plug-ins
    - Example plug-in: org.jcryptool.games.numbershark
    - Branding plug-in: org.jcryptool.games
    - Feature: org.jcryptool.games

JCrypTool divides plug-ins into related groups (2)

- Mainly wizard based plug-ins
  - **Classic** algorithm plug-ins
    - Example plug-in: org.jcryptool.crypto.classic.caesar
    - Branding plug-in: org.jcryptool.crypto.classic
    - Feature: org.jcryptool.crypto.classic
  - **Modern** algorithm plug-ins
    - Example plug-in: org.jcryptool.crypto.modern.stream.dragon
    - Branding plug-in: org.jcryptool.crypto.modern
    - Feature: org.jcryptool.crypto.modern

It is not required that classic and modern plug-ins must be wizard-based, nor that analysis, game, and visualization plug-ins must be view-based, but that tends to be how it works out.

Creating an official JCrypTool plug-in

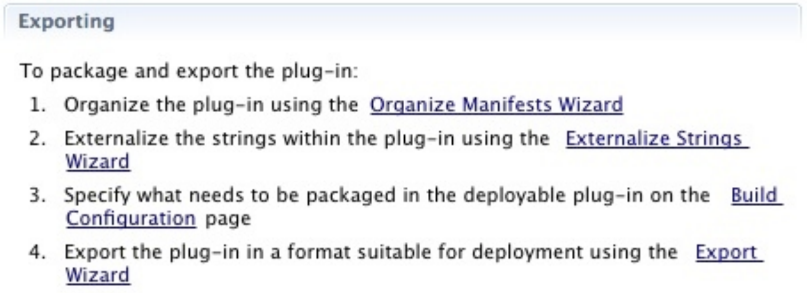
- Official JCrypTool plug-ins (hosted in our repository)
  - Must have a name starting with **org.jcryptool.**
  - Followed by one of the following (or a new one)
    - analysis.[name]
    - games.[name]
    - visual.[name]
    - crypto.classic.[name]
    - crypto.modern.[name]
    - crypto.xml.[name]
- Add your plug-in(s) to the corresponding feature
  - One feature project for every crypto plug-in family
  - Example
    - Plug-in: org.jcryptool.games.numbershark
    - Feature: org.jcryptool.games

Creating an unofficial JCrypTool plug-in

- Unofficial plug-ins (hosted by yourself)
  - Can have any name
  - Stick to Java recommendations (reverse domain name)
- Create a feature for your plug-ins
  - Always provide a feature for your plug-ins
- Shipping your plug-ins
  - Offer downloadable archives for your plug-ins
  - Create your own update site
    - Users can manually add this site to their JCrypTool installation

Exporting your plug-in

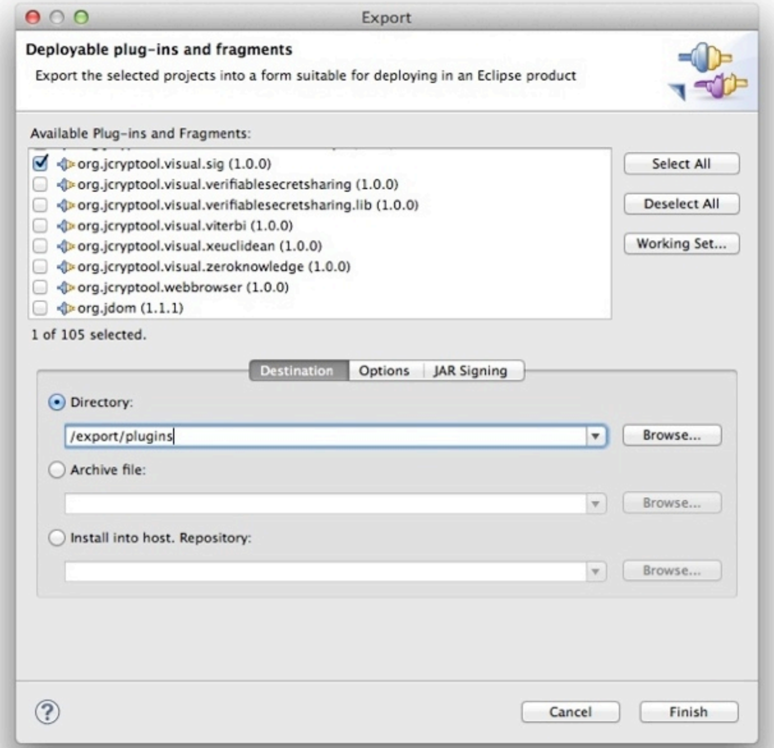
- Use the default Eclipse export wizard to export your plug-in into a single jar file



Exporting

To package and export the plug-in:

1. Organize the plug-in using the Organize Manifests Wizard
2. Externalize the strings within the plug-in using the Externalize Strings Wizard
3. Specify what needs to be packaged in the deployable plug-in on the Build Configuration page
4. Export the plug-in in a format suitable for deployment using the Export Wizard



Export

Deployable plug-ins and fragments

Export the selected projects into a form suitable for deploying in an Eclipse product

Available Plug-ins and Fragments:

- [✓] org.jcryptool.visual.sig (1.0.0)
- [ ] org.jcryptool.visual.verifiablesecretsharing (1.0.0)
- [ ] org.jcryptool.visual.verifiablesecretsharing.lib (1.0.0)
- [ ] org.jcryptool.visual.viterbi (1.0.0)
- [ ] org.jcryptool.visual.xeuclidean (1.0.0)
- [ ] org.jcryptool.visual.zeroknowledge (1.0.0)
- [ ] org.jcryptool.webbrowser (1.0.0)
- [ ] org.jdom (1.1.1)

1 of 105 selected.

Destination | Options | JAR Signing

● Directory: /export/plugins

○ Archive file:

○ Install into host. Repository:

- Place this file in the JCrypTool **dropins** folder and start JCrypTool

See https://github.com/jcryptool/core/wiki/Creating-a-new-Release for more information

Development hints

- Choose the crypto plug-in group
  - Make sure it matches with the plug-in you intend to develop
  - Learn from the existing plug-ins in the corresponding group
- Reuse existing functionality
  - Especially when provided by extension points
  - Especially when requiring existing third party jars
  - Use the **org.jcryptool.core.util** plug-in, which provides different services and interfaces used all over JCrypTool
- JCrypTool is an e-learning software
  - An extensive help with (cryptographic) background information and a tutorial should be part of your plug-in
  - Context sensitive help provides immediate support
  - Cheat Sheets provide a guided tour for new users

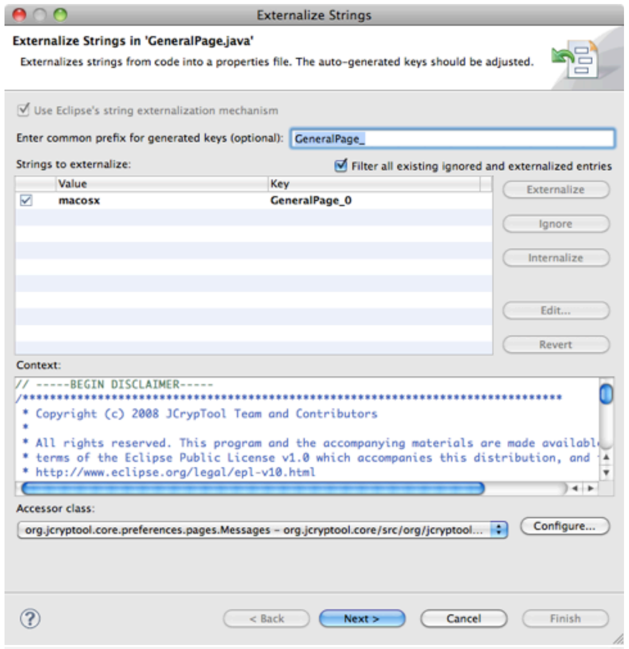
Mind some Eclipse RCP restrictions and quirks

- No garbage collection in SWT
  - Clean up all used OS resources after usage, especially fonts and images
- JCrypTool is shipped for multiple platforms
  - Exotic fonts may not be available on all platforms
  - Use the default font wherever possible
- Loading resources
  - Every plug-in ends up in a jar-file, which influences the required path to load a resource like an image

The org.jcryptool.core.util plug-in makes font handling easy: Simply request a font, the plug-in frees you from all other tasks.

Internationalize your plug-in

- English GUI and help are a must have
  - German is optional but strongly requested
- Use the **Externalize Strings** wizard in the **Source** menu
  - Always mark the checkbox **Use Eclipse's string externalization mechanism**
  - Provide property files for every supported language



Externalize Strings in 'GeneralPage.java'

Externalizes strings from code into a properties file. The auto-generated keys should be adjusted.

[✓] Use Eclipse's string externalization mechanism

Enter common prefix for generated keys (optional): GeneralPage_

Strings to externalize: [✓] Filter all existing ignored and externalized entries

	Value	Key
✓	macosx	GeneralPage_0

Context:

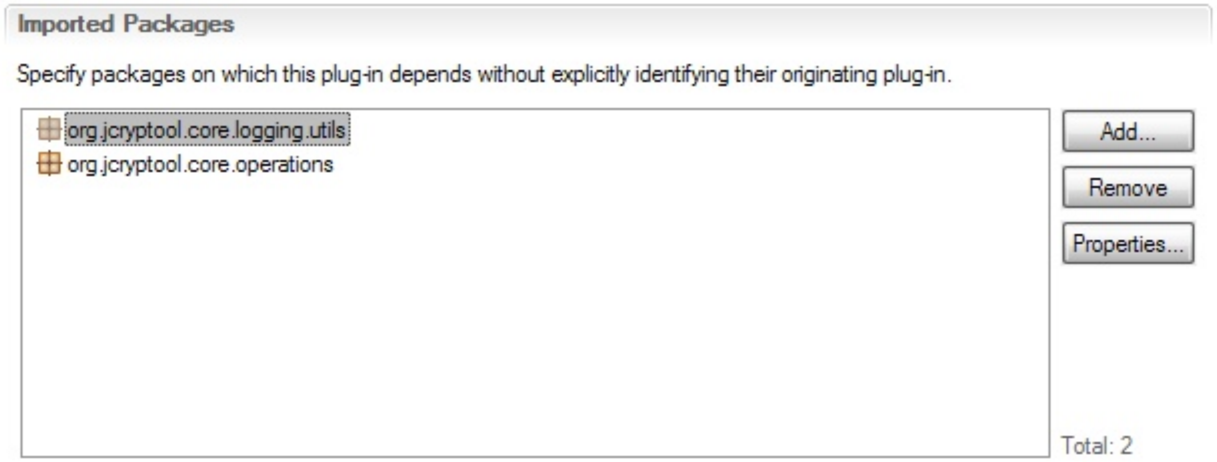
```
// -----BEGIN DISCLAIMER-----
/*******************************************************************************
 * Copyright (c) 2008 JCrypTool Team and Contributors
 *
 * All rights reserved. This program and the accompanying materials are made availabl
 * terms of the Eclipse Public License v1.0 which accompanies this distribution, and
 * http://www.eclipse.org/legal/epl-v10.html
```

Accessor class: org.jcryptool.core.preferences.pages.Messages – org.jcryptool.core/src/org/jcryptool...

- Externalize plugin.xml and MANIFEST.MF files
  - Use **PDE Tools - Externalize Strings**

Use the JCrypTool logging plug-in

- Use org.jcryptool.core.logging
  - Add the package **org.jcryptool.core.logging.utils** as dependency via **Imported Packages**
- The helper class LogUtil provides easy access points
  - LogUtil.logInfo(“*message*”)
  - LogUtil.logWarning(“*message*”)
  - LogUtil.logError(“*message*”)


Imported Packages

Specify packages on which this plug-in depends without explicitly identifying their originating plug-in.

- org.jcryptool.core.logging.utils
- org.jcryptool.core.operations

Using the JCrypTool extension points

- Documentation
  - Most of the extension points include documentation and an implementation sample
- Find a sample implementation
  1. Open the **plugin.xml** of the plug-in that provides the extension point
  2. Switch to the **Extension Points** tab
  3. Select the desired extension point in the list
  4. Click on the **Find references** link and examine the implementation

An extension point is a public API.

Available extension points in JCrypTool

- **org.jcryptool.core**
  - editorButton
  - platformLanguage
- **org.jcryptool.core.operations**
  - algorithms
  - alphabets
  - analysis
  - editorServices
  - games
  - keystores
  - operationsManager
  - pkcsFactories
  - providers
  - providers2
  - visuals
- **org.jcryptool.commands.core**
  - commands
- **org.jcryptool.crypto.flexiprovider.algorithms**
  - newOperation
- **org.jcryptool.crypto.flexiprovider.operations**
  - performFlexiProviderOperation
- **org.jcryptool.crypto.keystore**
  - keyStoreActions

Optimizing your run menu configuration

- **Main** tab
  - Activate the **Clear** checkbox to start with an empty runtime workspace all the time
- **Arguments** tab
  - Add **-consolelog** as last parameter in the **Program Arguments** box
  - Replace the **-nl ${target.nl}** in the **Program Arguments** box with **-nl en** for the English JCrypTool
- **Plug-ins** tab
  - Click the **Add Required Plug-ins** button

Name this run menu entry **JCrypTool English**, copy it, name the new one **JCrypTool German** and replace **-nl en** with **-nl de** for the English version on the Arguments tab.

Getting started with JCrypTool development

- JCrypTool – the cryptography e-learning platform
- Developing your own plug-in – extending JCrypTool
- Resources for a fast start – getting to know JCrypTool

Become a part of the JCrypTool community

- Discussion Groups
  - http://groups.google.com/group/jcryptool-developers
  - http://groups.google.com/group/jcryptool-users
- GitHub
  - https://github.com/jcryptool
- Issues
  - https://github.com/jcryptool/core/issues
  - https://github.com/jcryptool/crypto/issues
- Web
  - http://www.cryptool.org
- Wiki
  - https://github.com/jcryptool/core/wiki

Getting in touch with JCrypTool

- JCrypTool project lead
  - Dominik Schadow – dominikschadow@gmail.com

- CrypTool project lead
  - Prof. Bernhard Esslinger – esslinger@fb5.uni-siegen.de

JCrypTool Core project (1)

Core (org.jcryptool.core.feature)	org.jcryptool.core
	org.jcryptool.core.action
	org.jcryptool.core.cryptosystem
	org.jcryptool.core.help
	org.jcryptool.core.logging
	org.jcryptool.core.nl
	org.jcryptool.core.operations
	org.jcryptool.core.util
	org.jcryptool.core.views
Views (org.jcryptool.views.feature)	org.jcryptool.actions.core
	org.jcryptool.actions.ui
	org.jcryptool.commands.core
	org.jcryptool.commands.ui
	org.jcryptool.fileexplorer
	org.jcryptool.webbrowser

JCrypTool Core project (2)

Providers (org.jcryptool.providers.feature)	de.flexiprovider
	org.bouncycastle
Crypto (org.jcryptool.crypto.feature)	org.jcryptool.crypto
	org.jcryptool.crypto.keystore
FlexiProvider (org.jcryptool.crypto.flexiprovider.feature)	org.jcryptool.crypto.flexiprovider
	org.jcryptool.crypto.flexiprovider.algorithms
	org.jcryptool.crypto.flexiprovider.engines
	org.jcryptool.crypto.flexiprovider.integrator
	org.jcryptool.crypto.flexiprovider.keystore
	org.jcryptool.crypto.flexiprovider.operations
Editors (org.jcryptool.editors.feature)	org.jcryptool.editor.text
	net.sourceforge.ehep

JCrypTool Crypto project (1)

Analysis (org.jcryptool.analysis.feature)	org.jcryptool.analysis
	org.jcryptool.analysis.entropy
	org.jcryptool.analysis.freqanalysis
	org.jcryptool.analysis.friedman
	org.jcryptool.analysis.graphtools
	org.jcryptool.analysis.kegver
	org.jcryptool.analysis.textmodify
	org.jcryptool.analysis.transpositionanalysis
	org.jcryptool.analysis.vigenere

JCrypTool Crypto project (2)

Classic Algorithms
(org.jcryptool.crypto.classic.feature)

- org.jcryptool.crypto.classic
- org.jcryptool.crypto.classic.adfgvx
- org.jcryptool.crypto.classic.alphabets
- org.jcryptool.crypto.classic.autovigenere
- org.jcryptool.crypto.classic.caesar
- org.jcryptool.crypto.classic.delastelle
- org.jcryptool.crypto.classic.doppelkasten
- org.jcryptool.crypto.classic.model
- org.jcryptool.crypto.classic.playfair
- org.jcryptool.crypto.classic.substitution
- org.jcryptool.crypto.classic.transposition
- org.jcryptool.crypto.classic.vernam
- org.jcryptool.crypto.classic.vigenere
- org.jcryptool.crypto.classic.xor

JCrypTool Crypto project (3)

Modern Algorithms (org.jcryptool.crypto.modern.feature)	org.jcryptool.crypto.modern
	org.jcryptool.crypto.modern.sha3
	org.jcryptool.crypto.modern.stream.dragon
	org.jcryptool.crypto.modern.stream.lfsr
XML Security (org.jcryptool.crypto.xml.feature)	org.jcryptool.crypto.xml.core
	org.jcryptool.crypto.xml.help
	org.jcryptool.crypto.xml.ui
	org.jcryptool.crypto.xml
Games (org.jcryptool.games.feature)	org.jcryptool.games
	org.jcryptool.games.numbershark
	org.jcryptool.games.sudoku

JCrypTool Crypto project (4)

Visualizations (1/2)
(org.jcryptool.visuals.feature)

- org.jcryptool.visual
- org.jcryptool.visual.aco
- org.jcryptool.visual.aup
- org.jcryptool.visual.crt
- org.jcryptool.visual.des
- org.jcryptool.visual.dsa
- org.jcryptool.visual.ecc
- org.jcryptool.visual.ecdh
- org.jcryptool.visual.elGamal
- org.jcryptool.visual.extendedrsa
- org.jcryptool.visual.grille
- org.jcryptool.visual.he
- org.jcryptool.visual.kleptography
- org.jcryptool.visual.library

JCrypTool Crypto project (5)

Visualizations (2/2)
(org.jcryptool.visuals.feature)

- org.jcryptool.visual.pairingbd2
- org.jcryptool.visual.rsa
- org.jcryptool.visual.secretsharing
- org.jcryptool.visual.sidechannelattack.dpa
- org.jcryptool.visual.sidechannelattack.spa
- org.jcryptool.visual.verifiablesecretsharing
- org.jcryptool.visual.viterbi
- org.jcryptool.visual.xeuclidean
- org.jcryptool.visual

Happy coding…