Encryption Database and Biometrics Library and Software Suite (EnDaBi)

Aly Shmahell, Alya Salman, Elias Soud, Ruaa Sleiman

May 25, 2015

Documentation

(English Version)

This Documentation, along with Documentation in other languages, the Source Code, Licenses and all Material Related to the EnDaBi Project are registered and hosted on :

https://github.com/EnDaBi/EnDaBi

Index

License

GNU Free Documentation License

Version 1.3, 3 November 2008
Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed

using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “**publisher**” means any person or entity that distributes copies of the Document to the public.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

EnDaBi Licenses

Our Documentation is (as explained above) licensed under the GNU Free Documentation License.

However, our Source Code is Licensed under the GNU Lesser General Public License.

Legal

This Document is a Manual for a collection of Software Source Code, which along with the Documentation, was put together by a small team of young Engineers as a third year project at

Tishreen University, Latakia, Syria.

The Code Implements well known **Generic** Mathematical and Computational algorithms, using **Free Open Sourced** Programming Languages. Compiled and tested on **Free Open Sourced** Platforms using **Free Open Sourced** Software.

Our Work does **NOT** Include any Close Sourced or Propriety Software.

Some of our Combined Implementations are Genuine as in Different. However, the Idea Behind the Project and the Individual Implementations of the mathematical algorithms and formulas are all **Generic**. Which means there might be huge similarities with some of the works out there in the world.

In case that happens, if the resemblance is non-distinguishable, we are willing to re-implement. In the case that happens, please contact `aly.shmahell@gmail.com` and there will be a dialogue once we're able to read the email and respond.

Please Notice This does not mean we're open to **trolling**. any attempt to troll our work or drag us into an unlawful debate of ownership will **NOT be tolerated**.

Disclaimer

About The Authors

We are Syrian nationals, who consider themselves citizens of their Country and the World.

Below is the list of authors and their contact info :

Name	Contact
Aly Shmahell	aly.shmahell@gmail.com
Ruaa Sleiman	ruaa.s.sleiman@gmail.com
Elias Soud	Thegamebest21es@gmail.com
Alya Salman	el57la.9595@gmail.com

Special Thanks

We give our thanks to those who have helped us :

Eng. Sami Abobala from Tishreen University whose patience and assistance were crucial to our success.

Professor. Dr.Ing. Christof Paar from Ruhr-Universität Bochum, whose work we've built upon. and for his kindness in allowing us to use parts of his slides in our documentation.

Introduction

The caveman, the life choice he made to settle down from hunting and gathering provided him with more security inside his cave. but it limited his freedom to roam vast lands.

The same aspect can be found in modern society. If a company decreases its security measures on its entrances, the employees can get in and out faster (not having to pull so many cards out, remember so many passwords and such), but it loses some of its security points. and vice versa.

The EnDaBi Project tackles that problem, and tries to find balance between security and mobility.

By utilizing state of the art schemes in Encryption, Database and Biometrics technologies and embedding them all into one efficient and coherent library or suite.

About The Project

The premise of the project is simple. For starters, we're focusing on Encryption. We spent a fair amount of time searching for material that serve that particular purpose.

We found those published by Professor Christof Paar to be most suitable for implementing and most easy to understand.

The Current Project Steps :

1) Choosing the RSA algorithm for implementation. for the sake of mobility and security (it being an Asymmetric Cryptographic Scheme).

2) Trying to make the Implementation as Simple and Usable as possible.

3) Trying to Enhance Speed and Efficiency.

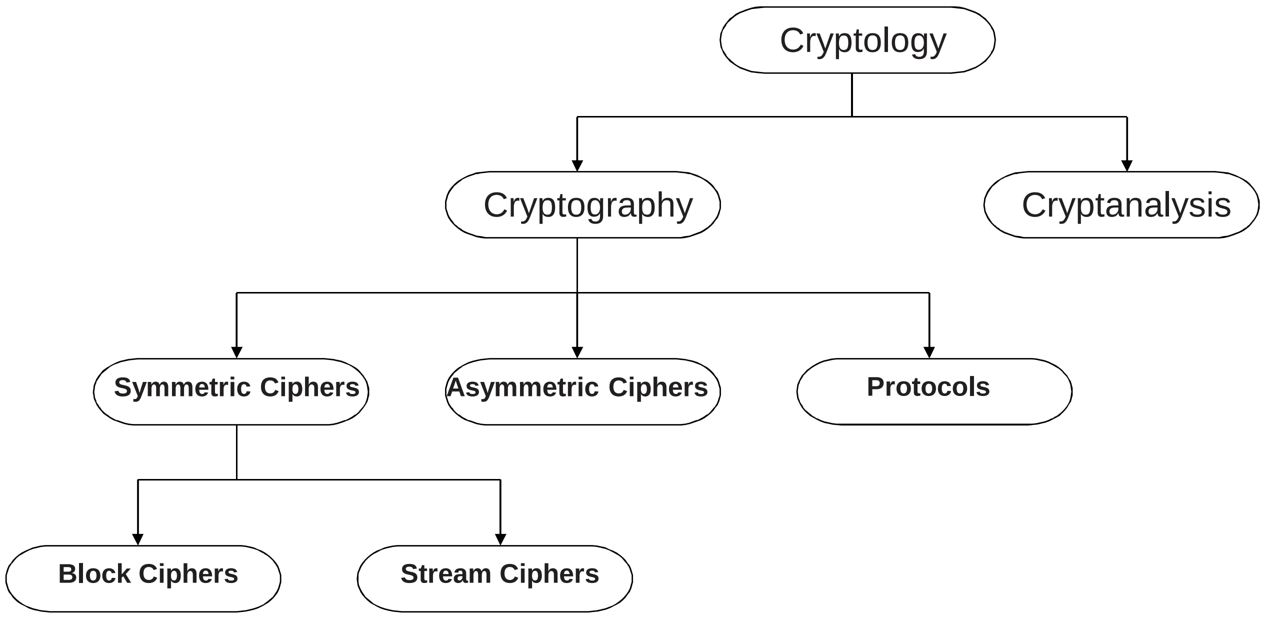
4) Porting the Library to as many languages and platforms as possible.

Cryptography

Cryptography is a science that deals with Rendering Information that is available for everyone to see,and make it such that only a few can understand.

Cryptography can be divided into a handful of Classifications.

Classification of the Field of Cryptology



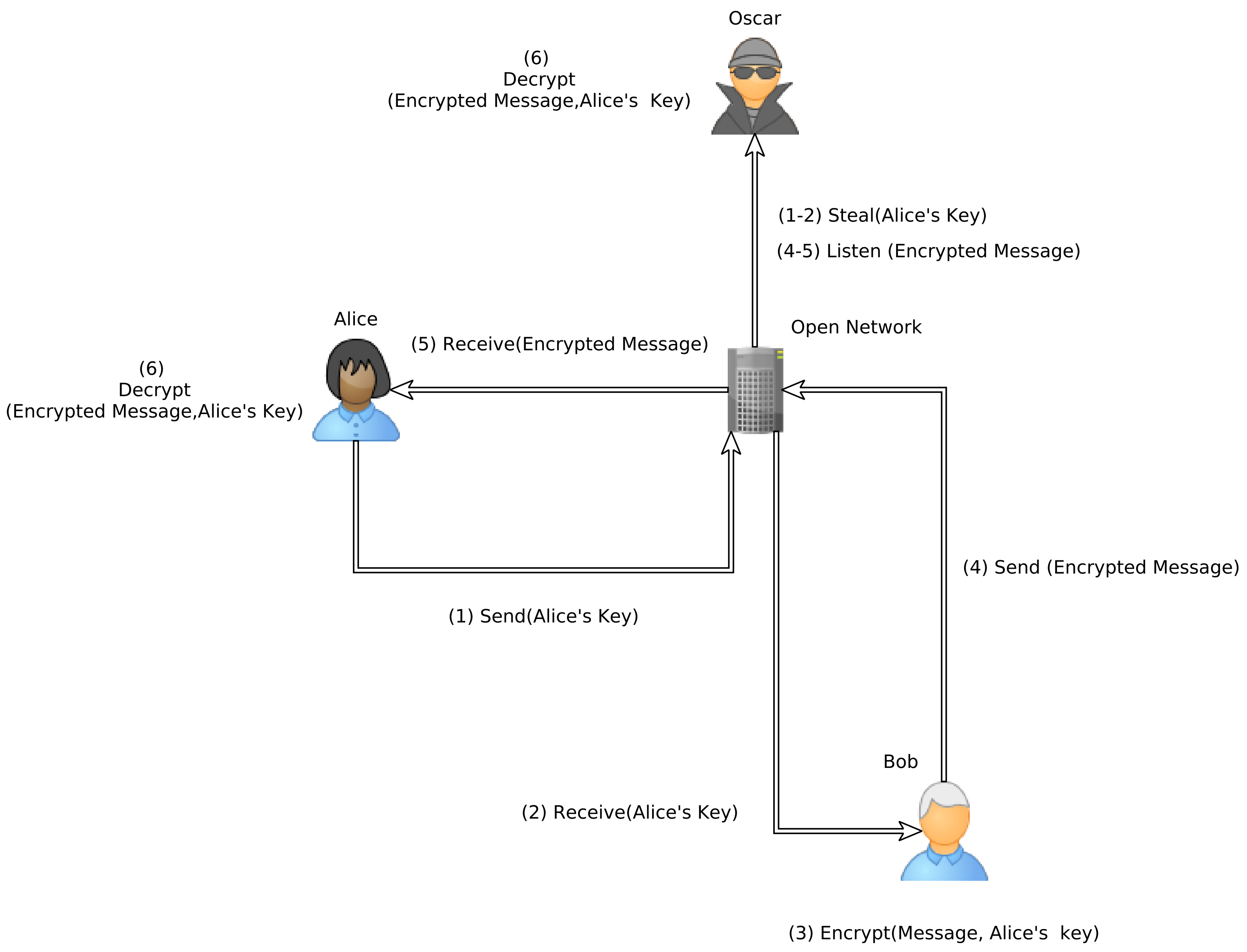
Symmetric Cryptography

Symmetric Cryptography is a classification in which a **Shared Key** is exchanged prior to Encryption.

The **Key** is used for both **Encryption and Decryption**.

The **Key** must be Exchanged over a **secure channel** or else if captured it will be used by a **Malicious Third Party** to decrypt the Encrypted message.

Symmetric Cryptography


Oscar
(6)
Decrypt
(Encrypted Message,Alice's Key)
(1-2) Steal(Alice's Key)
(4-5) Listen (Encrypted Message)
Alice
(5) Receive(Encrypted Message)
Open Network
(6)
Decrypt
(Encrypted Message,Alice's Key)
(4) Send (Encrypted Message)
(1) Send(Alice's Key)
Bob
(2) Receive(Alice's Key)
(3) Encrypt(Message, Alice's key)

Asymmetric Cryptography

This Encryption Classification Includes a **Pair of Keys (Public and Private)**.

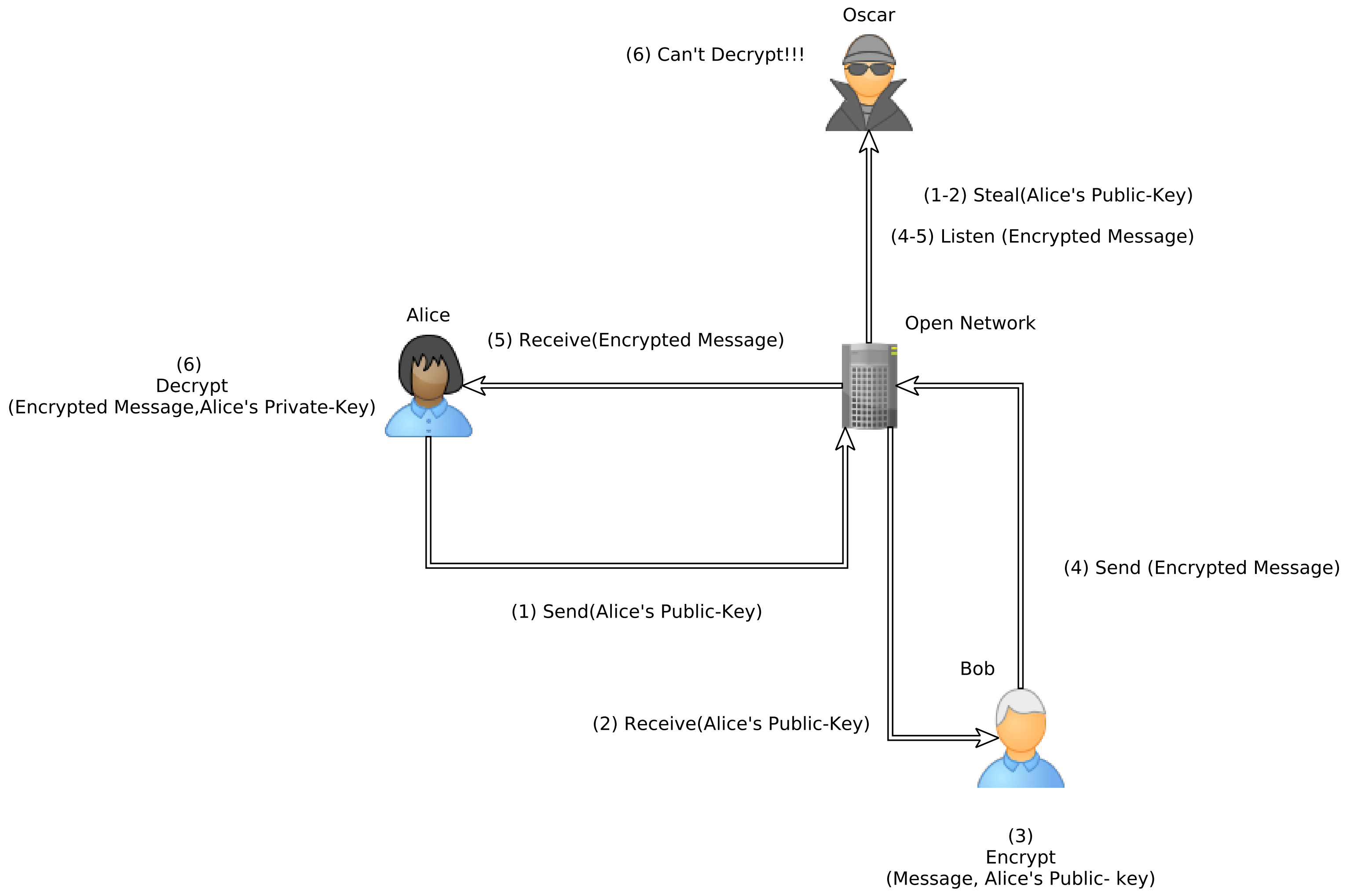
The Public Key is used for **Encryption** solely.

The Private Key is used for **Decryption** solely.

Only the **Public Key is Exchanged**.

If a Malicious Third Party Captures the Public Key, it is of no use to them **(it can't Decrypt)**.

ASymmetric Cryptography


Oscar
(6) Can't Decrypt!!!
(1-2) Steal(Alice's Public-Key)
(4-5) Listen (Encrypted Message)
Alice
(5) Receive(Encrypted Message)
Open Network
(6)
Decrypt
(Encrypted Message,Alice's Private-Key)
(4) Send (Encrypted Message)
(1) Send(Alice's Public-Key)
Bob
(2) Receive(Alice's Public-Key)
(3)
Encrypt
(Message, Alice's Public- key)

Mathematical Background

The RSA Algorithm utilizes a number of Well known mathematical Methods and Formulae to achieve a Successful and Secure Encryption/Decryption.

Those are :

1) The Euclidean Algorithm.

2) The Extended Euclidean Algorithm.

3) Euler's Phi Function.

4) Fermat's Little Theorem.

5) Euler's Theorem.

6) Binary Exponentiation (Square-and-Multiply).

7) Primality Testing.

The Euclidean Algorithm

This Algorithm Computes the Greatest Common Divisor of r_0 and r_1. $gcd(r_0, r_1)$.

It does so by following these simple steps :

1) Test if ($r_1 == 0$), if that's the case, then the final answer is the current r_0.

2) Make $Temp = r_1$

3) Make $r_1 = r_0 \bmod r_1$

4) Make $r_0 = Temp$

5) Repeat Recursively.

The Extended Euclidean Algorithm

Suppose $gcd(r_0, r_1) = 1$.

The theory states that you can write that as the following :

$$gcd(r0, r1) = s * r_0 + t * r_1$$

Just like the Euclidean Algorithm, we go on calculating the gcd recursively,making

$$r_i = r_{i-2} \text{ mod } r_{i-1}$$

$$q_{i-1} = (r_{i-2} - r_i)/r_{i-1}$$

$$t_i = t_{i-2} - q_{i-1} * t_{i-1}$$

until we reach :

$$gcd(r_0, r_1) \equiv 1$$

at that point $t = t_{i-1}$

now we submit the equation to the modulo operation :

$$gcd(r_0, r_1) \equiv 1 \equiv s * r_0 + t * r_1$$

$$1 \text{ mod } r_0 \equiv (s * r_0 + t * r_1) \text{ mod } r0$$

$$1 \text{ mod } r_0 \equiv t * r_1 \text{ mod } r_0$$

and since : $1 \text{ mod } r_0 \equiv r_1^{-1} * r_1 \text{ mod } r0$

then : $r_1^{-1} \equiv t$

And That is **One** way to calculate the **Modular Inverse**

Euler‘s Phi Function 1/2

- *New problem, important for public-key systems, e.g., RSA:*
  Given the set of the *m* integers {0, 1, 2, …, *m* -1},
  How many numbers in the set are **relatively prime to *m*** ?

- Answer: **Euler‘s Phi function *Φ(m)***

- **Example** for the sets {0,1,2,3,4,5} (*m*=6), and {0,1,2,3,4} (*m*=5)

$$\begin{aligned}\gcd(0,6)&=6\\ \gcd(1,6)&=1 \longleftarrow\\ \gcd(2,6)&=2\\ \gcd(3,6)&=3\\ \gcd(4,6)&=2\\ \gcd(5,6)&=1 \longleftarrow\end{aligned}$$

$$\begin{aligned}\gcd(0,5)&=5\\ \gcd(1,5)&=1 \longleftarrow\\ \gcd(2,5)&=1 \longleftarrow\\ \gcd(3,5)&=1 \longleftarrow\\ \gcd(4,5)&=1 \longleftarrow\end{aligned}$$

→ 1 and 5 relatively prime to *m*=6,
hence ***Φ*(6) = 2**

→ ***Φ*(5) = 4**

- Testing one gcd per number in the set is **extremely slow for large *m*.**

Euler‘s Phi Function 2/2

- **If** canonical factorization of *m* known**:** (where p_i primes and e_i positive integers)

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_n^{e_n}$$

- **then** calculate Phi according to the relation

$$\Phi(m) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i - 1})$$

- Phi especially easy for $e_i = 1$, e.g., $m = p \cdot q$ → $\Phi(m) = (p-1) \cdot (q-1)$

- **Example** $m = 899 = 29 \cdot 31$:
  ***Φ*(899)** = (29-1) · (31-1) = 28 · 30 **= 840**

- **Note:** Finding *Φ(m)* is computationally easy **if factorization of *m* is known**
  (otherwise the calculation of *Φ(m)* becomes computationally infeasible for large numbers)

Fermat‘s Little Theorem

- Given a **prime *p*** and an **integer *a***: $\boxed{a^p \equiv a \pmod{p}}$
- Can be rewritten as $a^{p-1} \equiv 1 \pmod{p}$

- **Use: Find modular inverse**, if p is prime. Rewrite to $a \cdot a^{p-2} \equiv 1 \pmod{p}$
- Comparing with definition of the modular inverse $a \cdot a^{-1} \equiv 1 \bmod m$

  → $a^{-1} \equiv a^{p-2} \pmod{p}$ is the modular inverse modulo a prime *p*

Example: *a* = 2, *p* = 7

$$a^{p-2} = 2^5 = 32 \equiv 4 \bmod 7$$

$$\text{verify: } 2 \cdot 4 \equiv 1 \bmod 7 \checkmark$$

- Fermat‘s Little Theorem works only **modulo a prime *p***

Euler‘s Theorem

- Generalization of Fermat‘s little theorem to **any integer modulus**
- Given two **relatively prime integers *a*** and ***m* :** $a^{\Phi(m)} \equiv 1 \ (\bmod\ m)$
- **Example**: *m*=12, *a*=5
  1. Calculate Euler‘s Phi Function

     $$\Phi(12) = \Phi(2^2 \cdot 3) = (2^2 - 2^1)(3^1 - 3^0) = (4-2)(3-1) = 4$$

  2. Verify Euler‘s Theorem

     $$5^{\Phi(12)} = 5^4 = 25^2 = 625 \equiv 1 \bmod\ 12$$

- Fermat‘s little theorem = special case of Euler‘s Theorem
- for a prime ***p***: $\Phi(p) = (p^1 - p^0) = p - 1$

  → Fermat: $a^{\Phi(p)} = a^{p-1} \equiv 1 \ (\bmod\ p)$

Square-and-Multiply

- **Basic principle**: Scan exponent bits from left to right and square/multiply operand accordingly

```
Algorithm: Square-and-Multiply for $x^H$ mod $n$
Input:  Exponent $H$, base element $x$, Modulus $n$
Output: $y = x^H \bmod n$
1.  Determine binary representation $H = (h_t, h_{t-1}, \dots, h_0)_2$
2.  FOR $i = t-1$ TO $0$
3.          $y = y^2 \bmod n$
4.      IF $h_i = 1$ THEN
5.          $y = y * x \bmod n$
6.  RETURN $y$
```

- Rule: Square in every iteration (Step 3) and multiply current result by x if the exponent bit $h_i = 1$ *(Step 5)*
- Modulo reduction after each step keeps the operand y small

Example: Square-and-Multiply

- Computes x^{26} without modulo reduction
- Binary representation of exponent: $26 = (1,1,0,1,0)_2 = (h_4,h_3,h_2,h_1,h_0)_2$

Step		Binary exponent	Op	Comment
1	$x = x^1$	$(1)_2$		Initial setting, h_4 processed
1a	$(x^1)^2 = x^2$	$(10)_2$	SQ	Processing h_3
1b	$x^2 * x = x^3$	$(11)_2$	MUL	$h_3 = 1$
2a	$(x^3)^2 = x^6$	$(110)_2$	SQ	Processing h_2
2b	-	$(110)_2$	-	$h_0 = 0$
3a	$(x^6)^2 = x^{12}$	$(1100)_2$	SQ	Processing h_1
3b	$x^{12} * x = x^{13}$	$(1101)_2$	MUL	$h_1=1$
4a	$(x^{13})^2 = x^{26}$	$(11010)_2$	SQ	Processing h_0
4b	-	$(11010)_2$	-	$h_0 = 0$

- Observe how the exponent evolves into $x^{26} = x^{11010}$

Primality Testing

There are many ways to determine whether integer P is prime or not.

One way is we make sure P is prime, By factorization.

The other is depending on **Equivalence Rules** that only apply to primes. and Testing whether these Rules apply to P or not.

If They do, then P is **probably** Prime.

If They don't, then P is **definitely** Composite.

Using Sieves for Factorization

Sieves are a great way to Shrink the amount of time needed to test the primality of **Many or More** test cases.

A Sieve is a Boolean Array, that uses the Index to signal the number we're storing, and uses the boolean value to signal Primality (1 being Prime, 0 being Composite)

The Array is initialized with 1. as in all numbers are initially considered Primes.

Then we start going through numbers (indexes).

for each index that has a value of 1, do the following:

go through all the indexes that are multiples of that prime index, and switch their values to 0.

Sieve Tricks

Consider the Following :

1) For a given Prime index, all multiples of that index up until $index^2$ are switched to composites (value to 0), by the **Previous Prime** index **Passes**.

Therefor : only switch the multiples of a Prime index that are above $index^2$.

2) All Prime indexes needed to determine the primality of an index are below $\sqrt[2]{index}$.

Therefor : keep going through prime indexes until index = $\sqrt[2]{Array_{limit}}$.

Prime Rules

Consider a Prime number P, it can be written as : $P = (r_1 * r_2 * .... * r_i) + 1$

where : $r_1 .... r_i$ is a factor of $P - 1$.

Take a random number a where :

$a > 1$ and $a < P \Rightarrow \text{gcd}(P, a) = 1$.

now find the result of this equation :

$$a^{P-1} =? \text{ mod } P \ .$$

$$a^{(r_1 * r_2 * .... * r_i)} =? \text{ mod } P \ .$$

$$((a^{r_1})^{r_2})^{r_i} =? \text{ mod } P$$

with $answer = (a^{r_i} \text{ mod } P) \in [1 ... P - 1]$

We've got P-1 multiplications, and each multiplication produces a different $answer$ (since the modulus operation is cyclic and gcd$(P, a) = 1$) until we reach $a^{r_i} = 1 \bmod P$

at that point $1^{r_i} \equiv 1 \bmod P$

The above proves **Fermat's Little Theorem**.

It makes it that if a given number P passes the above Equivalence Rule, then it's a **Probable Prime**.

But also if we consider a^P, by a few steps of mathematical manipulation we derive that :

$$a^P \equiv a \bmod P$$

if we multiply both sides by a^{-2} :

$$a^P * a^{-2} \equiv a * a^{-2} \bmod P$$

$$a^{P-2} \equiv a^{-1} \bmod P$$

and that is a **Second** way to calculate the

Modular Inverse .

The RSA Cryptosystem

- Martin Hellman and Whitfield Diffie published their landmark public-key paper in 1976
- Ronald Rivest, Adi Shamir and Leonard Adleman proposed the asymmetric RSA cryptosystem in1977
- Until now, RSA is the most widely use asymmetric cryptosystem although elliptic curve cryptography (ECC) becomes increasingly popular
- RSA is mainly used for two applications
  - Transport of (i.e., symmetric) keys (cf. Chptr 13 of *Understanding Cryptography*)
  - Digital signatures (cf. Chptr 10 of *Understanding Cryptography*)

Encryption and Decryption

- RSA operations are done over the integer ring Z_n (i.e., arithmetic modulo n), where $n = p * q$, with p, q being large primes
- Encryption and decryption are simply exponentiations in the ring

Definition

Given the public key $(n,e) = k_{pub}$ and the private key $d = k_{pr}$ we write

$$y = e_{k_{pub}}(x) \equiv x^e \bmod n$$

$$x = d_{k_{pr}}(y) \equiv y^d \bmod n$$

where x, y ε $Z_{n.}$

We call $e_{k_{pub}}()$ the encryption and $d_{k_{pr}}()$ the decryption operation.

- In practice x, y, n and d are very long integer numbers (≥ 1024 bits)
- The security of the scheme relies on the fact that it is hard to derive the „private exponent“ d given the public-key (n, e)

Key Generation

- Like all asymmetric schemes, RSA has set-up phase during which the private and public keys are computed

Algorithm: RSA Key Generation

Output: public key: $k_{pub} = (n, e)$ and private key $k_{pr} = d$

```
1.  Choose two large primes $p, q$
2.  Compute $n = p * q$
3.  Compute $\Phi(n) = (p-1) * (q-1)$
4.  Select the public exponent $e \in \{1, 2, \ldots, \Phi(n)-1\}$ such that
    gcd$(e, \Phi(n)) = 1$
5.  Compute the private key $d$ such that $d * e \equiv 1 \bmod \Phi(n)$
6.  RETURN $k_{pub} = (n, e), k_{pr} = d$
```

Remarks:

- Choosing two large, distinct primes p, q (in Step 1) is non-trivial
- gcd$(e, \Phi(n)) = 1$ ensures that e has an inverse and, thus, that there is always a private key d

Why the RSA is Secure!

1) To calculate the private Key Efficiently you need to use the Extended Euclidean Algorithm on the e and $\phi(n)$.

2) to Calculate $\phi(n)$ you need to have p and q.

3) The only way to get p and q from a non-secure channel is by getting n (which is public).

4) Factoring n proves Impractical for (> 1024) bit Integers ,by today's standard computing power.

RSA's Shortcomings

1) When choosing a relatively small Exponent e as a Public-Key, if $message^e < modulus$, then by deduction :

$$message^e \equiv message^e \text{ mod } modulus.$$

And the message can be regenerated using $message = \sqrt[e]{message^e}$

Therefor : Avoid Relatively Small numbers when choosing a Public-Key.

2) Attackers can Encrypt a Predicted Plain text with the Public-Key, and Compare it to the Original Encrypted text.

Solution : **Padding** the Text Message prior to Encryption with Random Values.

EnDaBi Implementations

These Implementations include :

. The EnDaBi RSA Core Library.
. a Console Demo showcasing the Library's main functions.
. a GUI Demo showcasing the Library's main functions.
. an Example Segmented Sieve Program
. a Small java program that utilizes built-in primality testing.
. a make file used to compile the demos.

```
/*
Copyright 2015 ALY SHMAHELL

 This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/

#ifndef ENDABI_RSA_CORE_INCLUDED
#define ENDABI_RSA_CORE_INCLUDED
#include <stdio.h>
#include <string.h>
#include <sstream>
using namespace std;
#define WORD 200

/* This struct is used to store the public key sequence and return it as a whole aft
er Public-Key generation
The sequence is : the suitable Public-Key, the corrected Prime p, the corrected Prim
e q, the Modulus n = p*q  */
template <typename typename_> struct pub_key_sequence
{
    typename_ pk, p_, q_, n;
};

/* The phi(modulus) function denotes the number of coprimes the encryption modulus (
modulus = n) has */
template <typename typename_> typename_ phi(typename_ p,typename_ q)
{
    return ((p-1)*(q-1));
}

/* The Greatest Common Divisor function building on the Eucledian Algorithm */
template <typename typename_> typename_ gcd(typename_ r0,typename_ r1)
{
    return (r1==0)?r0:gcd(r1,r0%r1);
}

/* Modular Exponentiation function, using the famous & fast (square and multiply)  m
ethod */
template <typename typename_>
typename_ modular_exponentiation(typename_ base,typename_ exponent,typename_ modulus
_)
{
    typename_ temp;
    if(exponent==1)
        return base;
    else
    {
        if(exponent%2==0)
        {
            temp = modular_exponentiation(base,exponent/2,modulus_);
            return ((temp*temp)%modulus_);
        }
        else
        {
            temp = modular_exponentiation(base,(exponent-1)/2,modulus_);
            temp*=temp;
            temp*=base;
            return (temp%modulus_);
        }
    }
}
```

```
/* This function finds the Modular Inverse using the Extended Eucledian Algorithm.
The modular Inverse is described as  : (a*a^-1 = 1 mod modulus) where (a^-1) is
the inverse of a.  */
template <typename typename_> typename_ EEA(typename_ r0,typename_ r1)
{
    typename_ t;
    typename_ t0=(typename_)0;
    typename_ t1=(typename_)1;
    typename_ temp;
    typename_ r2,r3;
    if(r0<r1)
    {
        temp=r0;
        r0=r1;
        r1=temp;
    }
    r2=r0;
    r3=r1;
    while(r3!=0)
    {
        if(r0>r1)
        {
            t=t0-((r2-(r2-r3*(r2/r3)))/r3)*t1;
            t0=t1;
            t1=t;
        }
        temp = r3;
        r3 = (r2-r3*(r2/r3));
        r2 = temp;
    }
    if(t0<0) t0+=r0;
    return t0;
}

/* a Wrapper for the previous function, this one finds the inverse of the Public-Key
 over Phi(modulus) */
template <typename typename_> typename_ modular_inverse(typename_ a,typename_ p,type
name_ q)
{
    typename_ phi_=phi(p,q);
    return EEA(a,phi_);
}

/*this function encrypts a message character by exponentiating the ascii representat
ion of the character
to the power public-key and reducing the result modulo modulus */
template <typename typename_>
typename_ encrypt(typename_ message,typename_ public_key,typename_ modulus_)
{
    return modular_exponentiation(message,public_key,modulus_);
}

/*this function encrypts a message character by exponentiating the ascii representat
ion
of the character to the power private-key and reducing the result modulo modulus */
template <typename typename_>
typename_ decrypt(typename_ encrypted_message,typename_ private_key, typename_ modul
us_)
{
    return modular_exponentiation(encrypted_message,private_key,modulus_);
}

/*this is a call function that opens a pipe to an external java application that uti
lizes
a built-in java function that verifies the primality of line2 with accuracy line3 */
template <typename typename_> int isprime(typename_ line2, int line3)
{
    FILE *fp;
    int status;
    char prime[WORD];
    string line1="java -classpath 3rd_party isProbablePrime ";
    stringstream line,totalline;
    totalline<<line1;
    totalline<<line2;
    totalline<<" ";
    totalline<<line3;
    string order = totalline.str();
    fp = popen(order.c_str(),"r");
    fgets(prime, WORD, fp);
    return (prime[0]-'0');
    status = pclose(fp);
}

/*this function decreases a given number (wanted) until that number is a prime*/
template <typename typename_> typename_ round_to_prime(typename_ wanted)
{
    while(!isprime(wanted,4))
        wanted--;
    return wanted;
}

/* a function that generates a proper public-key that satisfies the RSA constraint g
cd(public-key,phi(modulus))=1. */
template <typename typename_>
pub_key_sequence<typename_> generate_public_key(typename_ pub, typename_ p,typename_
 q)
{
    p=round_to_prime(p);
    q=round_to_prime(q);
    while((pub!=0)&&((!isprime(pub,4))||(gcd(pub,phi(p,q))!=1)))
        pub--;
    pub_key_sequence<typename_> result = {pub,p,q,(typename_)(p*q)};
    return result;
}

/* this is a wrapper function for the modular_inverse function that initializes the
previous with the proper paramiters */
template <typename typename_> typename_ calculate_private_key(typename_ public_key,t
ypename_ p,typename_ q)
{
    typename_ pub = public_key;
    typename_ temp_p = p;
    typename_ temp_q = q;
    typename_ private_key = modular_inverse(pub,temp_p,temp_q);
    return private_key;
}
#endif
```

```
/*
Copyright 2015 ALY SHMAHELL

 This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
#include <iostream>
#include <string.h>
#include <sstream>
#include <stdio.h>
#include <stdint.h>
#include "ENDABI_RSA_CORE.hpp"
#include "3rd_party/InfInt.h"

using namespace std;
int main()
{
    InfInt desired_pk,desired_p,desired_q;
    cout<<"please input the desired public-key sequence \n(desired public-key, desir
ed p, desired q) and wait for us to make the right \ncorrections and generate a suit
able public-key\n>> ";
    cin>>desired_pk>>desired_p>>desired_q;
    pub_key_sequence<InfInt> pks = generate_public_key(desired_pk,desired_p,desired_
q);
    stringstream pks_string;
    pks_string<<"public-key sequence : pk: "<<pks.pk<<"  n:  "<<pks.n;
    cout<<pks_string.str()<<endl;
    cout<<"do you want a private key? Y or N\n";
    string answer;
    cin>>answer;
    if(answer=="Y"||answer=="y")
    {
        InfInt prvk = calculate_private_key(pks.pk,pks.p_,pks.q_);
        stringstream prvks_string;
        prvks_string<<"private-key : "<<prvk<<" n: "<<pks.n;
        cout<<prvks_string.str()<<endl;
        printf("Do you want to encrypt a numerical message? \nType either Y or N.\n>
> ");
        cin>>answer;
        if(answer=="Y"||answer=="y")
        {
        printf("Good, now we can continue,\nIf typed Y please input the desired mess
age (max char count is 200) :\n>> ");
            char message [200];
            InfInt plain_numerical[200];
            InfInt encrypted_numerical[200];
            InfInt decrypted_numerical[200];
            cin.ignore();
            gets(message);
            int mssg_length = strlen(message);
            for(int i =0; i<mssg_length; i++)
            {
                plain_numerical[i]=message[i]-'0';
                encrypted_numerical[i]=encrypt((InfInt)plain_numerical[i],pks.pk,pks
.n);

                cout<<encrypted_numerical[i];
            }
            printf("\n");
            cout<<"Now, do you want to decrypt the message?\n>> ";
                cin>>answer;
            if(answer=="Y"||answer=="y")
            {
                for(int i =0; i<mssg_length; i++)
                {
                    decrypted_numerical[i]=decrypt(encrypted_numerical[i],prvk,pks.n
);
                    cout<<decrypted_numerical[i];
                }
                printf("\n");
            }
            else
            {
                printf("Well, there's nothing more we can do for you then.\nHappy in
securty :)\n");
                return 0;
            }
        }
        else
        {
            printf("Well, there's nothing more we can do for you then.\nHappy insecu
rty :)\n");
            return 0;
        }
    }
    else
    {
        printf("Well, there's nothing more we can do for you then.\nHappy insecurty
:)\n");
        return 0;
    }

}
```

```
/*
Copyright 2015 ALY SHMAHELL

 This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/
#include <iostream>
#include <sstream>
#include <stdio.h>
#include <FL/Fl.H>
#include <FL/Fl_Window.H>
#include <FL/Fl_Button.H>
#include <FL/Fl_Input.H>
#include <FL/Fl_Output.H>
#include <FL/Fl_Box.H>
#include <FL/Fl_Text_Display.H>
#include <string>
#include <string.h>
#include <stdint.h>
#include "ENDABI_RSA_CORE.hpp"
#include "3rd_party/InfInt.h"
using namespace std;

/*Global Widgets*/

//Define Buffers
Fl_Text_Buffer *pubbuff = 0;
Fl_Text_Buffer *modbuff = 0;
Fl_Text_Buffer *prvbuff = 0;
Fl_Text_Buffer *encbuff = 0;
Fl_Text_Buffer *decbuff = 0;

//Define Display Boxes
Fl_Text_Display *private_key = 0;
Fl_Text_Display *public_key = 0;
Fl_Text_Display *Modulus = 0;
Fl_Text_Display *encrypted_message = 0;
Fl_Text_Display *decrypted_message = 0;

//Define Input Text Boxes
Fl_Input *desired_pubk = 0;
Fl_Input *d_p =0;
Fl_Input *d_q =0;
Fl_Input *mssg =0;

/**/

//global variables
InfInt desired_pk,desired_p,desired_q;
pub_key_sequence<InfInt> pks = {(InfInt)0,(InfInt)0,(InfInt)0,(InfInt)0};
int mssg_length;
InfInt encrypted_numerical[200];
InfInt decrypted_numerical[200];
InfInt prvk;

//Callback Function to generate both Public & Private Keys
void gen_keys(Fl_Widget*,void*)
{
    //Input
    desired_pk=(InfInt)(desired_pubk->value());
    desired_p =(InfInt)(d_p->value());
    desired_q =(InfInt)(d_q->value());
    //Public Key Generation
    pks = generate_public_key(desired_pk,desired_p,desired_q);
    //Private Key Calculation
    prvk = calculate_private_key(pks.pk,pks.p_,pks.q_);
    //Output Public Key
    {
        stringstream pks_strings;
        pks_strings<<pks.pk;
        string pks_string = pks_strings.str();
        pubbuff->text(pks_string.c_str());
        public_key->buffer(pubbuff);
    }
    //Output Modulus
    {
        stringstream pks_strings;
        pks_strings<<pks.n;
        string pks_string = pks_strings.str();
        modbuff->text(pks_string.c_str());
        Modulus->buffer(modbuff);
    }
    //Output Private Key
    {
        stringstream prvks_strings;
        prvks_strings<<prvk;
        string prvks_string = prvks_strings.str();
        prvbuff->text(prvks_string.c_str());
        private_key->buffer(prvbuff);
    }
}

/*Callback Function to Encrypt a message in the message Text Box
onto the Encrypted Message Display Box*/
void enc_mssg(Fl_Widget*,void*)
{
    //Input
    stringstream msgs;
    msgs<<mssg->value();
    string msg = msgs.str();
    const char* message = msg.c_str();
    //variables
    InfInt plain_numerical[200];
    mssg_length = strlen(message);
    stringstream encrypted_msgs;
    //Encryption
    for(int i =0; i<mssg_length; i++)
    {
        plain_numerical[i]=message[i]-'0';
        encrypted_numerical[i]=encrypt((InfInt)plain_numerical[i],pks.pk,pks.n);
        encrypted_msgs<<encrypted_numerical[i];
    }
    //Output
    string encrypted_msg = encrypted_msgs.str();
    encbuff->text(encrypted_msg.c_str());
    encrypted_message->buffer(encbuff);
}

/*Callback Function to Decrypt a message in the Encrypted Message Display Box
onto the Decrypted Message Display Box*/
void dec_mssg(Fl_Widget*,void*)
{
    stringstream decrypted_msgs;
    for(int i =0; i<mssg_length; i++)
    {
        decrypted_numerical[i]=decrypt(encrypted_numerical[i],prvk,pks.n);
        decrypted_msgs<<decrypted_numerical[i];
    }
    string decrypted_msg = decrypted_msgs.str();
    decbuff->text(decrypted_msg.c_str());
    decrypted_message->buffer(decbuff);
}

int main()
{
    //Initialize a Window
    Fl_Window *win =  new Fl_Window(990, 680, "ENDABI_RSA_DEMO_GUI");
    //Initialize Buffers
    pubbuff =  new Fl_Text_Buffer();
    modbuff =  new Fl_Text_Buffer();
    prvbuff =  new Fl_Text_Buffer();
    encbuff =  new Fl_Text_Buffer();
    decbuff =  new Fl_Text_Buffer();
    //Initialize Input Text Boxess
    desired_pubk= new Fl_Input(150,10,800,30,"Desired Public Key");
    d_p= new Fl_Input(150,50,800,30,"Desired p");
    d_q = new Fl_Input(150,90,800,30,"Desired q");
    mssg = new Fl_Input(450,370,200,30,"Message");
    //Initialize Display Boxes
    public_key = new Fl_Text_Display(150,200,800,30,"Public-Key");
    Modulus = new Fl_Text_Display(150,260,800,30,"Modulus");
    private_key = new Fl_Text_Display(150,310,800,30,"Private-Key");
    encrypted_message = new Fl_Text_Display(150,430,800,130,"Encrypted Message");
    decrypted_message = new Fl_Text_Display(450,640,200,30,"Decrypted Message");
    //Define & Initialize Buttons
    Fl_Button *generate_keys = new Fl_Button(480,140,150,30,"Generate Keys");
    Fl_Button *encrypt_message = new Fl_Button(150,370,150,30,"Encrypt Message");
    Fl_Button *decrypt_message = new Fl_Button(470,580,150,30,"Decrypt Message");
    //Link Buttons to callback functions
    generate_keys->callback(gen_keys);
    encrypt_message->callback(enc_mssg);
    decrypt_message->callback(dec_mssg);
    //Run Application
    win->show();
    //Return
    return(Fl::run());
}
```

```
/*
Copyright 2015 ALY SHMAHELL

 This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Lesser General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
*/

import std.stdio, std.bigint, std.string, std.conv, std.stream,std.datetime;

/* this simple program finds all primes between one given number b
passed to the program from the terminal as an argument, and
the value a=(b-999)  */

void main(string g[])
{
      //Input & Define Range
      BigInt b=g[1];
      BigInt a=(b-999);

        if(a==1) a++;

        //Define Sieve
        BigInt p[1000];

        //Initialize Sieve
        for(ulong i=0;i<=999;i++)
         p[i]=1;

        //Start Sieving
        for(BigInt i=2;i*i<=b;i++)
        {
            BigInt k=a/i;
            k*=i;
            for(BigInt j=k;j<=b;j+=i)
              if(j!=i&&j>=a)
                {
                 BigInt m=j-a;
                 ulong n=0;
                 for(BigInt o=0;o<m;o++)
                  n++;
                  p[n]=0;
                }
        }

        //Output
        for(ulong i=0;i<=999;i++)
        if(p[i]==1)
         writeln(a+i);
}
```

```java
import java.math.BigInteger;

public class isProbablePrime
{
  public static void main(String[] args)
  {
    BigInteger test_case = new BigInteger(args[0]);
    int certainty = Integer.parseInt(args[1]);

/* 1 means Probably Prime, 0 means definitely composite. */
    System.out.println((test_case.isProbablePrime(certainty) ? "1" : "0"));
  }
}
```

```
#This is the makefile used by the make utility to compile the EnDaBi Demos
ENDABI_RSA_DEMO_GUI: ENDABI_RSA_CORE.hpp 3rd_party/isProbablePrime.java 3rd_party/In
fInt.h
        javac 3rd_party/isProbablePrime.java
        gdc segmented_sieve.d -o segmented_sieve
        fltk-config --compile ENDABI_RSA_DEMO_GUI.cxx
        g++ ENDABI_RSA_DEMO.cpp -o ENDABI_RSA_DEMO
```

Toolkits, Libraries and Programming Languages We Used

C++ : Strongly typed, Fast and efficient, library extended Programming-Language.

D : Strongly typed, Fast and efficient Programming-Language with Syntax Similar to that of C and Java.

Java : Strongly typed, byte-code compiled programming language that devotes to binary portability (compile once, run everywhere).

FLTK : Fast Light Tool Kit ("FLTK", pronounced "fulltick") is a C++ graphical user interface toolkit for the X Window System, MacOS, and Microsoft Windows that supports OpenGL.

ENDABI RSA DEMO GUI is based in part on the work of the FLTK project `http://www.fltk.org`.

LaTeX : is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is the de-facto standard for the communication and publication of scientific documents.

Software we Used

Ubuntu 14.04 LTS : Free and Open Source Linux-Based Operating System.

GCC : GNU project C and C++ compiler.

GDC : A GCC-based compiler for the D language.

Javac : Java programming language compiler.

TeXstudio : is a LaTeX editor with a graphical user interface.

Code::Blocks : The open-source, cross-platform IDE.

SciTE : a programmers text editor

Vim : is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs.

Eclipse : extensible tool platform and Java IDE.

nano : is a small, free and friendly editor which aims to replace Pico, the default editor included in the non-free Pine package.

make : GNU make utility to maintain groups of programs

Git : is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to internals.

yEd : is a powerful desktop application that can be used to quickly and effectively generate high-quality diagrams.

How To Use Our Software

1) This Installation process was tested on Ubuntu 14.04 LTS.

2) **Install Prerequisites**

In Terminal, Type :

```
sudo apt-get install gcc g++ gdc openjdk-7-jdk
vim libfltk1.3-dev fltk1.3 fltk1.3-doc
build-essential make git
```

3) **Pull EnDaBi Sources from Github**

In Terminal, Type :

```
git clone https://github.com/EnDaBi/EnDaBi.git
```

4) **Navigate to the EnDaBi Folder**

In Terminal, Type :

```
cd EnDaBi
```

5) **Compile the Sources**

In Terminal, Type :

```
make
```

6) **Run Demos**

In Terminal, Type :

```
./ENDABI_RSA_DEMO_GUI
```

```
or
```

```
./ENDABI_RSA_DEMO
```

What’s Next

Regarding the RSA Core :

1) Implementing the Chinese Remainder Theorem for Faster Private Key Exponentiation (Decryption).

2) Implementing Our Own BigInteger Library.

3) Implementing our Own Padding System.

4) Implementing Our own Primality Testing Classes.

Regarding the Project :

1) Adding more Encryption Schemes.

2) Developing Database Classes.

3) Developing Biometrics Classes.

References

[1] Paar, C., and Pelzl, J.
Understanding Cryptography
A Textbook for Students and Practitioners.
2010.
www.crypto-textbook.com
Prof. Dr.-Ing. Christof Paar
Chair for Embedded Security
Ruhr-Universitat Bochum
D - 44780 Bochum.

Team EnDaBi