# Ensuring Authentication and Integrity of Open Source Software using Digital Signature

M. Tariq Banday
P.G. Department of Electronics and Instrumentation Technology
University of Kashmir, India

## ABSTRACT
A group of programmers participate in the development of Open Source Software and its source code is publically made available for review, reporting, fixing bugs and enhancing its functionalities. The Open Source Software, its patches and new releases are made available to users through multiple hosts on the Internet and by distribution on media like on CD's and DVD's. A hacker may modify the software and incorporate virus, spyware, adware or other similar routines into it that may lead to manifold of security breaches. It is thus essential to ensure authenticity and integrity of the Open Source Software before compiling and installing it to avoid falling prey to any such possible security breach. This paper discussesmethods for attaining authentication and integrity of Open Source Software for the purpose of its distribution.

## General Terms
Open Source Software, Software Distribution, Cryptography, PKI, Authentication, Free Software, PAIN.

## Keywords
Digital Signature Certificate, Digital Signature, Privacy, Authentication, Integrity, Non-repudiation, Open Source Software, OSS.

## 1. INTRODUCTION
An Open Source Software (OSS) is software for which the human readable source code is available for use, study, reuse, modification, enhancement and distribution by the users of that software. Contrary to OSS, most Commercial Software (CS) does not allow users others than the developers to view the source code. Instead only compiled object code is distributed. The source code is the structured and modularized representation of the software's functionality written in a programming language targeted for understanding and changeability by humans. Language translators e.g. compilers are used to translate source code into object which is machine-oriented and therefore very hard to read and understand by humans. In comparison to CS whose source code is closed and not modifiable by hacker, OSS's source code is available for modification and can be modified by the hacker for illegitimate purposes. It is thus essential to ensure integrity of the OSS and the authentication of distributor to avoid security lapses. Public Key Infrastructure (PKI) ensures privacy, integrity, authentication and non- repudiation and thus can be used for the distribution of Open Source Software to public.

The remaining paper is organized as follows: section 2 presents brief history and definition of OSS and section 3 portrays four key factors to achieve information security. It also describes their importance for OSS distribution. Section 4 demonstrates the application of digital signature to ensure authentication, integrity and non-repudiation of OSS. Section 5 briefly introduces *CrypTool*, an OSS to learn and practice cryptographic mechanisms which is followed by conclusion.

## 2. OPEN SOURCE SOFTWARE
Open Source Software originated in early 1970's [1] and evolved from programming hobby to a mainstream commercial strategy for acquiring and maintaining competitive advantage by 1990's [2]. The first collaborative effort towards OSS was GNU (GNU is not UNIX) project announced by Stallman in 1983 to create free operating system to replace UNIX. The establishment of Free Software Foundation (FSF) [3] in 1985 was another collaborative effort for development of Free and OSS. Open Source Initiative (OSI) [4] was established in 1998 to promote OSS on pragmatic rather than ethical or philosophical grounds. The OSI has become steward of the Open Source Definition (OSD). OSD has laid down ten criteria [5] for software to be qualified as Open Source Software. These are: i) free redistribution of the software, ii) source code to be distributed with the software or well published access to it, iii) derived works arising from modification of the software should be licensed to be distributed, iv) integrity of the author's source code, v) no discrimination against a person or a group, vi) no discrimination against fields of endeavor, vii) distribution of license, viii) license must not be specific to a product, ix) license must not restrict other software, and x) license must be technologically neutral. For software complying with these criteria, the OSI introduced the seal 'OSI Certified'. Considered as open source in a broader sense are license models that differ, for instance, on the extent to which changes made to an Open Source program have to comply with the same license as the original program, or on the extent to which restrictions on the license to use the program are allowed. Open source software is generally thought to be free as it has no costs. Though that is true in some cases but generally the term "free" is used in reference to the liberty of interested parties to freely distribute the source code. The definition of free software, as enunciated by the FSF, states that four essential freedoms should be contained within its license: i) to run the software for any purpose, ii) to study how the software works and adapt it, iii) to redistribute copies of the software, and, iv) to improve the software, and release those improvements.

## 3. PAIN
PAIN which collectively refers to Privacy, Authentication, Integrity and Non-repudiation are four key factors to achieve information security [6]. Privacy also called confidentiality, guarantees non-disclosure of information to unauthorized

persons. Authentication ensures that the document or software is genuine. Integrity as a concept means that there is resistance to alteration or substitution of data, and/or that such changes are detected and provable. The information should not be changed except by an authorized agent. Non-repudiation is a security service that prevents a party from falsely denying that it was the source of data that is did indeed create. Privacy is not considered for OSS distribution as its distribution is made to public and not to a particular individual, however, authentication, integrity and non- repudiation are essential parameters to attain high degree of reliability and security in Open Source Software distribution.

Authentication is the security process that validates the identity of a communicating party. In the simplest implementation, this takes the form of a password. Passwords can be easily compromised through indiscretion and typically do not address who is entering the password. Another variant of authentication is known as strong authentication. In this implementation, authentication is provided by a digital signature which is an encrypted value provided by the entity requesting authentication that can only be decoded by the public key of the signature's owner. Authentication is used to ensure that software downloaded from the Internet comes from a reputable source.

Integrity ensures security against forgery which include policies to stop distribution of software contaminated with viruses, Trojans, Spywares, etc. This usually involves the use of checksums, one-way hashes, or other algorithmic validation of the data. Whether the data might be changed by accident or malice, preventing that change is the foremost concern, and detecting it is second. Integrity can be maintained at many levels, from the hardware all the way to the application logic. At first glance it might seem that authenticity is included in the concept of Integrity. Integrity is more specifically about the content of the data itself. Integrity of OSS downloaded from the Internet can be obtained using digital signature.

Non-repudiation ensures non-denial by the sender ensuring that an OSS distributor will not be able to disown its software distribution later. Digitally signing of the OSS using the private key of the distributor can ensure non-repudiation as breaking of digital signature is harder than any traditional method.

## 4. DISTRIBUTING OSS USING DIGITAL SIGNATURE

A digital signature is identification information encrypted with a private key and therefore decryptable with the corresponding public key [7]. Digital signature technology requires the use of public key cryptography, that is, the use of public and private key pairs. The process of digitally signing OSS is illustrated in figure 1.

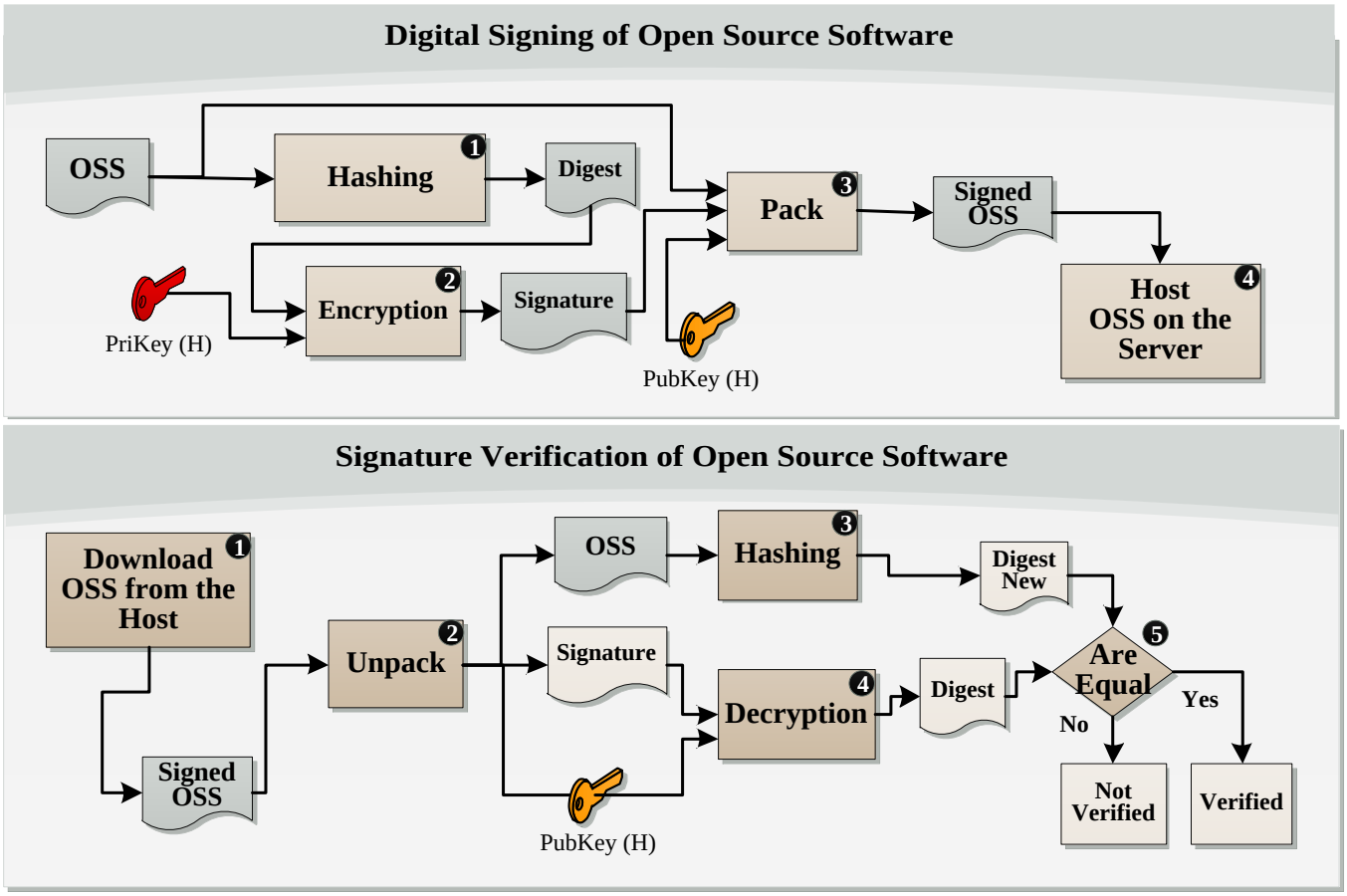


**Figure 1: Digital Signing and Verification of OSS**

Once the publishable release of an OSS is ready, it would be deposited on the host which would digitally sign the software by computing a hash (or checksum) over the source code creating a digest and then encrypt the digest using its private key. The encrypted digest would be made available to the user as an accompaniment to the OSS. The user would verify the authenticity of the OSS by recalculating the digest, decrypting the transmitted digest using the host's public key (which had been previously obtained and cached, or was obtained from a public key directory or is included with the OSS) and comparing the two digests. If they match, then the OSS is authentic and came from the claimed host, has not been modified since signing. Various steps involved in signing and verifying the signature are given hereunder:

***A) Digitally Signing of Open Source Software***

1. **Hashing**: Hashing is a one way encryption function used to calculate OSS digest, which is small and unique representation of the OSS. The purpose of evaluating a digest is to ensure OSS integrity. The digital signature is applied on the OSS digest which is smaller than the software itself. Further, hashing functions are faster than symmetric and asymmetric encryption algorithms. Various hashing algorithms and their characteristics are given in table 1.
2. **Signing**: Signing is performed to obtain non-repudiation by encrypting the OSS digest using the private key of the host. The message digest can be recovered by decrypting the encrypted OSS digest called OSS signature using the corresponding public key (public key of host). Various signing algorithms and their characteristics are given in table 2.
3. **Packing**: The OSS, the signature of the OSS and the host's public key are packed together into single unit.
4. **Hosting:** The signed and packed OSS is hosted on the host.

***B) Signature Verification of Open Source Software***

1. **Downloading:** The signed and packed OSS is downloaded by the user from the host.
2. **Unpacking:** The signed OSS is unpacked into OSS, signature and the public key of the host.
3. **Hashing:** Hashing function is used to compute OSS digest from the OSS obtained after decrypting and unpacking the downloaded OSS.
4. **Decryption:** In this step, the downloaded OSS signature is decrypted using the downloaded public key of the host to obtain the OSS digest computed before uploading the OSS.
5. **Digest Comparison:** The OSS digest computed from the OSS obtained after decrypting and unpacking the downloaded OSS (step 3) is compared to the OSS digest which is decrypted from the downloaded signature (step 4) to determine their equality. In case the two are same, the signature is considered to be verified otherwise not.

**Table 1: Characteristics of HashAlgorithms**

| Name and Characteristics | Hash Size |
|---|---|
| **SHA1 (**Secure Hash Algorithm 1)[8]:<br>It's a FIPS approved algorithm; its various other versions include SHA256, SHA384 and SHA512 which provide longer hash size. | 160 bit |
| **MD5 (**Message Digest 5) [9]:<br>Potential weakness is that it can be used as a keyed hash. | 128 bits |
| **RIPEMD-160**(RACE Integrity Primitives Evaluation Message Digest 160) [10]:<br>It has been developed as part of the EC's Research and Development in Advanced Communications Technologies in Europe (RACE). | 160 bits |
| **TIGER Hash** [11]:<br>Tiger Hash has been designed for efficient operation on 64-bit platforms. | 192 bits |

**Table 2: Characteristics of Digital Signature Algorithms**

| Name, Type and Characteristics | Min. Key Size |
|---|---|
| **DSS** (Digital Signature Standard) [12]:<br>FIPS 186-2 Digital signature based on SHA1 hash, unencumbered (no patents, no licenses). | 1024 bits |
| **RSA** Digital Signature [13]:<br>RSA digital signature (FIPS approved)Previously patented digital signature (expired 2000). | 1024 bits |
| **ECDSA** (Elliptic Curve Digital Signature) [14]:<br>Digital signature based on elliptic curve key technology uses smaller keys than other public key technologies but may be encumbered by various Certicom intellectual property, licenses, and patents. Apparently, ECDSA is not covered by any Certicom patents and there are open-source ECC libraries; but Certicom does have over 300 patents on various aspects of ECC including efficient implementations of ECC in hardware and software, key agreements, etc. | 160 bits |

## 5. EXPERIMENTS WITH CRYPTOOL

Originally,CrypTool [15] was designed as an internal business application for information security and training but subsequently it has transformed into an important open-source project in the field of cryptology. CrypTool is free e-learning, open source software, which enables analysis and application of cryptographic mechanisms. It is available in multiple languages and includes both classis cryptosystem like the Caesar cipher, the ADFGVX cipher, the double-column transposition (permutation), the Enigma encryption algorithm, etc. and modern cryptosystem like the RSA and AES algorithms, hybrid encryption, algorithms based on lattice reduction and elliptic curves, etc. Besides offline versions CrypTool is available online through a browser without downloading or installing any kind of software. The offline versions are coded in C++, C# and Java. CrypTool provides online help, documentation, and tutorials to ensure that context-sensitive help, explanations of all basic cryptographic terms, a list of cryptography references, a chronology of the development of cryptography, scenarios (tutorials) for an easy introduction, and a well-sorted index of cryptographic topics. The current version of CrypTool offers analysis and applications of various classic and modern cryptographic algorithms (encryption and decryption, key generation, secure passwords, authentication, secure protocols, etc.), visualization of several algorithms (Caesar, Enigma, RSA, Diffie-Hellman, digital signatures, AES, etc.), cryptanalysis of several algorithms (Vigenère, RSA, AES, etc.), Cryptanalytical measurement methods (entropy, n-grams, autocorrelation, etc.), related auxiliary methods (primality tests, factorization, base64 encoding, etc.), number theory tutorial, accompanying script with additional information about cryptology, etc.

This study has used CrypTool to analyze the use of digital signature for distribution of Open Source Software. MD5 hashing algorithm was used to encrypt the source code and RSA digital signature algorithm was used for signing. The signed software was uploaded on web Host. The hosted signed software was downloaded and its signature was verified using the CrypTool. The results obtained have confirmed that the use of digital signature for distribution of Open Source Software ensures its integrity and authentication of the host.

## 6. CONCLUSION

An increasing number of Open Source Software projects are undergoing and numerous such software packages along with source code are available in different versions from several OSS data warehouses. It is as such essential to ensure integrity and authenticity of such software packages. This paper underlines the importance of integrity of OSS and authentication of the host of OSS for its security. Digital signature that contains identity information along with the hash of the entire software can be used to prove that the signed OSS has not been modified or altered by unauthorized person. It has further demonstrated the application of digital signature in OSS distribution using CrypTool.

## 7. REFERENCES

[1] Mohony and Naughton (2004). Open Source Software Monetized: Out of the Bazaar and into Big. The Computer & Internet Lawyer, vol. 21, no. 10, October 2004.

[2] Mark Henley and Richard Kemp (2008). Open Source Software: An introduction, Computer Law & Security Report, Vol. 24, no. 1, 2008, pp. 77-85.

[3] Free Software Foundation, http://www.fsf.org/.

[4] Open Source Initiative, http://www.opensource.org/.

[5] The Open Source Definition, http://www.opensource.org/osd.html.

[6] John E. Canava, (2001), Fundamentals of Network Security, Artech House, London, ISBN 1-58053-176-8.

[7] Subramanya, S.R.; Yi, B.K, (2006). Digital Signatures, Potentials, IEEE vol. 25, no. 2, 2006.

[8] SHA, (1995). Federal Information Processing Standards Publication 180-1, available online at: http://www.itl.nist.gov/fipspubs/fip180-1.htm.

[9] R. Rivest (1992).The MD5 Message-Digest Algorithm, IETF RFC 1321, available online at: http://www.ietf.org/rfc/rfc1321.txt.

[10] RIPE (1995). Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040)," LNCS 1007, Springer-Verlag, 1995.

[11] Ross Anderson and Eli Biham (1996). Tiger: A Fast New Hash Function, Fast Software Encryption, Third International Workshop Proceedings, Springer-Verlag, pp. 89—97.

[12] FIPS (1996). Digital Signature Standard (DSS), FIPS PUB 186-3, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-890, available online at: http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf.

[13] RSA (2002). RSA Cryptography Standard, RSA Security Inc, available online at: ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf.

[14] ANSI X9.62, (199). Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1999.

[15] CrypTool, http://www.cryptool.org/.