Java Implementation of an Authentication Protocol with Application on Mobile Phones

Bogdan Groza[1], Dragos Pop[2], Ioan Silea[1]
[1] Politehnica University of Timisoara, Faculty of Automatics and Computers, Romania,
bogdan.groza@aut.upt.ro, ioan.silea@aut.upt.ro
[2] Continental Corporation, Chassis & Safety Department, Romania,
dragos.pop@continental-corporation.com

Abstract-**Mobile phones are ubiquitous devices that begin to be involved in plenty of tasks in addition to their standard voice function. Using mobile phones to access remote information over TCP/IP is a common demand. However, when travel across networks, information is exposed to several security risks, and the use of cryptographic security unavoidable. Mobile phones still have low computational power and implementing cryptography is not easy as it requires more computational resources. In this paper we deal with the implementation of a broadcast authentication protocol based on quadratic residues chains, constructed with the squaring function, and time synchronization. The advantage of this approach is that the broadcast protocol can run for an unbounded period of time without requiring re-initialization. The protocol is implemented in Java and it is used to authenticate the information received from a remote sender. As the use of the squaring function induces the requirement for more computational power, testing this protocol in a more constrained environment, such as mobile phones, is also relevant in order to establish its performance. The results can be also viewed as the first step to implement an analogous solution to the well known hash chains.**

I. Introduction and Related Work

As information travels over insecure networks, guarantees over its security are need. For this purpose, the use of cryptographic techniques is the only alternative, as cryptography offers a large variety of functions and protocols to withstand malicious adversaries. Security objectives vary from application to application; however, four security objectives are central: confidentiality, integrity, authentication, non-repudiation. Among them, authentication is acknowledged to be the most important, since, for example it is obvious that information has little value when one can not establish its source. In brief, authentication refers to two distinct objectives: entity authentication, also known as identification, and data origin authentication, also known as message authentication, which refers to a guarantee on the source of information (this also assures the integrity of information).

Mobile telephony is a large scale industry, and more demands on mobile phones come from the market every year. In addition to their standard voice function, more services are requested by the users. As these devices became ubiquitous, and start to handle information that travels over insecure networks, the use of cryptography has become mandatory. Still, the use of cryptography can be a problem when we are faced with performance. This is because in complex scenarios, where expensive cryptographic operations are performed, a lot of computational power may be needed, and still mobile phones have somewhat low computational power compared to standard computers.

A good example of a scenario, that is frequent in practice, and complex enough to create performance problems, is assuring the authenticity, by the use of cryptographic functions, of some information that is broadcasted to a large number of receivers. Broadcast scenarios are quite frequent in practice, since there are many situations in which the same information needs to be sent to a large number of receivers.

Several papers addressed this problem and the best solution from the performance point of view is the use of elements from a one-way chain as keys for Message Authentication Codes (MAC) and time synchronization [1], [8], [9], [10]. The idea of using one-way chains in authentication originates from a identification scheme proposed by Lamport [7], also the use of time synchronization can be avoided if a challenge response mechanism is used as in [1], [4]. This is because in this way the use of secret shared keys can be avoided.

One limitation that these solutions are facing is the size of the chain that is used. The problem that occurs is that the size of the one-way chains, from which the authentication keys emerge, is limited and when the chain exhaust there are no further keys to compute the MAC codes. Therefore a new chain must be computed and the initialization procedure followed again. In order to improve on this, a solution that removes this disadvantage by using chains that are unbounded in practice was proposed in [5], [6]. As stated, the main advantage of this solution is that the chain never exhausts and the authentic broadcast can run for an unlimited period of time without requiring re-initialization for the senders or receivers. By removing the re-initialization requirement the inconveniencies that such a task raises are also removed. Since this task basically consists in the computation and verification of a digital signature this solution will remove the requirements for these and can also remove the requirement for storing digital certificates over a long period of time.

The only drawback of this approach is that the squaring function, used to compute the chain, is more computational intensive than the hash functions usually involved in this operation. As shown by the experimental results, this function is several tens of times more computational intensive than a hash function. But although the solution uses a function that is more computational intensive than hash functions, it is still affordable in many situations. What we are interested in, is to test the applicability of such a protocol in a more constrained environ-

1-4244-2577-8/08/$20.00 ©2008 IEEE

ment. Therefore, in this paper we explore the potential use of this solution on an application that runs on mobile phones. For this purpose we build a client application that we run on a Nokia 6288 phone in order to receive authentic information from some server. The mobile phone proves to be a good experimental device, since it has lower computational power than standard computers but still can handle such a protocol. Also the applications on mobile phones in which authentication is needed are quite frequent in practice. In large, this paper can be viewed as a first tentative to bring the use of a broadcast authentication protocol based on quadratic residues chains in practice. The results are positive since the protocol proves to be efficient as shown by the experimental results.

The paper is organized as follows. In section 2 the environment that we address is presented and in section 3 the cryptographic protocol is described. In section 4 experimental results are presented which show both the computational requirements for the involved cryptographic primitives but also the communication performance that can be achieved by the use of this protocol. Section 5 holds the conclusions.

II. The Environment and the Mobile Phone that we used

A brief description of the environment that we address is given. We imagined a scenario that fits in the concept of smart-home, which addresses the use of automation in private homes in order to increase the comfort and security of its inhabitants. In our scenario, some home-automation installation sends information over TCP/IP that can be accessed through mobile phones via Internet connection by its holders. We assume that the holders may not be honest to each other and therefore the use of a secret shared key between the sender, i.e. the home automation installation, and receivers, i.e. the house owners, is excluded. Therefore, this application will serve as a reduced example but relevant enough for the application of the protocol. Also the setting of the application is relevant as it is certainly that in the nearby future home automation applications will need to send information over the Internet to be downloaded on the mobile phones of their owners.

As a concrete example we used a steam boiler where water is heated under pressure to a given temperature and the boiler must give reports on its temperature. This scenario should be wlog as any other home automation may be placed instead of the heating boiler. The application setting is depicted in figure 1, and a description of the mobile phone that we used is now given.

Mobile phones are widespread devices and although their computational power, memory, capabilities and functionalities vary a lot, they still fit in the category of low computational power devices. To write an application for a mobile phone, one can use the C and C++ environments for mobile phones with Symbian or Windows Mobile as operating systems. However, the most portable and used solution is to write applications in Java 2 Micro Edition (J2ME) [17].

Even though Java implementations on mobile phones respect one or more of the following standards: Connected Limited Device Configuration (CLDC), Mobile Information Device Profile (MIDP) 1.0, MIDP 2.0 or MIDP 2.1; the application capabilities may differ from one phone to another due to the lack of support for some services.

The client application, that we developed to receive authentic information, was tested on a Nokia S40 mobile phone emulator and on a Nokia 6288 phone which is build on the S40 platform. Still, the application should run on any other MIDP 2.0 and CLDC 1.1 compatible phone. The mobile phone simulator is packed together with the Series 40 5th Edition SDK, Feature Pack 1 and it is based on Nokia 6500s Device [14].

Several technical details about the Nokia 6288 that we worked on now follow. Nokia 6288 heap (RAM) size is 2048KB, and it has 6MB internal memory, expandable with a MicroSD card. For data transfer it supports Wideband Code Division Multiple Access (WCDMA), Enhanced Data rates for GSM Evolution (EDGE), General Packet Radio Service (GPRS) and High-Speed Circuit-Switched Data (HSCSD) standards. The Java implementation from Nokia 6288 supports MIDP 2.0, CLDC 1.1 [15].

The server application which sends authentic information was developed using Java, and tested on an Intel compatible computer. The computer on which the server runs, is connected to a water heating installation which is controlled over the serial port, i.e. RS232. The server sends the reference temperature to the installation, and the installation sends back to the server the temperature measured in three different points of the vessel with heated water at every two seconds. Multiple clients can connect to the server to receive the temperature values; in particular, we run the clients on mobile phones. The transmission is done over TCP/IP, the server has a fixed IP which is known by the application that runs on the mobile phones.

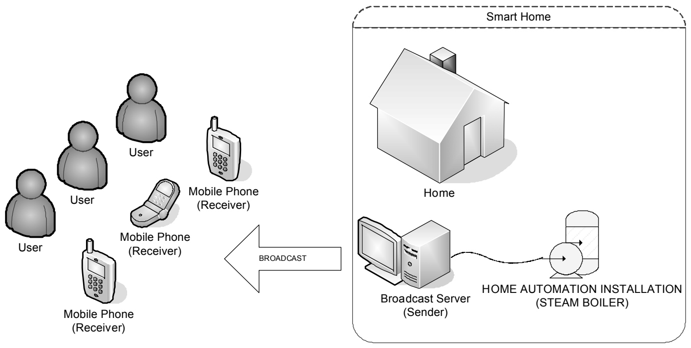

Figure 1. Application setting.

III. One-way Chain based Authentication with the Squaring Function

We now give a brief description on how to construct a one-way chain based on the squaring function, the reader is referred to [6] for a complete description of such a protocol. A one-way key chain can be defined as:

$$k_i = f^{\eta-i}(x), i = \overline{0,\eta} \qquad (1)$$

Here *f* denotes a one-way function and η is the length of the chain. Instead of using a hash function for the function *f*, we will use the squaring function:

$$f(x) = x^2 \bmod n \qquad (2)$$

Here n is composite modulus, usually in the form of $n = p \cdot q$ where p, q are two large primes. This function is commonly used in public key cryptosystem as it forms the basis of the Rabin cryptosystem [11] and also the RSA [12] uses a similar function with a different value for the exponent. It is provable that breaking this function is equivalent to solving the problem of integer factorization, which is known to be infeasible for now. The advantage in using this function instead of hash functions is that chains of lengths unbounded in practice can be computed because there is no need to sequentially compute the chain. This is because by Euler's theorem we have:

$$x^{\phi(n)} = 1 \bmod n \qquad (3)$$

And therefore, by using relation (1), we can efficiently compute the successive composition of the function by reducing the exponents modulo the order of the group, i.e. $\phi(n)$:

$$f^{\eta}(x) = x^{2^{\eta}} \bmod n \ = x^{2^{\eta} \bmod \phi(n)} \bmod n \qquad (4)$$

The computation of $f^{\eta}(x)$ can be efficiently done by first computing an exponent $e = 2^{\eta} \bmod \phi(n)$ and then computing $f^{\eta}(x) = x^e \bmod n$. By the use of the repeated square and multiply algorithm this can be done in logarithmical complexity. Also the computation of one element will not require the previously two modular exponentiations if we use a time-memory trade-off to compute sequences of elements since only one modular multiplication can be done to compute an element from another since $f^{\eta-i}(x) = f^{\eta-i-1}(x) \cdot f^{\eta-i-1}(x)$. Therefore, the computational power required for computing such a chain tends to one modular multiplication. Also, the verification on the receiver's side will always be at the cost of one modular multiplication. This property of the squaring function was also used in [2], [3] to construct a pseudo-random number generator as secure as the integer factorization problem and in [13] to create time-lock puzzles.

Now, in our scenario we will broadcast packets with the structure:

$$\mathit{Sender} \rightarrow \mathit{Receivers}:$$
$$P_i = \left\{ M_i = \{\theta_1, \theta_2, \theta_3, T\}, MAC_{KD(k_{i+1})}(M_i), k_i \right\}, i = \overline{1,\eta}$$

$MAC_{KD(k_{i+1})}(M_i)$ is a message authentication code computed on the message M_i which consists in the three temperature values from the boiler and the local time at which they were measured; each temperature value has 4 bytes and the time is 8 bytes long. $KD(k_{i+1})$ is a key derivation process from which the key of the MAC is computed from the element of the one-way chain. The MAC is of 256 bits as the SHA-256 hash function is used and the key is 1024 bits. Each packet is sent to all clients at time $t_{start} + i \cdot \delta$, where t_{start} is the time at which broadcast starts and δ is the key disclosure interval. Upon receiving each packet the receiver must verify that $t_{\max,Sender}(t_{receive}) < t_{start} + (i+1) \cdot \delta$. Here $t_{receive}$ is the time at which the packet is received, and $t_{\max,Sender}(t_{receive})$ is computed based on the time synchronization procedure, see [6] for details. If this proves to be correct the packet is stored and only later checked for authenticity when the next key is disclosed – this happens at time $t_{start} + (i+1) \cdot \delta$. Each client has to follow an initialization stage which consists in a time synchronization procedure, as described in [6], and obtaining the initialization packet which is the most recent packet sent by the server with an additional digital signature required to commit the one-way chain (the same procedures as in [6] can be also used).

The only disadvantage in using this function is the computational power required to compute the chain. The experimental results show that the computational power is affordable on the mobile phone that we used. Also, another disadvantage may be the size of the keys which is larger than for hash chains, however, this is in the order of thousands of bits and it should not be relevant when message size is large enough, for example when video-frames are broadcasted.

IV Experimental Results

A. Implementing cryptography and the communication protocol on the Nokia 6288

For the application on the server side the Java SE 6 Development Kit (JDK) was used and for the client application we used Java Wireless Toolkit 2.5.2 together with Series 40 5th Edition SDK, Feature Pack 1 [17]. As an IDE we chosed to work in NetBeans 6.0 IDE with mobility pack [18]. The cryptographic primitives from the Bouncy Castle Crypto API 1.38 are used – the package can be found at [16]. The decision to use this library was made because of it's support for both mobile and SE versions of Java. From this library we used the *BigInteger*, *HMAC* and *SHA256Digest* classes to implement the cryptographic protocol.

Both the server and the client part of the cryptographic protocol are implemented in the same package, sharing the same code, which is compiled differently for each Java version (Java SE and J2ME), one to be used on a standard computer and the other for a mobile phone. The architecture of this package is depicted in figure 2. The *IPackageProvider interface* is an interface used by the server to generate new packages and by the client to verify the authenticity of the packages received from the server. It has four member functions with the following functionalities: *byte[] getNextPackage()* returns the next package, *public int getLength()* returns the package length, *boolean isPackageValid(byte[] b)* returns true if the package is valid, *public byte[] getUnpackagedData(byte[] b)* returns the information from the package that is used for the application, i.e. the value of the temperatures, without the additional cryptographic information. The *DeMAPackageProvider class* is a class that implements the interface described above. It's main functions are those described in the interface. The *getNextPackage* function computes the new MAC on the data provided by *IDataProvider* and the current time, and then it concatenates the data, MAC code and the key of the previous MAC, along with the current time and returns it to be sent by the server. Another function is *isPackageValid*, which checks the package with the last known good key, and also verifies the time constraint. The *IMac interface* is used to generate and check the MAC. It can use any MAC function as a backend. In this application the SHA-256 HMAC was used. It was implemented in *MySHA256HMac* class which is just a wrapper for the HMAC class found in Bouncy Castle library. The *IDataProvider* interface has just two functions: *getNextData* which returns next data to be transmitted and *getLength*, which returns the length of the data. This interface has to be implemented in the application that transmits the data, in our case, the server. The *IKeyProvider* interface has two roles: to generate new keys and to verify if the received key is authentic. Its functions are: *getNextKey*, *getKeyLength* and *checkKey*. This interface is implemented in the *DeMaKeyProvider* and uses a one-way function defined in the *IOneWayFunction interface*, and implemented by the *Pow2OneWayFunction.* Finally, *Pow2OneWayFunction* implements the discrete squaring function $f(x) = x^2 \bmod n$.

As for the communication protocol, on the client side we used sockets from the *javax.microedition.io.** package. The classes that were used are the *Connector* and the *SocketConnection* classes. On the server side the *java.net.ServerSocket* and *java.net.Socket* classes were used. In both cases the TCP/IP protocol was used for it's reliability.

B. Performance of the cryptographic primitives

The speed of the cryptographic primitives was evaluated on a Nokia 6288 mobile phone. Because it has a multitasking operating system, its java virtual machine performance can be influenced by other tasks. To limit this effect we ran the benchmarking application in flight mode - in this mode of operation all phone's communication functions are stopped. For *Hash*, *HMAC* and *BigInteger* the classes from the Bouncy Castle Crypto API were used and the computational time was measured by taking the current time with the *Date* class. The computational time shown in table 1 was measured by running each cryptographic primitive 100 times and then computing the mean value. Because running the test right after the start of the application would negatively influence the performance, since initialization code will be still running in background, we ran the tests only after some user input.

C. Communication performance of the protocol

The communication test was done on a Nokia 6288 mobile phone connected to Internet using an Orange E-mail & Internet Start service plan from Orange Romania [19]. The network connection type was EDGE, at the maxim speeds of 220 kbps, but it is influenced by other factors, like radio conditions, number of connected mobiles in the cell or obstacles between the phone and the transmission station [19].

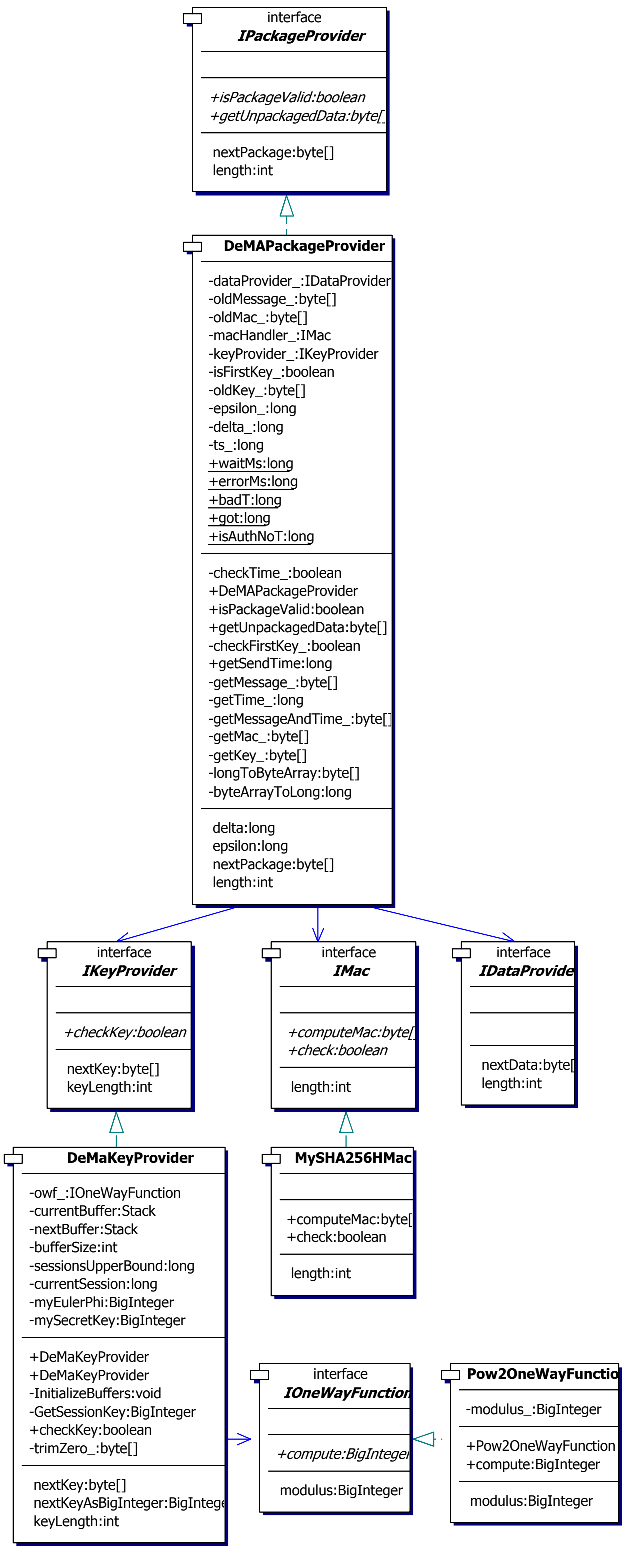

Figure 2. Application architecture.

TABLE I

COMPUTATIONAL TIME FOR THE CRYPTOGRAPHIC PRIMITIVES ON NOKIA 6288

Cryptographic primitive	**Runtime**
1024 Bit Modular Multiplication	$77.70\times10^{-3}s$
1536 Bit Modular Multiplication	$159.30\times10^{-3}s$
2048 Bit Modular Multiplication	$283.00\times10^{-3}s$
1024 Bit Modular Exponentiation with 1024 Bit Exponent	$5617.00\times10^{-3}s$
HMAC SHA1	$21.60\times10^{-3}s$
HMAC SHA256	$33.50\times10^{-3}s$
HMAC SHA512	$52.80\times10^{-3}s$
HMAC MD5	$12.00\times10^{-3}s$
SHA1	$3.90\times10^{-3}s$
SHA256	$6.40\times10^{-3}s$
SHA512	$10.10\times10^{-3}s$
MD5	$2.00\times10^{-3}s$

TABLE II

COMMUNICATION STATISTICS

Key disclosure period (seconds)	Packets sent	Authentic packets received	Loss rate (%)
1	200	200	0.00
0.1	522	516	1.15
0.05	520	514	1.15
0.025	562	461	17.97
0.015	562	374	33.45
0.01	592	248	58.11
0.005	590	0	100.00

Table 2 contains values for the loss rate of the packages based on the key disclosure period. At a rate of 100 packets/second we have a 58.11% loss rate, while at a sending rate of 40 packets/second we have a 17.97% loss rate. At 20 packets/second the loss rate is a little over 1%, which we consider as an acceptable loss rate for the addressed application.

V. CONCLUSIONS

A broadcast authentication protocol based on one-way chains constructed by the squaring function was implemented. By using this protocol a large number of receivers can get information from a sender over an unbounded period of time as the chain never exhaust. Although the function used in the construction of the chain, the squaring function, is more computational intensive than regular hash functions, our experimental results show that it is feasible to use this function even in a more constrained environment such as the Nokia 6288 mobile phone where the client application has run. As potential future work extending such a protocol even in a more constrained environment, such a microcontroller, can be a challenge.

REFERENCES

[1] F. Bergadano, D. Cavagnino, B. Crispo, "Individual Authentication in Multiparty Communications". *Computer & Security*, Elsevier Science, vol. 21 n. 8, 2002, pp.719-735.

[2] L. Blum, M. Blum, M. Shub, "Comparison of Two Pseudo-Random Number Generators", Advances in Cryptology, Proceedings of Crypto 82, 1982, pp. 61-78.

[3] L. Blum, M. Blum, M. Shub, "A Simple Unpredictable Pseudo-Random Number Generator", *SIAM Journal on Computing*, Volume 15 , Issue 2, 1986, pp. 364 – 383.

[4] B. Groza, "Using one-way chains to provide message authentication without shared secrets", Second International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, SecPerU 2006, IEEE Comp. Soc., 2006.

[5] B. Groza, T.-L. Dragomir, D. Petrica, "Using the discrete squaring function in the delayed message authentication protocol", International Conference on Internet Surveillance and Protection, ICISP'06, IEEE Comp. Soc., 2006.

[6] B. Groza, "Broadcast authentication protocol with time synchronization and quadratic residues chains", Second International Conference on Availability, Reliability and Security, ARES'07, pp. 550-557, IEEE Comp. Soc., 2007.

[7] L. Lamport, "Password Authentication with Insecure Communication", Communication of the ACM, 24, 770-772, 1981.

[8] A. Perrig, R. Szewczyk, V. Wen, D. Culler, J.D. Tygar, "SPINS: Security Protocols for Sensor Network", Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM, 2001.

[9] A. Perrig, R. Canetti, J. D. Tygar, D. Song, "The TESLA Broadcast Authentication Protocol", In CryptoBytes, 5:2, Summer/Fall, pp. 2-13, 2002.

[10] A. Perrig, R. Canetti, J. D. Tygar, D. Song, "Efficient Authentication and Signing of Multicast Streams Over Lossy Channels", IEEE Symposium on Security and Privacy, 2000.

[11] M. Rabin. Digitalized signatures and public key functions as intractable as factorization. MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[12] R. Rivest, A. Shamir, L. Adleman, „A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, 1978.

[13] L Rivest, A. Shamir, D.A. Wagner, "Time-lock puzzles and timed-release Crypto", available at http:// theory. lcs.mit.edu/~rivest/publications.html.

[14] Forum Nokia, Series 40 Platform 3rd Edition SDK, http://www.forum.nokia.com/info/sw.nokia.com/id/cc48f9a1-f5cf-447b-bdba-c4d41b3d05ce/Series_40_Platform_SDKs .html.

[15] Nokia, Nokia 6288 technical specifications, http://europe.nokia.com/A4197006.

[16] Bouncycastle.org, Bouncy Castle Crypto API, http://www.bouncycastle.org/latest_releases.html.

[17] Java Sun, Java 2 Micro Edition, http://java.sun.com/javame/index.jsp.

[18] NetBeans IDE, Free and Open Source, available at http://www.netbeans.org/.

[19] Orange Mobile Operator, Mobile Internet Access http://www.orange.ro/abonamente-date/mobile-internet.html.