A Cloud-aided group RSA scheme in Java 8 Environment

Agnieszka Krok
Tadeusz Kociuszko Cracow University of Technology, Cracow, Poland

Abstract — In this paper RSA based security system which enables the group of users upload the single masked massage to the cloud environment is proposed. The data are stored in the encrypted form by using RSA algorithm. The receiver of the data is able to encrypt the message retrieved from the cloud environment using his own private key. Two different separate RSA systems are used. The approach is divided into three parts. First: an RSA key generating for each member of the group of senders. Then masking the massage by newly chosen mask proposed individually by every member, additionally encrypted by individual RSA private key of each member. Second: encrypting the gathered message inside the cloud environment, using the public key of the receiver. Third: decrypting the message by the receiver using his private RSA key. The scheme is design in the way that it reduces of the computational load of the end users and transfers calculations and storage effort to the cloud environment. Algorithm is used for storing and sending the data that originally are produced by a group of users but the receiver of the data is single.

Keywords — Cloud computing, RSA cryptosystem, confidentiality

1. Cloud computing

Cloud computing model is based on a business model in which resources are shared (multiple users use the same resource) at the network level, host level, and application level. It provides massive scalability to tens of thousands of systems, and the ability to store large amount of data and the very efficient computational power. The cloud offers resources such as virtual-machine disks, image libraries, file storage, firewalls, mailing systems, load balancers, IP addresses, virtual local area networks, software bundle,operating systems, programming languages execution environments, database, and web servers, [8].
Users may access to cloud environment using their client devices, such as desktop computers, laptops, tablets and smart phones. They are thin clients because cloud services do not require specific advanced software on the client device. The procedures for clients and cloud environment should transfers calculations effort on the machine in the cloud. The most widely used examples of cloud computing are Google Cloud, Microsoft Cloud, Amazon Cloud and Adobe Creative Cloud, [10-13].
The following features of Cloud Systems distinguish them from traditional systems of services and resources:

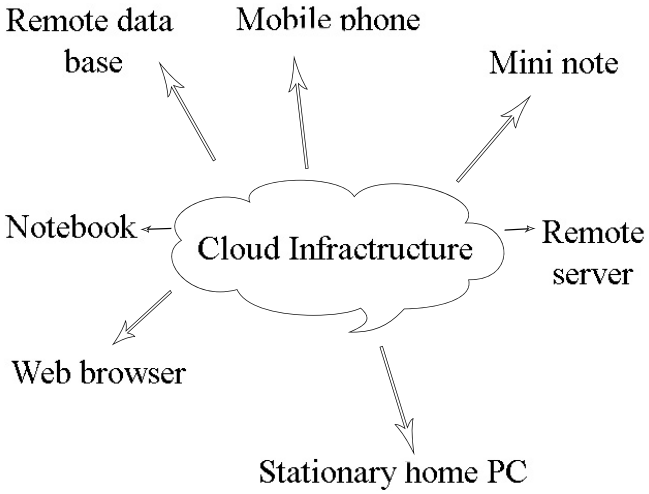

Fig. 1. Features of Cloud Systems

- Multitenancy - unlike previous computing models, which assumed single user resources in Cloud model resources are shared
- Massive scalability - traditional models might have hundreds or thousands of systems, cloud computing provides the ability to scale to tens of thousands of systems, and the ability to massively scale bandwidth and storage space.
- Elasticity - users can increase and decrease their computing resources when they need
- Pay as you go - users pay for only the resources they actually use and for only the time they require them
- Self-provisioning of resources - Users may introduce additional systems (processing capability, software, storage) and network resources,[7].

From a customers perspective traditional models have high upfront IT investments for new builds, high complexity of IT environment, complex infrastructure and high cost of reliable infrastructure. Cloud models: low upfront IT investments;Modular IT architecture environments; no infrastructure and reliability built into the cloud architecture.

2. Security issues in cloud environment

Cloud environment is a target for malicious individuals and organizations as far as traditional security threats, threads related to system availability and third-parity data control. At the user site, he must protect the infrastructure used to connect to the cloud. The service provider have to deliver the authentication and authorization techniques moreover individuals may be assigned different levels of privileges or may act together sharing the resources. Data and services in the clouds may be affected by numbers of attacks from distributed denial of service, phising, data loss or leakage, shared technology and the abuse of the resources, [7]. The security objectives of an organization are a key factor for decisions about outsourcing information, technology services and for decisions about transitioning organizational data, applications, and other resources to a public cloud computing environment, [9].
All three service models may be a target for the threats: Infrastructure as a service (IaaS), Platform as a service (PaaS) Software as a service (SaaS).
Infrastructure Security at the Network Level requires:

- Ensuring the confidentiality and integrity of user data-in-transit to and

  from your public cloud provider

- Ensuring proper access control (authentication, authorization, and auditing) to

  resources clients are using at their public cloud provider

- Ensuring the availability of the Internet resources in a public cloud that are being used by user or have been assigned to user by his

  cloud providers, [7]

The proposed scheme may be used in two first above requirements.
Infrastructure Security at the Host Level requires:

- PaaS and SaaS platforms abstract and hide the host operating system from end users

  with a host abstraction layer.

- The abstraction layer is not visible to users and is available only to the developers

- Host security responsibilities in SaaS and PaaS services are transferred to the Cloud model,[7].

The proposed scheme as all above requirements.
The data in Cloud models should be treated deferentially according the their status: data-in-transit, data-at-rest, processing of data, including multitenancy, data lineage, data provenance, data remanence. Data-in-transit should be encrypted during transfer to and from a cloud provider. Encrypting data-at-rest is possible and is suggested. But data -at-rest in stored inside cloud-based infrastructure is generally not encrypted, because encryption would prevent indexing or searching of that data. Data lineage that it following the path of data is important for an auditors assurance. Integrity of data ensures that data has not been changed in an unauthorized manner or by an unauthorized person. Provenance means that the
data is computationally accurate. Data remanence refers to the policy of treating data that are not used, not actual, has been erased by user from his applications, [7]. In the proposed scheme data-in-transit are treated differently from data-in-rest, data lineage is strictly defined, processing of the data and data provenance are supported only by cloud software, data remanence depend on both: user and Cloud infrastructure. Data -at-rest in stored inside cloud-based infrastructure is encrypted.

3. RSA crypto system

RSA (proposed by Ron Rivest, Adi Shamir and Leonard Adleman, [15]) is the public key algorithm for encrypting and decrypting the data. Encryption is making a communication private. The sender enciphers each message before transmitting it to the receiver. The receiver knows the proper deciphering function to apply to the received message to obtain the original message. An eavesdropper (who hears the transmitted message) knows only the cipher text which can not be understood since he does not know how to decrypt it. Only the authorized user can have an access to the data stored. RSA algorithm may be used also for digital signatures evaluating, [14]. Every plain text written in natural language is mapped into integer number and encrypted using modulo arithmetic algorithm that proceeds on big numbers. Algorithm involves three stages:

1. Key generation starts from choose two different large random prime numbers p and q and calculating $n = *pq$ the modulus for the public key and the private keys. Calculate the totient: $\phi(n) = (p-1)(q-1)$. Choose an integer e such that $1 < e < \phi(n)$ and e, is coprime to e, ie: e, and $\phi(n)$ share no factors other than 1; $gcd(e, \phi(n)) = 1$.

   The public key is made of the modulus n and the public exponent eThe private key is made of the modulus n and the private exponent d, which must be kept secret.

2. To encript the message M for particular receiver the sender is using public key of receiver. First the sender turns M into a number m smaller than n by using an agreed-upon reversible protocol known as a padding scheme, [1]. He then computes the ciphertext c corresponding to:

$$c = m^e \mod n$$

3. The receiver can recover m, from c, by using his private key d:

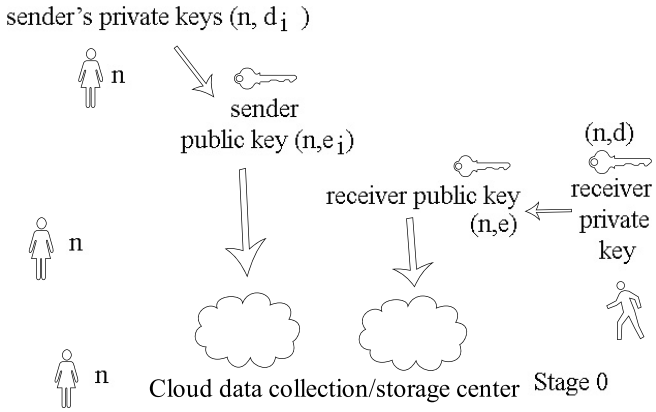

Fig. 2. Key preparation during stage 0

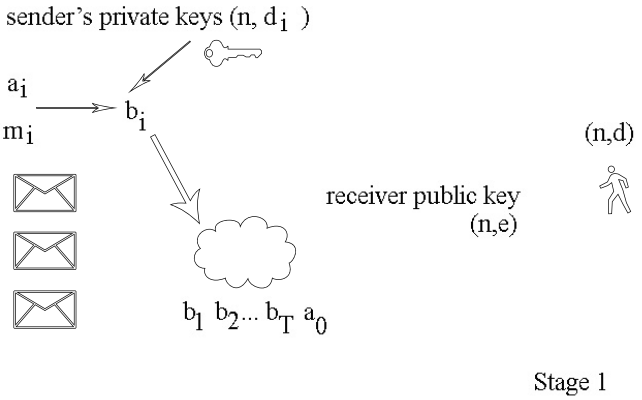

Fig. 3. Gathering messages from recipients inside Cloud data collection center during stage 1

$$m = c^d \mod n$$

Given m one can recover the original message M.

It is currently recommended that n be at least 2048 bits long, [2].
The RSA algorithm is used as a part of security system inside Google Cloud, [3], [4], and in many handshaking protocols as a part of negotiation that dynamically sets parameters of a communications channel established between two entities before normal communication over the channel begins in the Transport Layer, [16].

4. Proposed Cloud-aided group RSA scheme

Proposed Cloud-aided group RSA scheme is based on the procedure [17] but additional stage was added to secure the first step of the algorithm before data are send to the Cloud computing center. Moreover, proposed algorithm enables sending not only the single message (as in [17]) but the message may be composed from different parts coming from different senders.

4.1. Stage 1

Lets assume that a group of users (called senders) $S_1, S_2, .., S_T$ is sharing the same file or data in the Cloud infrastructure. They may modify and send it to receiver chosen from outside this group. All parts are gathered to portion of data called a message m in the Cloud environment so that the receiver R may retrieve it when he needs. The message is stored in the Cloud storage center in the encrypted form.
Four actors are considered: group of senders, Cloud data collection center, Cloud data storage center, receiver of the data. The group of senders contacts only with Cloud data collection center, receiver contacts only with Cloud data storage center, Cloud data collection center and Cloud data storage center are contacting each other.

4.2. Stage 0

Key preparation.
The receiver generates his RSA private key (d,n) and public RSA key (e,n) and sends public RSA key to the Cloud data storage center. The Cloud data storage center sends to Cloud data collection center parameters n and e. Cloud data collection center sends parameter n to each sender. Each sender $i = 1,2,...,T$ generates the own private RSA key (d_i,n), and public RSA key (ein,n) and sends public RSA key to the Cloud data collection center. Only the public keys are transfered.
Sending decrypted data to Cloud data collection center.
Each sender would like to send his part m_i of the massage $m = m_1 * m_2 * ... * m_T$. He generates new random number a_i such that $a < n$ and computes $b_i = (m_i * a_i)^{d_i} mod n$. Then he sends b_i to Cloud data collection center. Additionally $a_0 = a_1 * a_2 * .. * a_T mod n$ is added. In case only one member is sending his message, all the a_i but his are set to 1.

4.3. Stage 2

Processing data in Cloud data collection center
Cloud data collection center applies public key of each sender e_i to proper part of data and then applies public key of the receiver e: compute

$$v_0 = {a_0}^e$$

$$v_i = ({b_i}^{e_i})^e mod n =$$

$$((m_i * a_i)^{d_i * e_i})^e mod n = (m_i * a_i)^e mod n$$

and composes data into $V = (v_0, v_1, v_2, ..., v_T)$ vector. Then permutes it by using random permutation P: $V_{perm} = (v'_0, v'_1, v'_2, ..., v'_T)$. Vector V_{perm} is send to Cloud data storage center as the data from the group of senders to single receiver.

4.4. Stage 3

Processing data in Cloud data storage center
Cloud data data storage center applies inverse permutation

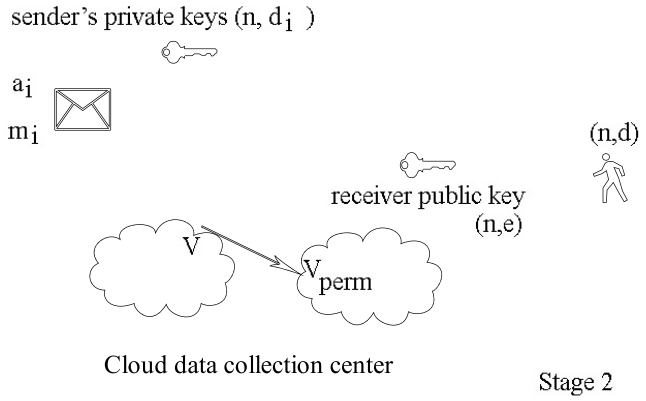

Fig. 4. Mixing messages from recipients inside Cloud data collection center into single message during stage 2

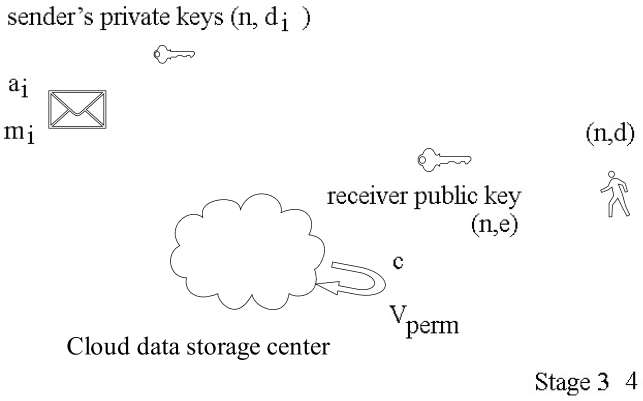

Fig. 5. Storing data inside Cloud storage center in the form of single message during stage 3

P^{-1} to V_{perm} , obtaining $V = (v_0, v_2, ..., v_T)$ vector. Then computes:

$$c = (v_1 * .. * v_T) * {v_0}^{-1} mod n =$$

$$(m_1 * m_2 * ... * m_T)^e mod n =$$

$$m^e mod n$$

This value is stored until the receiver would like to retrieve it.

4.5. Stage 4

Downloading decrypted data to Cloud data storage center and encryption by receiver.
Receiver uses his own private key d

$$m = c^d \mod n = (m^e)^d mod n = mmod n = m$$

Given m one can recover the original message M by using reverse padding scheme.
Data is more vulnerable in storage than while it is being processed so the strongest RSA keys should be applied during stage 2 and 3. It also moves the afford of storage the data into Cloud environment. Masks $a_1, a_2, ..., a_T$ are different for each message so there is no need for generating new RSA key for every single message. Using the masks, the true message m_i is indistinguishably from empty message m_i during sending stage 1. If sender sends no message computational afford is low, because masks may be much smaller then messages. The group is invisible to the receiver. The receiver do not know who from the group send the particular part of the message or even that the message was composed from parts.

5. Experimental results in Java 8 environment

Java 8 environment was chosen for three main reasons. Firstly, for the convenience of the use of security package. Secondly for the possibility of usage BigInteger library that is necessary for computing on such large numbers. Thirdly because secure pseudo-random number generator in Class SecureRandom that is necessary for searching for e,p and q. The RSA part of the scheme was implemented using the following Java 8 classes, methods and interfaces from java.security package:

- Key - interface models the base characteristics for the keys;
- KeyFactory - Key factories are used to convert keys into key specifications (transparent representations of the underlying key ;
- KeyPair - serves as a container for public and private keys;
- KeyPairGenerator - a class used to generate key-pairs for a security algorithm;
- generatePrivate- method generating a private key from the provided key specification;
- generatePublic - method generating a public key from the provided key specification;
- Interface KeySpec - transparent specification of the key that sets a cryptographic key. If the key is stored on a hardware device, its specification may contain information that helps identify the key on the device, [5].

To generate the keys from the Key Factory:

```
KeyPairGenerator key_par_gen =
  KeyPairGenerator.getInstance("RSA");
key_par_gen.initialize(2048);
KeyPair kp =
  key_par_gen.genKeyPair();
PublicKey publicKey =
  key_par.getPublic();
PrivateKey privateKey =
```

```
  key_par.getPrivate();
KeyFactory factory_rsa =
  KeyFactory.getInstance("RSA");
RSAPublicKeySpec pub =
  factory_rsa.getKeySpec
   (publicKey, RSAPublicKeySpec.class);
RSAPrivateKeySpec priv = factory_rsa.
  getKeySpec
   (privateKey,RSAPrivateKeySpec.class);
```

To retrieve the values of the key due to the key length 1024, 2048 bites long java.math.BigInteger library was used:

```
BigInteger
n = (BigInteger)
object_input_stream.readObject();
BigInteger
e = (BigInteger)
object_input_stream.readObject();
KeyFactory factory_rsa =
KeyFactory.getInstance("RSA");
if (keyFileName.startsWith("public"))
 return factory_rsa.
 generatePublic
 (new RSAPublicKeySpec(n, e));
else
 return factory_rsa.
 generatePrivate
 (new RSAPrivateKeySpec(n, d));
```

Public class Cipher was used to encrypt and decrypt the messages during stage 1, [6].

- javax.crypto.Cipher - class provides the functionality of a cryptographic cipher for encryption and decryption;
- javax.crypto.CipherInputStream - constructs a CipherInputStream from an InputStream and a Cipher;
- javax.crypto.CipherOutputStrea - Constructs a CipherOutputStream from an OutputStream and a Cipher.

```
Key public_Key =
readKeyFromFile("public.key");
    Cipher cipher =
    Cipher.getInstance("RSA");
cipher.init
  (Cipher.ENCRYPT_MODE, public_Key);

 Key private_Key =
 readKeyFromFile("private.key");
   Cipher cipher =
   Cipher.getInstance("RSA");
cipher.init
(Cipher.DECRYPT_MODE, private_Key);
```

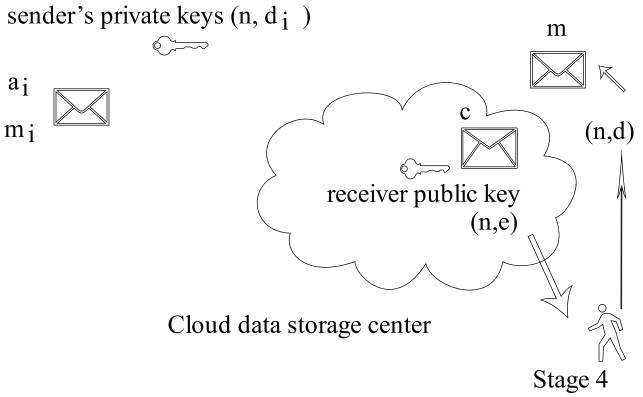

Fig. 6. Retrieving message from Cloud data collection center and encryption during stage 4

It was found profitable to proceed input message in chunks. Chunks were generated by dividing the message into equal parts. Specified number of bits was considered. 32, 64, 128,...,65536 bites of original message was tested. Chunks was coded and decoded in parallel mode. The time of calculation is highly dependable on the characterization of the computational unit. That is why the size of the chunks should be matched depending on computing capabilities. Instead of multiplication of fragmentary messages other operations may be performed then equivalent methods for generating a_0 have to be adopted.

6. Conclusions and future development

Proposed RSA based security system enables the group of users sending the single masked massage to the cloud environment where data are stored encrypted by RSA algorithm. Receiver of the data is able to encrypt the message retrieved from the cloud environment when he needs, regardless the time message was send. Two different separate RSA systems are used according to the computing capabilities: thin client for senders and parallel computations for Cloud environment. The RSA key generating for the each member of the group of senders and masking the massage by newly chosen mask proposed individually by every member, additionally encrypted by individual RSA private key of each member enable to send message coded with minimal computational effort: mask may be much smaller then message. Encrypting the gathered message inside the cloud environment using the public key of the receiver enables to hide the group from the receiver. Decryption the message by the receiver using his private RSA key ensures that if the private key was kept secret no one but receiver could read the massage stored inside Cloud environment.

References

[1] PKCS 1 RSA Cryptography standard, RSA Laboratories,2012 http://www.emc.com/emc-plus/rsa-labs/pkcs/files/h11300-wp-pkcs-1v2-2-rsa-cryptography-standard.pdf]

[2] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone,Handbook of Applied Cryptography, ISBN-13: 978-8189836122, 1996, CRC Press

[3] Amazon Cloud Documentation https://cloud.google.com/storage/docs/authentication-getting-a-private-key

[4] Amazon Elastic Compute Cloud User Guide for Linux http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html

[5] Java Platform Standard Edition 8 Documentation, 2015, Oracle https://docs.oracle.com/javase/8/docs

[6] Java Platform, Standard Edition 8 API Specification, 2015, Oracle, https://docs.oracle.com/javase/8/docs/api/javax/crypto/Cipher.html.

[7] T. Mather, S. Kumaraswamy, S. Latif, Cloud Security and Privacy An Enterprise Perspective on Risks and Compliance, T. Mather, S. Kumaraswamy, S. Latif, 2009, ISBN 978-0-596-80276-9, O'Reilly

[8] J. Rittinghouse, J. Ransome, Cloud Computing: Implementation, Management, and Security, ISBN-13: 978-1439806807,2009, CRC Press

[9] W. Jansen,T. Grance, Guidelines on Security and Privacy in Public Cloud Computing, W. Jansen,T. Grance, http://csrc.nist.gov/publications/nistpubs/800-144/SP800-144.pdf,2011

[10] https://cloud.google.com/

[11] https://www.amazon.com/clouddrive/home

[12] http://www.microsoft.com/enterprise/microsoftcloud

[13] http://www.adobe.com/pl/creativecloud.html

[14] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM 21 (2): 120126, 1978

[15] R. Rivest, Cryptography, Chapter 13 of Handbook of Theoretical Computer Science, ed. J. Van Leeuwen, vol. 1 (Elsevier, 1990), 717–755.

[16] The Transport Layer Security (TLS) Protocol Version 1.2, T. Dierks, Network Working Group , 2008, http://tools.ietf.org/html/rfc5246

[17] "A Cloud-aided RSA Signature Scheme for Sealing and Storing the Digital Evidences in Computer Forensics", Chu-Hsing Lin, Chen-Yu Lee, Tang-Wei Wu, International Journal of Security and Its Applications Vol. 6, No. 2, 2012

Agnieszka Krok received her M.Sc. in the field of stochastic processes at the Jagiellonian University, Cracow, Poland and Ph.D. degree in the field of neural networks at Tadeusz Kosciuszko Cracow University of Technology, Poland, in 2003 and 2007, respectively. From 2009 she is an Assistant Professor at Faculty of Physics, Mathematics and Computer Science, Tadeusz Kosciuszko Cracow University of Technology. Her main scientific and didactic interests are focused mainly on Artificial Intelligence: Artificial Neural Networks, Genetic Algorithms, and additionally on Parallel Processing and Cryptography.

E-mail: agneskrok@gmail.com

Faculty of Physics, Mathematics and Computer Science
Tadeusz Kosciuszko Cracow University of Technology
Warszawska st 24 31-155 Cracow, Poland