IMPLEMENTATION OF RSA 2048-BIT AND AES 256-BIT WITH DIGITAL SIGNATURE FOR SECURE ELECTRONIC HEALTH RECORD APPLICATION

Mohamad Ali Sadikin[1] and Rini Wisnu Wardhani[2]
Cryptography Engineering, Faculty of Engineering, National Crypto Institute, Bogor 16330, Indonesia
Email: [1]mohamadalisadikin@gmail.com, [2]rini.wisnu@stsn-nci.ac.id

***Abstract*—This research addresses the implementation of encryption and digital signature technique for electronic health record to prevent cybercrime such as robbery, modification and unauthorised access. In this research, RSA 2048-bit algorithm, AES 256-bit and SHA 256 will be implemented in Java programming language. Secure Electronic Health Record Information (SEHR) application design is intended to combine given services, such as confidentiality, integrity, authentication, and non-repudiation. Cryptography is used to ensure the file records and electronic documents for detailed information on the medical past, present and future forecasts that have been given only to the intended patients. The document will be encrypted using an encryption algorithm based on NIST Standard. In the application, there are two schemes, namely the protection and verification scheme. This research uses black-box testing and white-box testing to test the software input, output, and code without testing the process and design that occurs in the system.We demonstrated the implementation of cryptography in SEHR. The implementation of encryption and digital signature in this research can prevent archive thievery.**

Keywords: **Electronic Medical Record; Digital Signature; Cryptography; Java Programming**

I. INTRODUCTION

Medical records based on paper still have some flaws and problems. Those problems occur ranging from physical security, requiring storage area, difficult to transfer or communication the information, easily damaged and destroyed. If the storage process is not performed properly, it will complicate the search process or the information retrieval. In addition to the many possible disasters, health record information are personal data for someone and human life. For that, we need a solution to resolve the issue.

Received: August 1, 2016; received in revised form: August 10, 2016; accepted: August 11, 2016; available online: August 19, 2016.

The process of manually organising and managing on paper media has a few shortcomings in the aspect of information security that is confidentiality, data integrity, availability, non-repudiation, and authentication [1]. The electronic health records (EHR) has great benefits to health services such as primary and referral service facilities and hospitals.

The perceived benefits are increasing availability of electronic patient records in hospitals, improving the efficiency of the health care retrieval process [2], facilitating retrieval of patient information [3], easy access to patient information that ultimately help in clinical decision-making, and reducing operational impact cost and earnings improvement in health care facilities especially hospitals [4].

The EHR should only be accessed and shared by authorised healthcare providers such as doctors, nurses, lab technicians due to its function to record any critical information for every patient. That critical information such as the enforcement of diagnosis, therapy, avoids allergic reactions and drug duplication [5]. This practice is consistent with ethical considerations in the application of information [5] technology, where all healthcare providers have a moral code that requires the balance between the patient privacy and the care needs including the access to the records of patients [6].

A recent research article proposed that Public Key Infrastructure (PKI), symmetric key and login password for authentication are used for the security of the EHR [7]. Based on ISO/TS 18308 standard [8], the primary purpose of the EHR is to provide a documented record of care which supports both present and future care received by the patient from the same or other clinicians or care providers. This documentation provides a mean of communication among clinicians contributing to the patient’s care.

In this paper, EHR was designed and built using digital signature and file encryption. Digital signature and file encryption are used not only to solve confidentiality, data integrity, availability, non-repudiation, and authentication problem but also to prevent robbery, modification and unauthorised access. Secure Electronic Health Record Information (SEHR) is a secure electronic health record which uses RSA 2048 bit [9], AES 256 bit [10] and SHA 256-bit algorithm that is implemented in Java programming. The cryptography aspect is expected to ensure the file records and electronic documents on patient’s identity, examination, treatment, action and service given and the authorised person.

II. FUNDAMENTAL THEORY

A. Electronic Health Record

Electronic Health Record (EHR) is a comprehensive patient’s health information electronic record which is an integration of the multiple health information databases. The record contains patient demographics, progress notes, problems, medications, vital signs, past medical history, immunisations, laboratory data and radiology reports [11].

The EHR includes all information contained in a traditional health record including a patients health profile, behavioural and environmental information. The EHR also includes the dimension of time, which allows inclusion of information across multiple episodes and providers, which will ultimately evolve into a lifetime record [12]. The EHR defined here contains all personal health information belonging to an individual. Those data are entered and accessed electronically by healthcare providers over the patient’s lifetime. The EHR contains data beyond the acute inpatient situations, including all ambulatory care settings at which the patient receives care [13]. Based on the Regulation of the Minister of Health about the filling of medical records, it is stated that legal sanction can be given to the hospital or health workers who fail to pay a close attention and commit mistakes in filling the pages of medical records [14].

B. Digital Signature

Cryptography focuses on the issue of maintaining the confidentiality of information by using methods and mathematical techniques that include confidentiality, the data integrity, entity authentication, and data origin authentication [15]. RSA (Rivest, Shamir, Adleman) algorithm is an asymmetric cryptographic invented by Rivest, A. Shamir, and L. Adleman in 1997 [15]. In this research, RSA algorithm is applied as a digital signature scheme. The RSA algorithm is used due to its fast computation compared to ECDSA and DSA [16].

In the process of signature generation and verification, an entity A marks the message $m \in \mathcal{M}$. The entity B can verify A’s signature and return the message m from the signature. The procedure is of the following (see Fig. 1).

1) Key Generation in RSA Digital Signature.
   - Determine randomly two large prime numbers p and q.
   - Compute $n = pq$ and $\varnothing = (p-1)(q-1)$.
   - Choose a random integer e in $1 < e < \varnothing$, so that $\gcd(e, \varnothing) = 1$.
   - Use the extended Euclidean algorithm to compute d where $1 < d < \varnothing$, so that $ed \equiv 1(\text{mod}\varnothing)$
   - Public key: (n, e) and the private key: d.
2) Signing
   - Compute $\tilde{m} = R(m)$, an integer in the range of $[0, n-1]$
   - Compute $s = \tilde{m}^2 \text{mod}(n)$
   - A’s digital signature for m: s.
3) Verification
   - Getting A’s public key: (n, e)
   - Compute $s' = \tilde{s}^e \text{ mod}(n)$.

Verification: if the value of $s = s'$ then the digital signature is authentic.

C. Advanced Encryption Standard

Advanced Encryption Standard (AES) is a block cipher algorithm which is intended to replace DES algorithm as a standard and is recognized for some applications [18]. AES is also a standard algorithm for data encryption and decryption (Eric Conrad, Advanced Encryption Standard). In this research, AES is used because due to its advantages to secure documents and is proven to be safe based on NIST Standard [10]. The AES algorithm is outlined in Fig. 2.

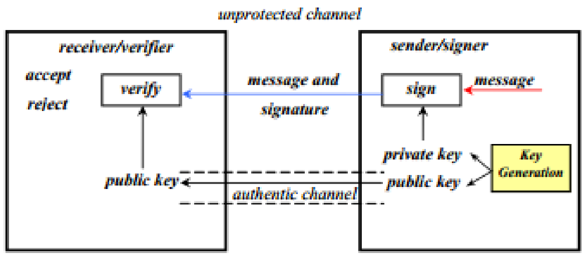


Fig. 1. The Digital Signature Scheme [17].

For the decryption process, inverse process is used at the transformation stage. The process starts from InvSubBytes, InvShiftRows, and ends in InvMixColumns. Because of this, S-box for encryption and decryption are different. In the decryption process, the used S-box is the inverse S-box.

D. Secure Hash Algorithm 256 bit

Secure Hash Algorithm (SHA) is used to generate hash value from the message M with the length l bit, where $0 \leq 1 \leq 2^{64}$. The algorithm is

1) Message Schedule for 64 words 32-bit. Words of the message schedule are labeled as $W_0, W_1, \ldots, W_6$.
2) Eight 32-bit variables labeled as $a, b, \ldots, h$.
3) The hash has value of eight 32-bit words are labeled as $H_0^i, H_1^i, \ldots, H_7^i$.

The preprocessing consists of three steps, namely padding of the message M, split up the message into message blocks, and set the initialization hash value H^0; then, proceed with to the SHA-256 computing process.

III. SECURE ELECTRONIC HEALTH RECORD APPLICATION

A. General Description The Application

Secure Electronic Health Record application is an application that applies the concept of digital signatures using RSA and SHA-256 algorithms and AES-256 block cipher algorithm for the encryption process.

The application will be implemented in Java programming language that guarantees the integrity, confidentiality, authentication and non-repudiation. Java language is used because it is more mobile, multi-platform, object-oriented, portable, and open source. It has two schemes which are the protection scheme and verification scheme. It is assumed that the protection and verification processes are contained in one application. The protection step is on the tab SIGN & ENCRYPT and the verification process is on the VERIFICATION tab. The details of the two processes are of the following.

1) Signing and Encryption Scheme: On the protection schemes, two processes are running: the securing process (encryption) and the authenticating process (signing). The generation of the private key and the public key is done before the encryption and signing process executed. The scheme of securing and authenticating documents is done in a simple manner following the scheme in Fig. 3.

2) Verification Scheme: The verification process is done by reversing the signing process (see Fig. 4). The file is firstly decrypted using a key that has been used previously in the encryption process. After this process, the file is hashed and then digital signature calculations using a public key that has been generated and stored in the protection stage is performed. The verification process is the process of calculating the digital signature value of the hashed document using the public key. If appropriate, it will display a notification that the document is successfully decrypted and proven to be authentic. If it does not match, the notification will show that the document was not authentic or has been a change.

B. Implementation of Secure Electronic Health Record Application (SEHR)

The implementation of digital signature File Encryption includes steps using SEHR application. The steps in the implementation of SEHR application are as follows:

1) Login process: When the user runs the application, the Welcome message will appear to start logging the process (see Fig. 5). The user needs to fill the USERNAME and PASSWORD fields. By pressing the LOGIN button, the login process

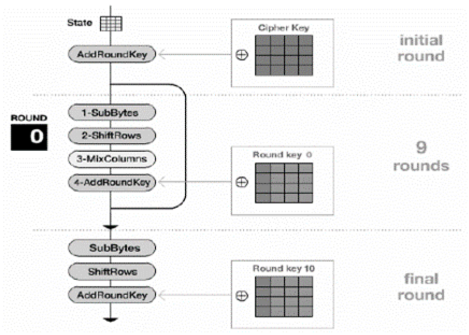


Fig. 2. Encryption Process Diagram [19].

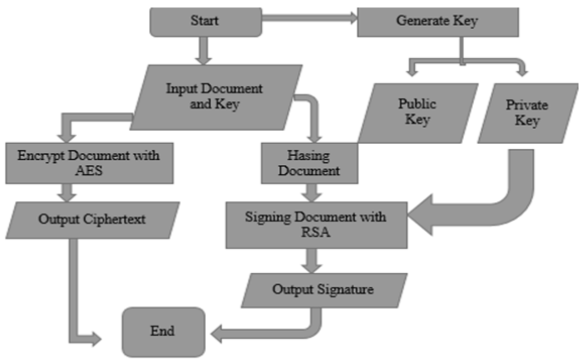


Fig. 3. The Encryption and Signing Scheme

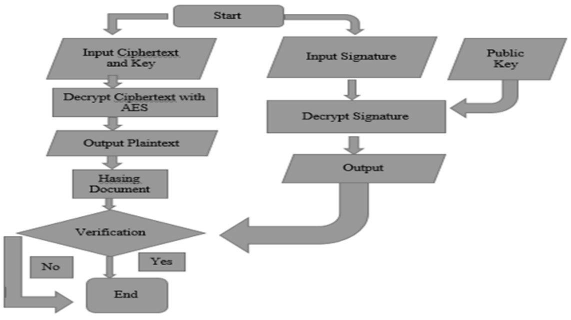
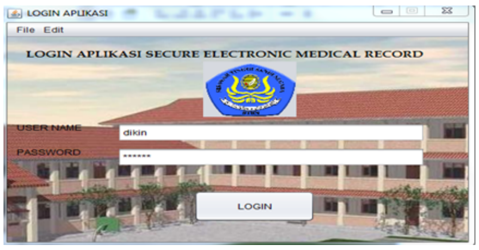


Fig. 4. The Verification Scheme



LOGIN APLIKASI SECURE ELECTRONIC MEDICAL RECORD

USER NAME dikin
PASSWORD ******

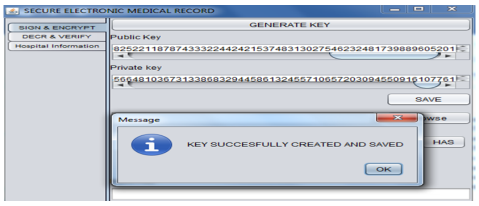
Fig. 5. The Login Page of the System.

will be executed. The application will verify the username and password submitted by the user. If the password corrects, the application main view will be displayed. However, if the username, password, or both incorrect, then notification appears and the user cannot access to the next application process. Following is a snippet from the source code of the `Class login()`:

```
String UserName = jTextField1.getText();
String Password = jPasswordField1.getText();
if(UserName.equals("dikin")&&(Password.equals("123456"))){
    JOptionPane.showMessageDialog(null,"LOGIN SUCCESS" + " \n
    WELCOME TO Electronic Medical Record Application" + " \n
    Application by: MOHAMAD ALI SADIKIN" + " \n
    SEKOLAH TINGGI SANDI NEGARA" + " \n
    1413101075", "file", JOptionPane.INFORMATION_MESSAGE);
    dispose(); new latjab().setVisible(true);
} else {
    JOptionPane.showMessageDialog(null,
    "WHO ARE YOU ?? I DONT KNOW YOU ( -_-')",
    "ERROR !!",JOptionPane.ERROR_MESSAGE);}
}
```

2) Generate RSA Key: The user generates the key by pressing Generate RSA KEY button (see Fig. 6). In the application, a secure random is used to generate RSA private key and public key parameters. Furthermore, by pressing the SAVE button, the private key and public key will be stored on the file extension `*.txt` and a notification that the private key and public key have been successfully generated and stored is displayed. The folowing is a snippet from the source code of `Class RSA` and the view of application when RSA key generated.

```
public class RSA {
    private BigInteger p;
    private BigInteger q;
    private BigInteger N;
    private BigInteger phi;
    private BigInteger e;
    private BigInteger d;
    private int bitlength = 1024;
    private int blocksize = 256; // blocksize in byte
    private SecureRandom r;
    BigInteger pesan;
    String Privat;
    String Public;
    string g;
    BigInteger s, v;
    String y;
    public RSA(){
        r = new SecureRandom();
        p = BigInteger.probablePrime(bitlength, r);
        q = BigInteger.probablePrime(bitlength, r);
        N = p.multiply(q);
        byte [] byt=p.toByteArray();
        phi= p.subtract(BigInteger.ONE).multiply(
           q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(bitlength/2, r);
        while(phi.gcd(e).compareTo(BigInteger.ONE) > 0 &&
            e.compareTo(phi) < 0 ){
            e.add(BigInteger.ONE);
        }
        d = e.modInverse(phi);
        Public=e.toString();
        Privat=d.toString()
    }
    public RSA (BigInteger e, BigInteger d, BigInteger N){
        this.e = e; this.d = d; this.N = N;
    }
```



Public Key
82522118787433322442421537483130275462324817398896052O1
Private key
56648103673133868329445861324557106572030945509161077 61

KEY SUCCESFULLY CREATED AND SAVED

Fig. 6. The Generated RSA Key.

3) Signing and Encryption: The user needs to input the document in the field FILE to start the signing and encrypting process (see Fig. 7). Furthermore, the user inputs the key in the field INPUT KEY. To encrypt the document user needs to press the ENCRYPT button. When the user pressing SIGN button, `signcrypt` method will be running and the document has been signed and encrypted. Following is a snippet from the source code of `Class signcrypt` and the application view when the application executes the signing and encryption process. The output of this process stored in a location where the document was taken with a different file name with the original file. Signed and encrypted files are stored as a file extension `*.txt`. Furthermore, these files will be used in the verification process.

```
Class signcrypt{Private void jButton4ActionPerformed
        (java.awt.event.ActionEvent evt)} {
    String kunciaes=jTextField2.getText();
    File file = new File(NamaFile);
    try {
       pesanstring = FileUtils.readFileToString(file);
    }
    catch(IOExceptionex){
       Logger.getLogger
          (latjab.class.getName()).log(Level.SEVERE, null, ex);
    }
    try{
       pesanterenkrip=AES.encryptAES(pesanstring,kunciaes);
    }
    catch(Exceptionex){
       Logger.getLogger
          (latjab.class.getName()).log(Level.SEVERE,null,ex);
    }
    String aku= "e:/LSM1";
    File file1=new File(aku,"encryptedaes.txt");
    try{
       FileUtils.write(file1, pesanterenkrip);
    }
    catch(IOExceptionex){
       Logger.getLogger
          (latjab.class.getName()).log(Level.SEVERE, null, ex);
    }
}
private void jButton5ActionPerformed
  (java.awt.event.ActionEvent evt){
   try {b.encrypt(a.k);
   JOptionPane.showMessageDialog(null,
      "FILE SUCCESSFULLY SIGNED AND ENCRYPTED",
      "DONE",JOptionPane.INFORMATION_MESSAGE);}
    catch (IOException ex) {
        Logger.getLogger(latjab.class.getName()).
           log(Level.SEVERE,null,ex);
    }
}
```

4) Verification: The implementation of verification scheme is made separately with the protection schemes even though they are contained in a single application (see Fig. 8). To run the verification process, it takes three inputs which are a document file that has been encrypted (file results from encryption process with extension `*.txt`), the signature file (file output from signing process with extension `*.txt`), and the key to decrypt the encrypted file. In this process, the user is asked to input encrypted file documents, digital signature, decryption keys and then press the Verify button. After that, the application will verify the digital signature. The output of this process is a notification whether the verified decrypted document is same as the original. Following is a snippet from the source code of Class verification and the application view when the verification process is executed.

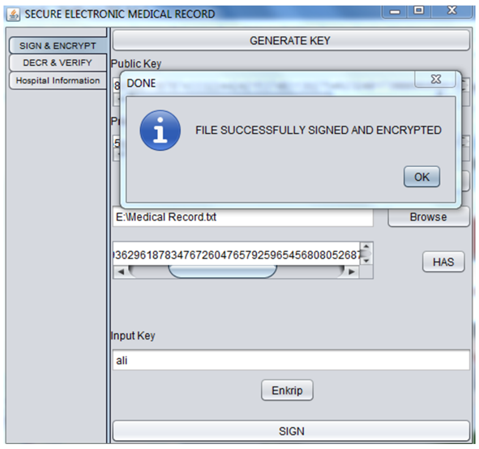
On verification process source code pieces and figure, it can be seen that in order to verify the documents it takes three documents which are the decrypted document, signature, and public key. If one among those there is not appropriate, then the document will not be verified. It can be stated that the verification failed.

```
Private void jButton10ActionPerformed
      (java.awt.event.ActionEvent evt)
   {
   String kunciaes=jTextField5.getText();
   try{
       pesandekrip=AES.decryptAES(pesanterenkrip, kunciaes);
   }
   catch (Exception ex){
       Logger.getLogger(latjab.class.getName()).
          log(Level.SEVERE, null, ex);
   }
   System.out.println("pesan hasil dekripsi = "+ pesandekrip);}
   private void jButton8ActionPerformed
      (java.awt.event.ActionEvent evt) {
       try{
          b.dekrip();
       }
       catch (IOException ex){
          Logger.getLogger(latjab.class.getName()).
              log(Level.SEVERE, null, ex);
       }
       if (b.v.equals(a.k)){
          JOptionPane.showMessageDialog(null,
              "DOCUMENT IS AUTHENTIC \n"
          + "FILE SUCCESSFULLY DECRYPTED \n",
          "VERIFIED", JOptionPane.INFORMATION_MESSAGE);
       } else {
          JOptionPane.showMessageDialog(null,
              "DOCUMENT IS NOT AUTHENTIC\n" +
          "FILE UNSUCCESSFULLY DECRYPTED \n", "UNVERIFIED",
              JOptionPane.ERROR_MESSAGE);
       }
    }
```

IV. RESULT

A. *Black-box Testing*

The black-box approach is a testing method in which the test data are derived from the specified functional requirements without regard to the final program structure [20]. It is also termed data-driven, input/output driven [21], or requirements-based testing [22]. In the current research, the black-box testing activities are presented in Tables I and II. The result of black-box testing shows that all output are exactly match the expected outputs.



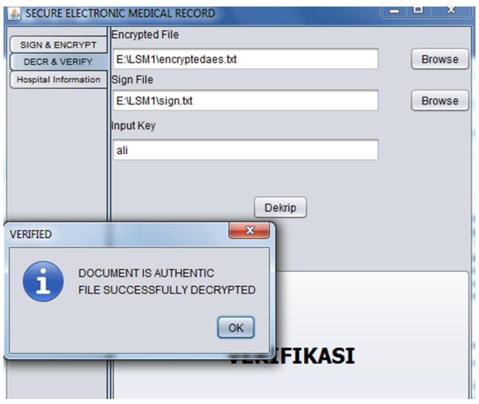
FILE SUCCESSFULLY SIGNED AND ENCRYPTED

E:\Medical Record.txt

3629618783476726047657925965456808052687

ali

Fig. 7. Signing and Encryption on SEHR Application.



E:\LSM1\encryptedaes.txt

E:\LSM1\sign.txt

ali

DOCUMENT IS AUTHENTIC
FILE SUCCESSFULLY DECRYPTED

Fig. 8. Verification on SEHR Application.

TABLE I
THE BLACK-BOX TESTING ON SIGNING AND ENCRYPTION.

No	Given Input	Expected Output
1	User input username, password, and press LOGIN button	The system checks the username and password. If the username and password are approperiate according to the database, it will go to the application
2	User press generate RSA KEY button	Application show key pair of RSA key
3	User press SAVE button	The system save key pair on the database
4	User press HASH button	The application show hash value of message and save it into database
5	User input key and press ENCRYPT button	The application show the key, encrypt the message with AES-256 bits and save the cipher text into database
6	User press SIGN button	The application will show notification that file is succesfully encrypted and signed. Then signature is saved into database

TABLE II
THE BLACK-BOX TESTING ON VERIFICATION.

No	Given Input	Expected Output
1	User press BROWSE button	The application show the encrypted and signed file.
2	User input key and press decrypt button	Application show the key, decrypt the file and saved it into database.
3	User press VERIFICATION button	The system show notification that file is succesfully decrypted and is authentic.

TABLE III
TESTING OF CORRECTNESS ASPECT

Filename	#lines of code
Folder/src	
`AES.java`	68
`RSA.java`	127
`SHA.java`	40
`Login.java`	150
`Main.java`	600
Folder/nbproject	
`Build-impl.xml`	1430
`Project.xml`	10
`Private.xml`	5
Total of Lines Code	2430
Total of KLOC	2.430

B. White-box Testing

The white-box testing is testing that takes the internal mechanism of a system or component into account [23]. In the development of the current software, the white box testing is done by using Kilo-Lines-of-Code calculation mechanism. For that, first, calculating the lines of code for each file that containing program code. Amount of code in each file are presented in Table III.

Doty Model is not used because total of code are less than 9000 lines. Folowing table are the results of correctness calculations testing using the Waltson-Felix, Bailey-Basili and Boehm methods in Table IV. The obtained error density value was 11.67, 7.54, and 8.29 with size of project less than 16,000 lines of code, then the value of the error density is in the range 0-40 per KLOC error as stated by Steve McConnell [24]. Therefore, it can be concluded that the application of Secure Electronic Health Record Using Java Programming Languages has meet the standards of software quality for correctness aspect.

TABLE IV
THE METHODS OF CORRECNESS CALCULATION [25].

Method	Fomulas
Walston-Felix Model	$E = 5.2 \times \text{KLOC}^{0.91}$
Bailey-Basili Model	$E = 5.5 + 0.73 \times \text{KLOC}^{1.16}$
Boehm simple Model	$E = 3.2 \times \text{KLOC}^{1.05}$
Doty model for KLOC > 9	$E = 5.288 \times \text{KLOC}^{1.047}$

V. CONCLUSIONS

This work has demonstrated the implementation of the encryption Secure Electronic Health Record. The implementation is performed using Java programming implementation and it has been tested. The result of black-box testing shows that all output exactly match what are expected. The white box testing shows that the obtained error density value was 11.67, 7.54 and 8.29 with the size of the project of less than 16000 lines of code.

REFERENCES

[1] M. H. Setiawan, "Perancangan secure electronic health record information system (studi kasus: Rumah sakit pusat angkatan darat gatot soebroto)," Bachelor Thesis, Sekolah Tinggi Sandi Negara, 2011.

[2] L. Wilcox, "Using the electronic medical record to keep hospital patients informed," *Sciences*, vol. 10, no. 4, 2010.

[3] J. L. Schnipper, J. A. Linder, M. B. Palchuk, J. S. Einbinder, Q. Li, A. Postilnik, and B. Middleton, ""smart forms" in an electronic medical record: documentation-based clinical decision support to improve disease management," *Journal of the American Medical Informatics Association*, vol. 15, no. 4, pp. 513–523, 2008.

[4] J. Spruell, D. Vicknair, and D. S., "Xxx," *XXX*, 2016.

[5] D. Garets and M. Davis, "Electronic medical records vs. electronic health records: yes, there is a difference," *Policy white paper. Chicago, HIMSS Analytics*, pp. 1–14, 2006.

[6] B. Kozier, *Praktik keperawatan profesional: Konsep dan perspektif*. Jakarta, Indonesia: EGC, 2007.

[7] R. Zhang and L. Liu, “Security models and requirements for healthcare application clouds,” in *2010 IEEE 3rd International Conference on Cloud Computing*. IEEE, 2010, pp. 268–275.

[8] *TS 18308 Health Informatics-Requirements for an Electronic Health Record Architecture*, ANSI ISO Std., 2004.

[9] *Recommendation for Transition the Use of Cryptography Algorithms and Key Lenghts.*, NIST Std. NIST Special Publication 800-131A, 2011.

[10] *Guideline for Implementing Cryptography In the Federal Government.*, NIST Std. NIST Special Publication 800-21A, 2005.

[11] NCH. (2006) Electronic health records overview. Healthcare Information and Management Systems Society. Download on October 15, 2011. [Online]. Available: http://www.himss.org/electronic-health-records-overview-nih-national-center-research-resources

[12] D. T. Mon, “Defining the differences between the cpr, emr, and ehr.” *Journal of AHIMA/American Health Information Management Association*, vol. 75, no. 9, pp. 74–5, 2004.

[13] WHO, *Electronic health records: manual for developing countries*. World Health Organization: Manila: WHO Regional Office for the Western Pacific, 2006.

[14] “Peraturan menteri kesehatan republik indonesia nomor 269/menkes/per/iii/2008 tentang rekam medis.”

[15] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography (1992 CRC Press)*. ISBN, 1997.

[16] S. P. Singh and R. Maini, “Comparison of data encryption algorithms,” *International Journal of Computer Science and Communication*, vol. 2, no. 1, pp. 125–127, 2011.

[17] Sumarkidjo and et al, “Jelajah kriptologi,” 2007, national Crypto Agency.

[18] W. Stalling, *Cryptography and Network Security*, 4th ed. Prentice Hall, 2005.

[19] R. Munir, “Otentikasi dan tanda tangan digital,” Departemen Teknik Informatika, Institut Teknologi Bandung., Tech. Rep., 2004.

[20] W. E. Perry, *A Standard for Testing Application Software, 1990*. Auerbach Publishers, 1989.

[21] G. J. Myers, C. Sandler, and T. Badgett, *The art of software testing*. John Wiley & Sons, 2011.

[22] W. C. Hetzel and B. Hetzel, *The complete guide to software testing*. John Wiley & Sons, Inc., 1991.

[23] *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std., Dec 1990.

[24] S. McConnell, *Code complete*. Pearson Education, 2004.

[25] R. S. Pressman, *Software engineering: a practitioner’s approach*, 7th ed. Palgrave Macmillan, 2010.