Performance Evaluation and Optimization of Digital Signature Component for ePedigree System

Yohanes Khosiawan, Seungwoo Jeon, Joonho Kwon[†] and Bonghee Hong

Department of Computer Engineering, Pusan National University
[†]Institute of Logistics Information Technology, Pusan National University
Jangjeon-dong Geumjeong-gu
Busan 609-735, Korea

{khosiawan,i2825t,jhkwon,bhhong}@pusan.ac.kr

To fight the counterfeiting of pharmaceutical products, many drug manufacturers are implementing ePedigree system using RFID technology. The ePedigree records all the distribution history of a product right from the start (manufacturer) until it reaches the consumer. But even the ePedigree information itself could be forged. Thus, the ePedigree contains digital signature(s) information, which is added respectively every time certain party appending information to the pedigree. The current algorithm used for the digital signature system is RSA. In this paper, we will use ECC instead of RSA. ECC algorithm for Digital Signature is called Elliptic Curve Digital Signature Algorithm (ECDSA), it has greater security and more efficient performance than the RSA algorithm.

Key Words: Digital signature, ECC, ECDSA, RSA, pedigree, RFID

1. INTRODUCTION

Nowadays, to protect the consumer from counterfeit products, many manufacturers use a system called Pedigree. Pedigree records all of the product's transaction histories resulting in a change of ownership along the distribution. There are 2 kinds of pedigree; they are paper pedigree and electronic pedigree – ePedigree. The ePedigree system allows us to distribute and track the product's transaction history via computer network connection easily. So far, there are several companies who offer commercial ePedigree software such as IBM [5], Oracle [8], and ePedigree Solutions [10].

In ePedigree, every time a party wants to append new information to the ePedigree document, it must include its digital signature too. Digital signature is an attribute in ePedigree which is responsible for the authenticity and integrity of the ePedigree content. This digital signature generation and authentication use the RSA algorithm as specified in the Pedigree Standard [9]. The same algorithm implementation proposed for EPCglobal architecture communication security also has been done in [3]. There is another algorithm for digital signature, called ECDSA. ECDSA performs faster and only needs much fewer bits than RSA to provide the equivalent security level. With shorter key size, there will be less bandwidth consumption and faster key distribution (during high network traffic). The comparison of Elliptic Curve and RSA key size for certain security levels can be found in [1]. In Section 2, we will discuss the overview of digital signature (especially ECDSA) and ePedigree. Section 3 describes the problem definition, and then solution would be presented in Section 4. At last, the experiment result and evaluation details would be explained in Section 5.

2. BACKGROUND

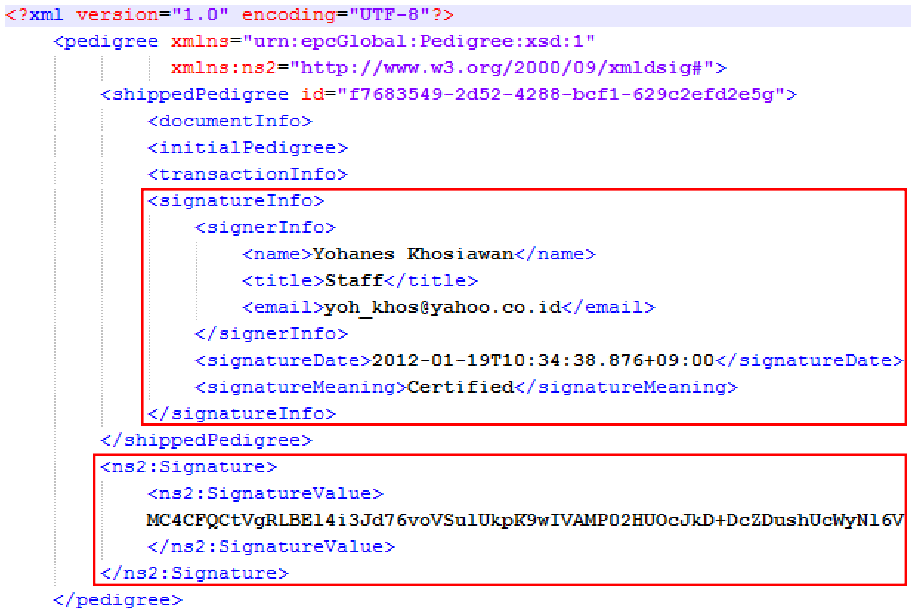
In this section, we will discuss the background knowledge, including the ePedigree, digital signature, ECDSA, and its security level.

2.1. EPEDIGREE

Based on [9], a pedigree is a certified record that contains information about each distribution of a prescription drug. It records the sale of an item by a pharmaceutical manufacturer, any acquisitions and sales by wholesalers or repackagers, and final sale to a pharmacy or other entity administering or dispensing the drug.

Rather than paper based pedigree, in this digital era, it's more effective using the digital version of pedigree, called ePedigree (electronic Pedigree). The document would be in an XML format based on the Pedigree Schema described in [9]. Generally, there are 6 types of ePedigree document involved: Initial Pedigree, Repackaged Pedigree, Shipped Pedigree, Received Pedigree, Unsigned Received Pedigree, and Received Pedigree.

The Figure 1 below is the ePedigree document cut of the "signatureInfo" and "Signature" element part.



```
<?xml version="1.0" encoding="UTF-8"?>
    <pedigree xmlns="urn:epcGlobal:Pedigree:xsd:1"
              xmlns:ns2="http://www.w3.org/2000/09/xmldsig#">
        <shippedPedigree id="f7683549-2d52-4288-bcf1-629c2efd2e5g">
            <documentInfo>
            <initialPedigree>
            <transactionInfo>
            <signatureInfo>
                <signerInfo>
                    <name>Yohanes Khosiawan</name>
                    <title>Staff</title>
                    <email>yoh_khos@yahoo.co.id</email>
                </signerInfo>
                <signatureDate>2012-01-19T10:34:38.876+09:00</signatureDate>
                <signatureMeaning>Certified</signatureMeaning>
            </signatureInfo>
        </shippedPedigree>
        <ns2:Signature>
            <ns2:SignatureValue>
            MC4CFQCtVgRLBE14i3Jd76voVSulUkpK9wIVAMP02HUOcJkD+DcZDushUcWyNl6V
            </ns2:SignatureValue>
        </ns2:Signature>
    </pedigree>
```

Figure 1: Digital Signature Component in the ePedigree Architecture

As we can see in Figure 1, the "signatureInfo", which is the signer's identity, is included in the "shippedPedigree" element that is signed afterwards. The email address contained in the "signatureInfo" element is used to retrieve the public key to verify the digital signature. Therefore, as stated before in Section 2, the public key is bound to a specific user's identity; hence guarantee the authenticity of the ePedigree document.

2.2. DIGITAL SIGNATURE ON EPEDIGREE

As its literal meaning, digital signature is a signature used for digital content. This signature is used to be represented in bytes that are appended to the respective digital content. Digital signature is generated using the sender's private key and verified using the sender's public key that could be downloaded from the key server by any user (the public).

Digital signature is used because of some important reasons [4]:

1. Authentication
   The ePedigree document contains user information which is used to retrieve public key from the Key Server. It is signed together with the other ePedigree contents, thus any alteration afterward would result an invalid digital signature. In this way, the public key is bound to a specific user. Hence, the digital signature would guarantee the authenticity of the ePedigree document's sender.
2. Integrity
   The user who receives the ePedigree document might doubt the information contained in the ePedigree document. But with this digital signature, when the content of the ePedigree document is altered, the signature verification is failed. So the user can trust the information as stated in the ePedigree document.
3. Non-repudiation
   Message digest encrypted with the signer's private key, binds the signer to the ePedigree document. Thus, the sender wouldn't be able to deny that he's the one who send the ePedigree document.

2.3. ECDSA (ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM)

ECDSA is a variant of digital signature algorithm that uses Elliptic Curve Cryptography. One thing we need to remember: both of the sender and receiver must agree upon the same elliptic curve parameter (q, FR, a, b, [$DomainParameterSeed$,]G, n, h).

In the Table 1 below, we can see the numbers of time units required to complete the given operations; if we assume that one 1024-bit modular multiplication requires one unit of time (t) [6].

Table 1 – Signing and verification process time (in custom time unit – t)

	ECDSA	**RSA 1024-bit**
Signing	60	384
Verification	120	17

2.4. SECURITY LEVEL

The security level of a cryptosystem is measured by how long it takes for a person to hack the secret key using a certain cryptanalysis technique. ECDSA classified as an Elliptic Curve Discrete Logarithm Problem (ECDLP). To solve ECDLP problem on ECDSA, we use Pollard Method Attacks that has $O = 2^{n/2}$ [7].

The time measurement unit used of course not in second, because it would be very inconsistent (based on the computer specification). Let us assume there are 1000000 instruction done in 1 second (1 Million Instruction Per Second), then in

ECDSA with n = 160 bits, the security level in Million Instruction Per Second (MIPS) is calculated as follows:

$2^{160/2} / (1000000*3600*24*365) \geq 3.8 \times 10^{10}$ MIPS *years*.

On the same key size, the security level of ECDSA is much better, if compared to RSA's as can be seen in [11].

3. PROBLEM DEFINITION

In RSA, for 1024 bit key size, ECDSA would only need 163 bit key size to provide the same security level. This means during the key distribution, by using ECDSA:

$$\frac{1024-163}{1024} \times 100\% \approx 84\% \text{ of bandwidth is saved.}$$

However, the amount of data calculation during the key distribution in the real world is not this straightforward; the detail implementation result would be explained in Section 5.

In the ePedigree work schema case, the performance time of ECDSA is much faster than RSA. Using the data from **Table 1**, we can calculate the amount of time needed when using RSA and ECDSA in the ePedigree system.

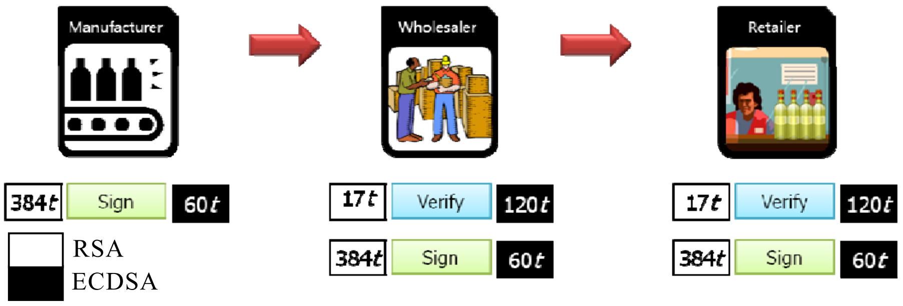


Figure 2: Time consumption comparison between the RSA and ECDSA implementation for digital signature in the ePedigree system work schema.

In the Figure 2 schema, we can see that RSA is consuming: 3x384*t*+2x17*t* = 1186*t*, while ECDSA is consuming 3x60*t*+2x120*t* = 420*t*. ECDSA key operation time for private key is fast, while the public key is slow. This means when the time consumed while verifying digital signature is longer than the one while generating the digital signature. From this point of view, ECDSA implementation has a balanced value of advantage and disadvantage. But since the Verification process (in Figure 2) is having less frequency than the Signing process, ECDSA still has the upper hand and offer significant improvement here.

4. PROPOSED SYSTEM

We will integrate the digital signature component using ECDSA implementation with the current ePedigree system. In other words, we are trying to substitute the current digital signature algorithm specification (RSA) in the ePedigree system.

As the Figure 3 shows, there will be a Key Server that will distribute the public keys. The Key Server will distribute the users' public keys to the ones who request it.

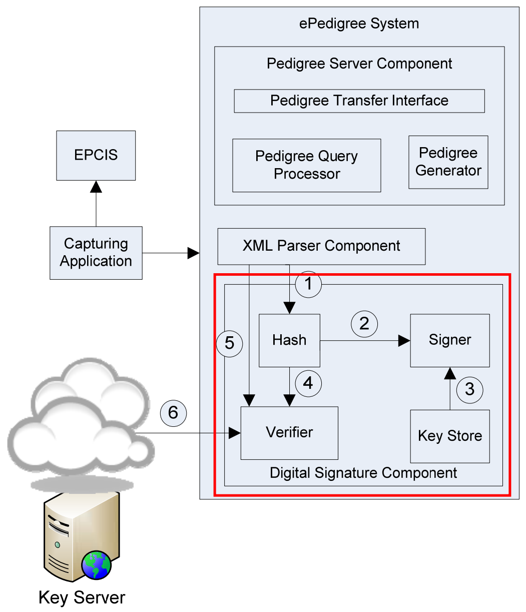


Figure 3: Digital Signature Component in the ePedigree Architecture

In Figure 3, we can see the architecture of the ePedigree system. The ePedigree system workflow when generating and verifying digital signature is like the following steps:

Digital Signature Generation and Verification

1. XML Parser Component decompose/separate the ePedigree information (plaintext) from the digital signature part (if it exist; in initialPedigree it doesn't exist). Then the Digital Signature Component receives the plaintext from the XML Parser Component, hashes it, and produces a message digest.

Digital Signature Generation

2. The message digest is sent to the signer and going to be signed later.
3. The message digest is encrypted using the private key of the sender. The private key is only owned and stored by the owner itself. Finally, the signing process will produce digital signature.

Digital Signature Verification

4. The message digest is sent to the verifier.
5. The digital signature part is also sent to the verifier from the XML Parser Component.
6. The digital signature is decrypted using the public key of the sender, which could be obtained from the Key Server, and producing a message digest. Finally, both message digests from the hash and the decryption process will be compared. If it is a match, then the digital signature is valid, otherwise it is invalid.

A scenario example of a sender (user A) and receiver (user B) is as follows. At the beginning, all users (A and B) have to register to its respective ePedigree system in EPCIS, and then the the Digital Signature component will generate a pair of public and private key. The Digital Signature Component will send the user's public key to the Key Server. After that, when A ships a product to B, A will append new shippedPedigree information to the product's ePedigree. When B receives the product, B will verify A's digital signature in the ePedigree using A's public key, which is obtained from the Key Server.

The signature generation process is called during the creation of Shipped Pedigree. The input could be Initial Pedigree, Repackaged Pedigree, Received Pedigree, or Unsigned Received Pedigree. The Digital Signature Component will read the ePedigree document and sign it in a way like the following Figure 4 pseudocode describes.

```
BEGIN input_pedigree_document
   Extract plaintext from the input_pedigree_document
   Generate message_digest
      Hash plaintext
   Generate digital_signature
      Sign (Encrypt) message_digest using private_key
   Generate shipped_pedigree_document
      Concatenate shipped_pedigree_element with the
                  digital_signature_element
END
```

Figure 4. Pseudocode of signature generation process

The digital signature verification process is invoked when we are handling an incoming Received Pedigree document. The algorithm is like the following Figure 5 pseudocode.

```
BEGIN input_pedigree_document
Extract plaintext from the input_pedigree_document
Extract digital_signature from the input_pedigree_document
 Generate message_digest
   Hash plaintext
 IF public_key is not exist in the program memory THEN
   Obtain public_key from the key_server
 Generate message_digest_2
   Decrypt the digital_signature using public_key
 Compare the message_digest to message_digest_2
   IF message_digest and message_digest_2 have the same value, THEN
        RETURN success_code
   ELSE
        RETURN failure_code
END
```

Figure 5. Pseudocode of signature verification process

The verification process only verifies the outermost digital signature element. It is because the previous signature(s) has/have been already verified by the respective receiver(s).

5. EXPERIMENTAL EVALUATION

5.1 Experimental Data and Environment

The ECDSA procedure in [2] involving key generation, signature generation, and signature verification will be implemented in Java using *SunEC* security provider. On the EPCIS side, we use Oracle XML DB that conforms to the ePedigree document schema specified in [9] to handle the XML document management.

We are evaluating the performance by implementing the Key Server using PHP server and MySQL as its database management system. We also add the new Digital Signature Component into the ePedigree system; this component will handle the communication with the Key Server, signature generation, and signature verification process.

The initial encoding of the public key is in an array of bytes format. To distribute it easily, we convert it into a Base64-encoded string. ECDSA public key only has an 88 characters length string, while RSA has 216 characters length string. This means ECDSA use around 59% less bandwidth than RSA. This would make the key distribution in a high traffic network faster; obviously in a non-high traffic network, 1 kB data in a common 100Mbps tunnel or higher is almost nothing.

In this experiment, we use 163 bits key for ECDSA and 1024 bit key for RSA; we are using these key sizes as a comparison based on the security level equality listed in [1]. In Figure 6, we can see that ECDSA (a) has a significantly lower signature generation time than RSA (b).

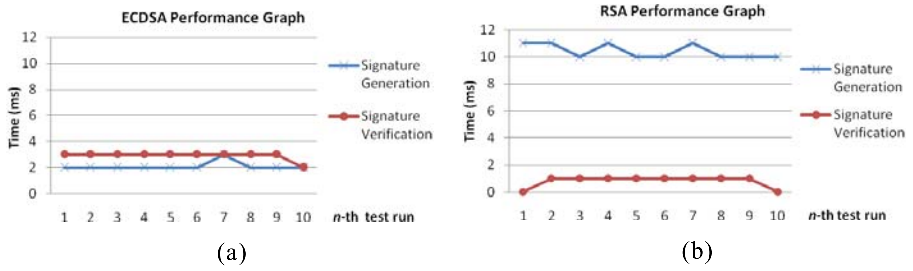


Figure 6. ECDSA and RSA Signature Operation Performance Graph

While the signature verification process in RSA has a slightly better performance time than ECDSA. In case of the ePedigree case, where the signature generation and verification process has an equal frequencies (in the common scenario, signature generation process actually has more occurances than verification process). Thus, the implementation of ECDSA for digital signature in ePedigree brings a significant performance improvement, especially to the computation time.

Based on Figure 6 data, Figure 7 shows the total time comparison of one signature generation and one verification process. It represents an approach of the ePedigree system scenario in Figure 2 as an equal-generation-and-verification-frequencies schema. It shows that ECDSA consistently and significantly outperforms RSA's performance.

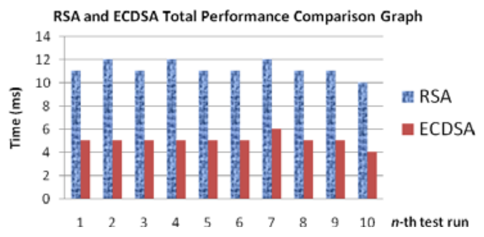


Figure 7. RSA and ECDSA Total Performance Comparison Graph

6. CONCLUSION

For the digital signature implementation in ePedigree, Elliptic Curve Digital Signature Algorithm provides greater security and more efficient performance than the first generation public key techniques (RSA) currently specified in [9]. ECDSA implementation shows around 50% performance improvement (based on Figure 7) on ePedigree system. And since the key size needed by ECDSA is much shorter than RSA at the same security level, it would save more bandwidth during the key distribution. For 1024 bit key of RSA, ECDSA need only 163 bit, around 80% less bandwidth used.

ACKNOWLEDGEMENTS

This work was supported by u-Food System Development Project of the Korea Food Research Institute.

REFERENCES

[1] C. Tony, B. Ayme, L. Marylin. 2010. Significantly Improved Performances of the Cryptographically Generated Addresses thanks to ECC and GPGPU. Computers & Security, Volume 29, Issue 4, June 2010, Pages 419-431.

[2] Johnson D., Menezes A., Vanstone S. 2001. The Elliptic Curve Digital Signature Algorithm (ECDSA). International Journal of Information Security, Volume 1, Number 1, 36-63.

[3] K. Dae-Jung, K. Jeong-Jai, L. Seung-Min, J. Moon-Seog. 2008. Design and Implementation for EPC System Method to Authentication and Cryptography. International Conference on Information Security and Assurance 2008.

[4] Gupta N. Kailash, Agarwala N. Kamlesh, Agarwala A. Prateek. 2005. Digital Signature: Network Security Practices. Prentice-Hall of India Provate Limited, New Delhi.

[5] IBM. IBM ePedigree Application for InfoSphere Traceability Server. <http://www-01.ibm.com /software/data/infosphere/traceability-server/epedigree.html> accessed on: August 2, 2012 – 02:19 GMT.

[6] M.J.B. Robshaw, Yiqun Lisa Yin. 1997. Overview of Elliptic Curve Cryptosystems. RSA Laboratories.

[7] Menezes A., Teske E., Weng A. 2004. Weak Fields for ECC. The Cryptographers' Track at the RSA Conference (CT-RSA) 2004.

[8] Oracle. 2010. Combating Counterfeit Medicine: Enabling Mass Serialization and Pedigree Management in the Pharmaceutical Industry. Oracle White Paper.

[9] Pedigree Standard v.1.0, EPCglobal Inc., 2007. <http://www.gs1.org/gsmp/kc/epcglobal/pedigree/pedigree_1_0-standard-20070105.pdf> accessed on: March 10, 2012 – 10:40 GMT.

[10] RxServer. 2009. ePedigree Software – Wholesaler Software – ePedigree Solutions. <http://epedigreesolutions.net/> accessed on: August 2, 2012 – 02:36 GMT.

[11] Schneier, Bruce. 1996. Applied Cryptography, second edition. John Wiley & Sons, Inc.