ECDSA and Public Key Infrastructure for Verifying Scientific Articles Integrity and Authors

Fairuz Astra Pratama – 13514104
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Jl. Ganesha No.10, Lb. Siliwangi, Coblong, Kota Bandung, Jawa Barat 40132, Indonesia
13514104@std.stei.itb.ac.id, pratamafairuz@gmail.com

***Abstract* – Following the widespread usage of the Internet, digital publication of scientific articles or papers have become a common occurrence. While enabling a much easier way of distributing information, this method also make it difficult to verify the published articles authors and integrity, since the usual stamp or signature no longer work on the digital space. As a replacement, digital signature and public key infrastructure can be used, and this paper will propose a method of using ECDSA and Certificate Authority to verify articles authors and integrity.**

Keywords — ECDSA, Digital Signature, Documents Integrity, Verifying Authors, Certificate Authority

I. Introduction

Following the widespread usage of the Internet as a medium for information dissemination, more and more scientific articles, journal and grey literature are published digitally. This is mainly done since a printed scientific articles and/or journals are slower and more expensive to distribute. Digital publishing of scientific articles can also accelerate the peer-review and editing process compared to the printed version. Furthermore, it also makes the journal content and the related supplementary data more accessible. Researcher in a developing country can access a scientific paper by using the many digital library such as ieeexplore.ieee.org instead of searching and buying the printed version.

One of the weakness of this method is that, as with most content that is published in the internet, it may be difficult to identify and verify the owner / publisher of a scientific paper, as well as the integrity of said paper. This is especially true for grey literature, which is a scientific paper that is not published by a traditional publisher, such as government's research report or independently published paper. In the physical work, owner and content integrity can be verified by checking the owner signature or a governmental stamps in the paper. Unfortunately, since digital images can be easily manipulated, edited, and added, this method will not work very well on the digital media, especially if the access to the paper is limited.

For example, imagine that someone wants to purchase a scientific report produced by a trusted organization from a third party seller. The buyer and the seller may have difficulties proving that the paper is written by the author attributed in the document, and that the content haven't been tampered with. One method that can be used to solve this is by using digital signature and public key infrastructure.

Digital signature works by using asymmetric encryption to encrypt a message digest using the owner private key. This way, the data integrity and owner can be validated by decrypting the encrypted digest using the owner public key, recalculating the message digest value, and comparing both value. One of many method to do this is called Elliptic Curve Digital Signature Algorithm (ECDSA), which uses Elliptic Curve as the asymmetric encryption algorithm.

On the other hand, public key infrastructure (PKI) is a way of binding a public key to an individual, or an organization. PKI provides a methodology for user to validate that a public key truly belongs to a particular entity. This way, user can safely use the key for encrypting message to, or validating digital signatures from the owner of the public key.

Digital signature, along with public key infrastructure (methodology for binding a public key value to an individual or an organization) is used in many aspect of the internet as a way of authenticating the owner and verifying the integrity of files. This paper will propose a method of using ECDSA and Public Key Infrastructure to verify a scientific paper integrity and the owner / publisher identity, as well as analyzing the effectiveness of the proposed method itself.

II. Literatures Study

A. *Elliptic Curve Cryptography*

Elliptic Curve Cryptography (ECC) is one of the newest asymmetric cipher algorithm. ECC is often claimed to provide a stronger security than RSA for the same key length, where a 160-bit ECC key is said to have the same security level as a 1024-bit RSA key. ECC is based on addition and scalar multiplication of points that exist in an elliptic curve on a Galois Field GF(p). The curve is often represented by the equation: $y^2 = x^3 + \mathbf{a}x + \mathbf{b} \bmod \mathbf{p}$. An example of a continuous curve with the equation $y^2 = x^3 - 4x$ can be seen in Figure 1, where the discrete (on a Galois Field) version can be thought of as a sparse collection of point that's located along the red line. Several important point operation in an elliptic curve on a Galois Field that is often used are processed as such:

1. Adding two points: P + Q
   $P(Xp, Yp) + Q(Xq, Yq) = R(Xr, Yr)$
   $\lambda = (Yp - Yq) / (Xp - Xq) \bmod \mathbf{p}$
   $Xr = \lambda^2 - Xp - Xq \bmod \mathbf{p}$
   $Yr = \lambda * (Xp - Xr) - Yp \bmod \mathbf{p}$

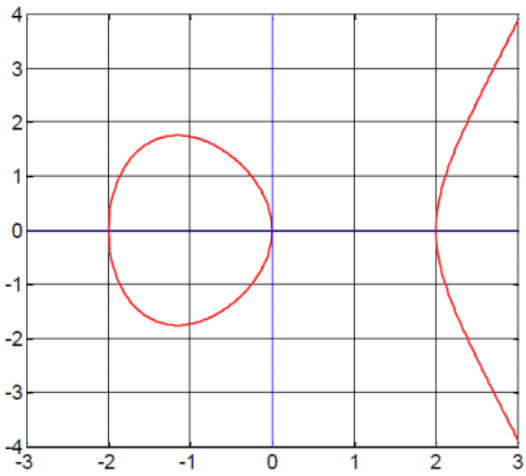

Figure 1 Example of Continuous Elliptic Curve (Andreas Steffen, Elliptic Curve Cryptography)

2. Subtracting two point: P - Q
   P(Xp, Yp) - Q(Xq, Yq) = P + (- Q)
   -Q = (Xq, -Yq mod **p**)

3. Point Duplication: 2 . P
   2 . P(Xp,Yp) = R(Xr, Yr)
   $\lambda = (3 * Xp + \mathbf{a}) / (2 * Yp) \bmod \mathbf{p}$
   $Xr = \lambda^2 - (2 * Xp) \bmod \mathbf{p}$
   $Yr = \lambda * (Xp - Xr) - Yp \bmod \mathbf{p}$

4. Point Iteration: k . P
   k . P(Xp,Yp) = Q(Xq, Yq)
   3 . P = (2 . P) + P
   4 . P = (2 . (2 . P))
   5 . P = (2 . (2 . P)) + P
   6 . P = (2 . ((2 . P) + P))
   7 . P = (2 . ((2 . P) + P)) + P
   Etc...

The strength of ECC comes from the fact that given a random integer **k** and a point **P** in an elliptic curve on Galois Field GF(p) where **Q** = **k** . **P**, **Q** can be easily be computed. Yet, when given a point **Q** and a point **P** on the same equation, the value of **k** is almost impossible to calculate. This problem is called as Elliptic Curve Discrete Logarithm Problem, and are the basis of many cryptography function such as ECDSA.

B. Elliptic Curve Digital Signature Algorithm

ECDSA is a variant of the US standardized Digital Signature Algorithm. While the original DSA had it strength from the discrete logarithm problem, ECDSA derived its strength the Elliptic Curve Discrete Logarithm Problem. In general, the usage of ECDSA consist of four main steps:

1. Specifying the Elliptic Curve Parameter

   Before two participant can send and verify a signed message, they must first agree on the Elliptic Curve Parameter. These parameters consist of:

   a. The elliptic curve equation used ($y^2 = x^3 + \mathbf{a}x + \mathbf{b}$) where both **a** and **b** must be specified
   b. The curve above will be defined at a Galois Field GF(**p**) The value of prime number **p** that represent the field size must also be defined.
   c. Base point **G(x0,y0)**, that exist in the above curve
   d. The order of base point **G**, where **n** . **G(x0,y0)** = $O(x,\infty)$ in the curve above ('.' operation represents an elliptic curve point multiplication by a scalar value)

   Calculating and verifying the curve parameters above are difficult and time-consuming. This is why most of the times, a standardized and commonly-used curve parameter such as secp256k1 or NIST P-256 are chosen in this step.

2. Generating the ECDSA Key Pair

   ECDSA key pair is identical to most ECC key pair which consist of a random integer **a** between [1, **p**-1], and point **pa** where **pa** = **a** . **G.** Integer **a** is the private key, where the point **pa** is the public key. The private key must be saved securely, and are used by the key owner to sign a document, while the public key can be sent through unsecure channel, and are used by other participant to verify the owner signature.

3. Making the ECDSA Signature

   To sign a message, document or data **M**, a participant with key pair (**a, pa)** and the curve parameter (**a**, **b**, **p**, **G**, **n**) must do the following:

   a. Calculate the message digest **m** = HASH(**M**) using one of the commonly used cryptographic hash function such as SHA-1 or MD5
   b. Generate a random number **k** between [1, **p**-1]
   c. Calculate the point (**x1,y1**) = **k** . **G** in the specified curve
   d. Compute **r** = **x1** mod **n**.
      If **r** = 0, go back to step 'b'
   e. Compute $\mathbf{s} = \mathbf{k}^{-1} * (\mathbf{e} + \mathbf{r} * \mathbf{a}) \bmod \mathbf{n}$.
      If **s** = 0, go back to step 'b'
   f. The value (**r**, **s**) is the signature for the message **M**, this value can be attached to the message or made available through some other means so that the message receiver can verify the sender and message integrity.

4. Verifying ECDSA Signature

   To verify a message, document or data **M** with signature (**r, s**) from a participant with public key **pa** and the curve parameter (**a**, **b**, **p**, **G**, **n**), the recipient must do the following:

   a. Verify that **r** and **s** are between [1, **p**-1]
   b. Calculate the message digest **m** = HASH(**M**) using the same cryptographic hash function used during signing
   c. Generate a random number **k** between [1, **p**-1]
   d. Compute $\mathbf{w} = \mathbf{s}^{-1} \bmod \mathbf{n}$
   e. Compute **u1** = **z** * **w** mod **n**
   f. Compute **u2** = **r** * **w** mod **n**
   g. Calculate the point (**x1,y1**) = (**u1** . **G)** + (**u2** . **pa)** in the specified curve ('.' operation represents an elliptic curve point multiplication by a scalar value and '+' operation represents an elliptic curve point addition)

h. The signature is valid if **r = x1** mod **n**. If the signature is valid, it means that the sender have the private key **a** and the message hadn't been tampered by a third party.

C. *Public Key Infrastructure*

In asymmetric encryption system such as ECC and ECDSA, key management is one of the most important aspect to consider. In short, there are two requirements that needs to be fulfilled when managing key pair: making sure that a private key remains secret from everyone except the owner, and guaranteeing that a public key belong to a specific entity. Public Key Infrastructure (PKI) are a mechanism to fulfill the second goal, giving an assurance that a certain key belong to a certain entity, as well as distributing this information.

There are many approaches of creating this PKI system, one of the more commonly used one is called Certificate Authorities. In this system, the declaration that a key pair belong to a specific entity are contained in what's called as Digital Certificate. This certificate contains information such as the owner name, email, key pair algorithm, as well as the public key, binding said key to the owner listed.

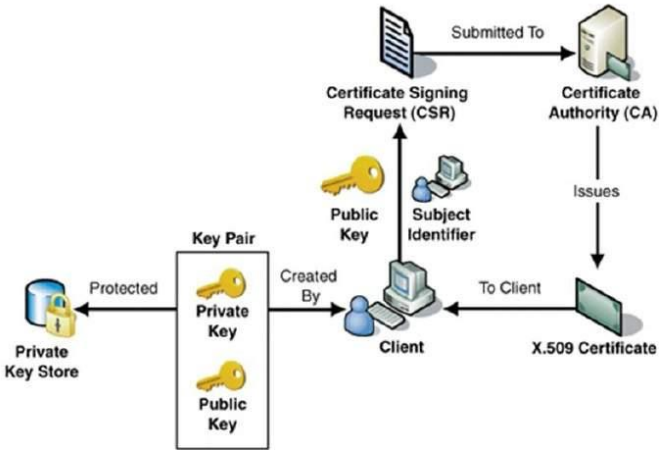

Figure 2 Illustration of Certificate Issuing (Tutorials Point, Public Key Infrastructure)

Alongside it, there are entity called Certificate Authorities (CA), which had a role of verifying, and publishing Digital Certificate. It does this by identifying the requester identity, generating the Digital Certificate, signing the Certificate using the CA's secret key, and distributing the signed Digital Certificate. This way, the validity of the certificate and the information within it (binding between key pair and owner) can be verified with the CA's digital signature. The way CA issues a certificate can be seen in Figure 2; user send his/her public key along with his/her identifier to CA, the CA verify the user identity, and returned a signed digital certificates with the user's information inside it.

Naturally, depending on a single CA for verifying every public key used in the world is not feasible. The solution for this is called hierarchical certification. In short, this method allows a CA to "trust" another organization by making a CA-Certificate of the organization public key, signed by the CA private key. This certificate signify that an organization had the trust and the authority to act as a Certificate Authority. The new CA can also choose to "trust" another organization, forming hierarchical tree of trust. An example of this CA hierarchy can be seen in Figure 3, where a trusted Root CA delegate its "trust" to three separate subordinate.

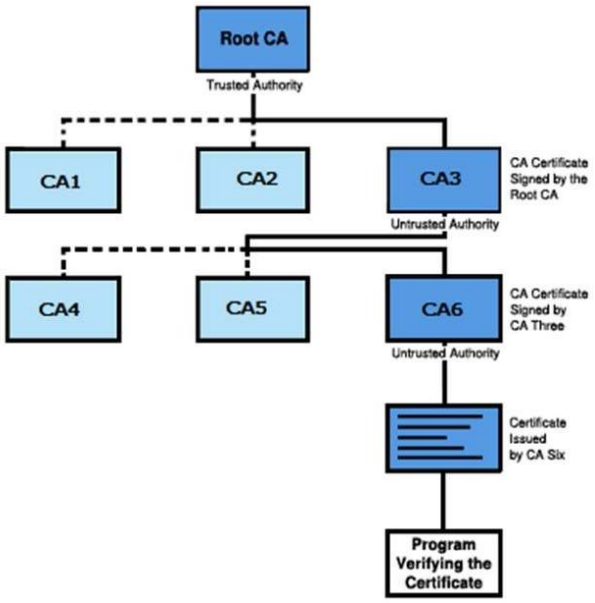

Figure 3 Illustration of Certificate Authority Hierarchy (Tutorials Point, Public Key Infrastructure)

III. Proposed Solution

This section will propose a method of using ECDSA to sign a scientific document, as well as a method of binding the documents author, publisher, and the document itself using a digital-certificate-like document.

A. *Binding a Document to a Public Key and its Authors*

A digital certificate is a document that binds a public key to an entity that is "signed" by an authority figure called Certificate Authorities as a proof of its validity. The entity that a particular key is bind to can be anything, from an individual, an organization, even a server. The certificate can also contains many additional field that provide more information regarding the public key owner listed; these information are also bound to the certificate owner by the CAs signature. An example of a digital certificate issued to bind a public key to a web server can be seen in Figure 4. This method can also be used to bind a scientific articles to its property and a public key. By creating a certificate that contains the papers name, authors, publisher, publishing date, and a public key, we can bind these values together by having an authority figure signing said certificate.

While the usual CA that exist in the internet may serve as the authority for signing scientific paper certificates, it may be more suitable to having the paper publisher (or an organization that collect scientific papers that is not published traditionally) sign it. This is more appropriate since most CA that exist in the

internet usually only works on verifying an individual or organization identity, and are not accustomed to verifying a documents identity that is needed to create a digital certificate of a scientific paper.

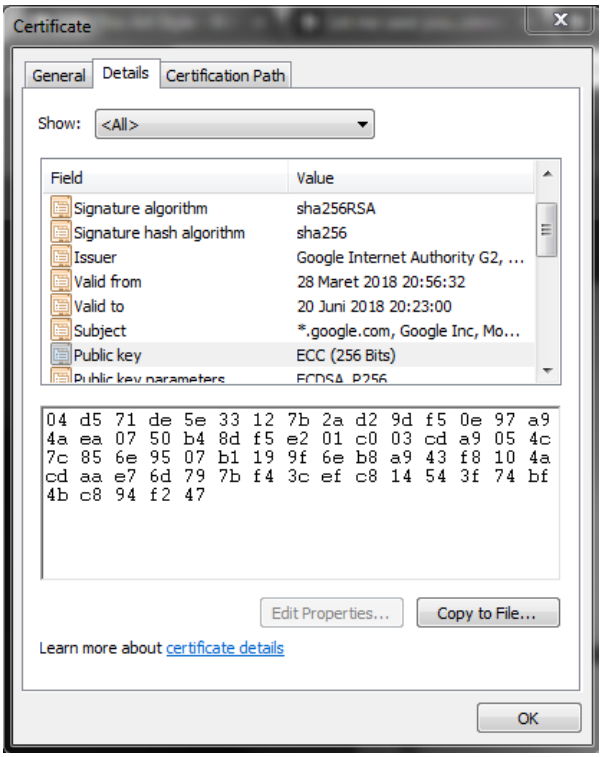

Figure 4 An Example of Digital Certificate

Publisher can make and sign a documents certificate using its private key to vouch its validity, where the publisher's public key is available in a digital certificate provided and signed by another CA, making this method similar to a CA Hierarchy, and the publisher can be thought of as a Certificate Authority for all documents that it has published. This way, the protocol of publishing a scientific articles online will be similar to the usual certificate issuing protocol in Figure 2 and can be done using this method:

a. The articles author / owner will send the papers that he/she want to publish along with any additional properties needed (such as the authors name, and a public key)
b. Before publishing it in the internet, the organization / publisher will create a certificate-like document that contains the paper information such as the papers ID, name, authors, and the papers public key
c. The organization will then digitally-sign the papers certificate, and making it accessible online so that anyone can verify the papers integrity and authors
d. Lastly, the paper itself will be published digitally

After using the methodology above, a digitally published scientific papers will be bound to its author, a public key, and its other properties. The documents public key can be used for many things, and are safe to use since the private key will be stored safely by the documents author / publisher and the public key validity can be verified by checking the publisher signature on the papers certificate. For example, the owner of the documents public key can use the key pair to create the documents digital signature using the method discussed in the next section.

B. Signing Published Scientific Articles

By having a secret private key and since the public key is mad publicly available and verifiable in the documents certificate, the owner of the documents private key can use it to calculate the documents signature. One of the ways to produce a document digital signature, is by using ECDSA algorithm to create a signature value using the documents key pair. The owner of the documents private key can create the documents ECDSA signature using this method:

a. Convert the document into standard read-only format that is ready to be published digitally such as the PDF format
b. Generate a ECDSA key-pair, use the private key to calculate the documents ECDSA signature
c. Send the document to the publishing organization along with the public key and any additional information. The publisher will the sign all that information into a certificate, and publish it along with the document

After doing the steps above, all that's left is to choose a method of distributing the documents signature. In most digital signature use case, the signature is sent along the message, so it is possible to just add the value to the document itself. The problem with this, is that by adding additional data to the document, the value of the documents message digest, and therefore, the documents signature will change. This is why it is be a better idea to add the documents signature to the documents certificate that will be signed by the publisher. This way, all the information that is needed to verify the documents integrity and authors are contained in a single certificate that is verified by the documents publisher.

C. Verifying a Signed Published Scientific Articles

Verifying a published scientific articles signed using ECDSA and the certificate scheme above is quite simple. An individual who wants to verify a documents integrity and authors, can do that using this method:

a. Search for the documents certificate from the publishers website for the documents of interest using its name or id
b. Validate the certificate by checking the publishers digital signature using the publishers public key (the key can be acquired from the publishers digital certificate signed by a certificate authority)
c. If valid, then validate the documents signature listed in the certificate, against the documents itself using the documents public key listed in the certificate
d. If the documents signature is valid, then the document certificate, and all the information within (including the authors) are bound to the document itself, and the documents integrity and authors is verified.

IV. IMPLEMENTATION

This section will consist of the implementation result of the document signer and document certificate scheme proposed in the previous section. This section will also examine the usability and security aspect of the implemented mechanism in fulfilling its purposes; providing a method of verifying a documents integrity and authors.

A. Implementation of the Documents Signer

The document signer is implemented using the Python programming language. User will interact with the application using a Command Line Interface (CLI), where the key pair, and signature value will temporarily be printed on screen for the user to copy and safe elsewhere in whatever format the user wishes. The implemented application will allow the user to do the following action:

1. Generating an ECDSA Key Pair. The key will be printed on screen temporarily so that the user can copy and save it to a safe location.
2. Generate a PDF Documents Signature. This action will need the file path and a private key value that is generated by the previous action
3. Validate a PDF Documents Signature. This action will need the file path, signature value that is generated by the previous action, as well as the public key that is generated by the first action above.

In this implementation, the message digest that is used to generate the signature value is calculated using the **MD5** algorithm which is a commonly used cryptographic hash function. Besides that, the curve parameter (**a**, **b**, **p**, **G**, **n**) that is used in this implementation is curve **secp256k1**, which is one of the standard curve that is defined in the Standards for Efficient Cryptography (SEC). Curve secp256k1 has a large field prime **p** and a large number of valid points, making it suitable to be used in ECDSA. The curve parameter of the secp256k1 curve are as follow:

Table 1 Secp256k1 Curve Parameter

a	0
b	-7
p	115792089237316195423570985008687907853 269984665640564039457584007908834671663
G.x	550662630222773436695787188951685343262 506034537775941755001873603891167292 40
G.y	326705100207588169780830851305070431844 712733806592432759389043357573374824 24
n	115792089237316195423570985008687907852 837564279074904382605163141518161494337

B. Implementation of the Documents Certificate

The document certificate generator will be implemented as additional feature of the document signer. This means that this feature is implemented using the Python programming language and user will use this feature through the Command Line Interface. This features will allow the user (a publishers or an organization) to do the following action:

1. Generating a documents certificate, which is a signed certificate that contains the title, authors, publisher, and other bibliographical properties of the documents alongside the documents signature value, and the public key of the key pair used to generate that signature.
2. Validate a documents certificate by using the signature in the certificate as well as the public key of the certificate creator (publisher) that is procured from a trusted source (such as the publishers digital certificate from a CA)

In this implementation, similar with the document signer specified in the previous section, the certificate digital signature will be calculated using the **MD5-ECDSA** method, where **secp256k1** will also be used as the curve parameter. Additionally, the certificate that is generated will be using an arbitrary format that can only be used within this application, and not using a standard digital certificate format such as x.509. This is done in order to ease the implementation and experiment process. The main screen for the completed program can be seen in the picture below, where the other screens will be displayed in the next section.

```
Welcome to Document Signer!

What can we help you with today?
(1) Create a New Key Pair
(2) Generate a PDF Documents Signature
(3) Validate a PDF Documents Signature
(4) Generate a Documents Certificate
(5) Validate a Documents Certificate
(6) Quit
>
```

Figure 5 Programs Main Menu

C. Usage Examples

Imagine that Alice had a well-known publishers company that is known as Alice Publishing Company. Alice's company had just decided to publish Bob's scientific journals, and had decided to secure the documents validity and authors by using the method that is proposed in this paper. For demonstration purposes, imagine that bob's paper is the standard IEEE paper format available at http://ieeeauthorcenter.ieee.org/wpcontent/uploads/JTEHM_Template.doc

After receiving the documents above in pdf format, Alice will first generate an ECDSA key pair for it. Using the application Alice can easily generate a random key pair by entering option '1', and save the key pair printed on screen as a TXT format safely on one of the company's computer.

Table 2 Generated Documents Key Pair

d	000915552695781592191956091570764571997000759931143289043466234426984991330962577
pd	(10331066632984859484770389081392043975719015462186402365445576476578083302959 7, 19015462186402365445576476578083302959 7, 00000775912414703938855256120004112068376392032146281567171908420818484737441 20, 63920321462815671719084208184847374412 0)

After generating the documents key pair, Alice will then generate the documents digital signature by entering option '2'. By entering the documents path and the private key **d** above as the parameter, Alice will receive the documents digital signature value. The screenshot of the program and the signature value itself can be seen in **Figure 6** below.

```
 Generating a PDF Documents Signature...
Documents path     : D:/JTEHM_Template.pdf
Private key        :
 915552695781592191956091570764571997000759931143289043466234426984991330962577

Calculating Signature...
Signature     :
 (107520822122844550847742083178056867965536055567968132161814007179754339038468,
  555452524586811136049774156151457139629757232987366324608912682730436986851 60)
Press anything to continue..
```

Figure 6 Generating the documents signature

Finally final step before Alice can publish Bob's article is generating the documents certificate using option '4' of the program. This option will give out prompt for Alice to enter the documents properties, such as its name, authors, along with the documents public key and its signature. Lastly the program will ask for the private key that will be used to sign the certificate, and where to save the certificate. In this example, Alice's company is assumed to have a key pair already distributed and verified via a Certificate Authorities. Alice's company key pair value are as follow:

Table 3 Alice's Company Key Pair and Documents Signature

a	00101874429999621238319993065864992091868741380112981621963259908608203855243028
pa	(02900259504183420779384725912052490121716992535345180747091040798943767992407 8, 00071624393183013992277809728232878811141365503142273439841212989986229620297 80)
ds	(107520822122844550847742083178056867965536055679681321618140071797543390384 68, 00555452524586811136049774156151457139629757232987366324608912682730436986851 60)

After generating the certificate, Alice will received a certificate that contains all the documents information, followed by her digital signature. Alice will then needs to make the certificate easily accessible online, before publishing the Bob's scientific journal digitally.

```
--------------------BEGIN-DOCUMENTS-CERTIFICATE---------------------
Documents ID           : 13514104
Documents Name         : Bob's Big Book
Documents Signature    :
(107520822122844550847742083178056867965536055679681321618140071797543390384 68,
555452524586811136049774156151457139629757232987366324608912682730436986851 60)
Documents Public Key   :
(10331066632984859484770389081392043975719015462186402365445576476578083302959 7,
77591241470393885525612000411206837639203214628156717190842081848473744120)
Publisher              : Alice's Publishing Company
Authors                : Bob, Bobby
---------------------END-DOCUMENTS-CERTIFICATE----------------------
(10276927553269154483147265915004401403673225815901843878018569602379891133112 5,
40945172090941006016653762480823110466505357133236058747529850675684315416602)
```

Figure 7 The Documents Digital Certificate

Now, suppose that someone, named Carol had received a digital copy of Bob's journal in pdf format from one of her friend as some sort of payment or gift. After a brief research in the internet, Carol know that Bob's journal is published by Alice's company. She wants to know if her copy of Bob's journal is a valid copy, but maybe the document is sealed behind a pay-wall, or maybe the document itself is too big to be checked manually. To solve this problem, by using the program implemented in the previous section, carol can easily verify her documents integrity and its authors.

First, Carol will need to get the documents certificate that correspond to Bob's journal that has been distributed by Alice. Next, she will need to procure Alice's company public key via using a trusted method such as Certificate Authorities. Next, Carol needs to verify Alice's signature on the certificate to verify that she is the one who made this certificate. This can be done using option '5' of the program, and the process can be seen in the picture below:

```
 Verifying a Documents Certificate...
Certificate path  : D:/cert.txt
Public Key        :
  (290025950418342077938472591205249012171699253534518074709104079894376799240 78,
  716243931830139922778097282328788111413655031422734398412129899862296202978 0)

Validating Signature...

The Certificate Signature is Valid!!
Press anything to continue..
```

Figure 8 Verifying a Documents Digital Certificate

Since the certificate is proved to be valid, the information within it can be trusted to be true since it came from the documents publisher (Alice). Using this method, Carol can be sure that Bob is indeed the one who wrote the journal that she is interested in. Next, Carol will verify the documents signature using the public key and the 'true' signature listed in the certificate. This is done using option '3' of the program, and the process can be seen in the picture below:

```
 Validating a PDF Documents Signature...
Documents path     : D:/JTEHM_Template.pdf
Public Key         :
  (10331066632984859484770389081392043975719015462186402365445576476578083302959 7,
   77591241470393885525612000411206837639203214628156717190842081848473744120)
Signature          :
  (107520822122844550847742083178056867965536055679681321618140071797543390384 68,
   555452524586811136049774156151457139629757232987366324608912682730436986851 60)

Validating Signature...

The Document Signature is Valid!!
Press anything to continue..
```

Figure 9 Verifying the Documents Signature

Since the documents signature match the 'true' signature listed in the documents certificate, Carol can be certain that the documents she own are a valid copy of Bob's journal.

V. Experiments Results and Analysis

Continuing from the usage example in the previous section, this section will explore some of the attack method that can be done against a document integrity and its author's identity. Some of the possible attack method can happen and will be discussed further are as such:

1. Publishing an altered version of a scientific articles that claims to be the original version
2. Falsely claiming to be one of the authors
3. Publishing a fake document certificate alongside an altered version of the scientific articles

For each attack method, the method of repelling said attack using the solution proposed in this paper will be tested. This way, the solution validity as a method of ensuring a published documents integrity and verifying a documents author can be proven.

A. *Detecting Documents Alteration*

Let's say that Bob has a rival called Eve, who wants to tarnish Bob's reputation by publishing an altered version of Bob's journal that claims to be the real one. For this experiment, we will use the documents from the previous section and change the title from "Preparation of Paper..." to "Preparation of Peppers..." The altered document will then be converted into a pdf that can now be published as a fake version of Bob's journal that is published by Alice.

Now, suppose that someone, named Dave had received a digital copy of this fake journal that claims to be Bob's journal the same way Carol received hers. Just like Carol, Dave will then search and get Bob's journal certificate that is published by Alice, and verify said certificate. As a reminder, all parameter in the certificate such as the documents public key, Alice's key pair, and the 'true' documents signature are the same as the one used in the previous section. Next, he will try and verify the documents integrity by using the listed 'true' signature and documents public key in the certificate.

```
 Validating a PDF Documents Signature...
Documents path    : D:/JTEHM_Template_fake.pdf
Public Key        :
  (10331066632984859484770389081392043975719015462186402365445576476578083302959 7,
  77591241470393885525612000411206837639203214628156717190842081848473744120)
Signature         :
  (10752082212284455084774208317805686796553605556796813216181400717975433903846 8,
  55545252458681113604977415615145713962975723298736632460891268273043698685160)

Validating Signature...

WARNING! The Document Signature is NOT Valid!!
Press anything to continue..
```

Figure 10 Verifying an Altered-Documents Signature

By changing the content of Bob's journal, Eve had also altered the message digest value, thus making the verification process failed. Since the document signature didn't match with the 'true' signature, Dave now know that the journal he owns are not a valid copy of Bob's journal that Alice had published.

B. *Disproving False Claims*

Suppose that Eve now claims to Carol that she was one of the authors of Bob's journal. Traditionally, the way to confirm this claim is for Carol to ask the document publisher (Alice) or one of the writer. Using the method given in this paper, there are a much better and faster way to do this.

Besides containing the digital signature and the public key used for a particular articles, the documents certificate can also be filled with additional information. In this example, the certificate created and signed by Alice's company contains the documents author list. By retrieving said certificate, Carol can disprove Eve's claim, since her name is not listed in the article's certificate, which means that her involvement in Bob's journal was not acknowledged by the journal publisher.

C. *Detecting False Certificate*

The last kind of attack method is publishing a fake or altered document certificate. By using this method, an attacker can change any information regarding a particular documents, including its signature value. By having the ability to change the signature value and the related public key, the attacker can also change the document content as he/she pleased, and change the signature value accordingly.

Let's say that Eve wants prove her claim of being one of the authors for Bob's journal. She may download the document's certificate that is published by Alice and add her name to the author list. Furthermore, she may recalculate Bob's journal signature using a random key pair after altering its content. Luckily, since every documents certificate carry the signature of its publisher, there's a way to repel this attack

```
--------------------BEGIN-DOCUMENTS-CERTIFICATE----------------------
Documents ID            : 13514104
Documents Name          : Bob's Big Book
Documents Signature     :
(107520822122844550847742083178056867965536055567968132161814007179754339038468,
55545252458681113604977415615145713962975723298736632460891268273043698685160)
Documents Public Key    :
(103310666329848594847703890813920439757190154621864023654455764765780833029597,
77591241470393885525612000411206837639203214628156717190842081848473744120)
Publisher               : Alice's Publishing Company
Authors                 : Bob, Eve
---------------------END-DOCUMENTS-CERTIFICATE----------------------
(102769275532691544831472659150044014036732258159018438780185696023798911331125,
40945172090941006016653762480823110466505357133236058747529850675684315416602)
```

Figure 11 Altered Documents Digital Certificate

Now, suppose that Dave received this altered / fake documents certificate of Bob's journal. Before using the information within the certificate, he must first verify that the certificate is made by a trusted and authorized party, which in this case is Alice, the publisher of Bob's journal. After getting Alice's public key, he will try and verify the certificate authenticity using one of the program's feature.

```
 Verifying a Documents Certificate...
Certificate path  : D:/cert_fake.txt
Public Key        :
  (29002595041834207793847259120524901217169925353451807470910407989437679924078,
  71624393183013992277809728232878811141365503142273439841212989986229620297 80)

Validating Signature...

WARNING! The Certificate Signature is NOT Valid!!
Press anything to continue..
```

Figure 12 Verifying an Altered Certificate

By changing the certificate content, or by recalculating the certificate signature using another key, the certificate will no longer be able to be verified using Alice's public key. In order to successfully alter a certificate content, one must either have the original owner private key (safely kept by Alice) or deceive a CA into registering a new key pair as Alice's key pair; both of which are very difficult to do. Therefore, since the certificate signature can't be verified, Dave now know that the certificate he received is not one that Alice had made, and that the information inside it are not to be trusted.

VI. Conclusions

Using digital signature and the public key infrastructure, maintaining a scientific article's integrity and authors that has been digitally-published in the internet. This paper had proposed a method of using ECDSA and Digital Certificates to achieve that purposes, and had proven that the proposed method is effective at detecting unauthorized changes to a published documents as well as a method of proving false claims regarding the document.

For further studies, exploring the possibility of using an alternative digital signature and public key infrastructure scheme in this method, as well as using a more standard way of creating the documents certificate may be an interesting subject to be explored further.

References

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2015-2016/Makalah2-2016/Makalah-Kripto-2016-05.pdf accessed on May 1th 2018

[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2014-2015/ECC%20(2015).pdf accessed on May 1th 2018

[3] https://safecurves.cr.yp.to/index.html accessed on May 1th 2018

[4] https://pdfs.semanticscholar.org/c06a/d6512775be1076e4abd43e3f2928729da776.pdf accessed on May 2th 2018

[5] https://www.tutorialspoint.com/cryptography/public_key_infrastructure.htm accessed on May 2th 2018

Originality Statement

I hereby declare that this paper is of my own writing, not an adaptation, nor translation of other papers, and is not a work of plagiarism.

Bandung, Mei 15th 2018

Fairuz Astra Pratama - 13514104