International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:5, No:2, 2011

Performance Analysis of Certificateless Signature for IKE Authentication

Nazrul M. Ahmad, Asrul H. Yaacob, Ridza Fauzi, and Alireza Khorram

***Abstract*—Elliptic curve-based certificateless signature is slowly gaining attention due to its ability to retain the efficiency of identity-based signature to eliminate the need of certificate management while it does not suffer from inherent private key escrow problem. Generally, cryptosystem based on elliptic curve offers equivalent security strength at smaller key sizes compared to conventional cryptosystem such as RSA which results in faster computations and efficient use of computing power, bandwidth, and storage. This paper proposes to implement certificateless signature based on bilinear pairing to structure the framework of IKE authentication. In this paper, we perform a comparative analysis of certificateless signature scheme with a well-known RSA scheme and also present the experimental results in the context of signing and verification execution times. By generalizing our observations, we discuss the different trade-offs involved in implementing IKE authentication by using certificateless signature.**

***Index Terms*—Certificateless signature, IPSec, RSA signature, IKE authentication.**

I. Introduction

Network security has become an essential part in today's network by preventing threats from damaging business operation of an organization. With the explosive growth of e-business applications, extra security measures are in need to protect and secure the exchange of the transactions. Transport mode IPSec can be used to efficiently protect data end to end between clients and servers, peers in a workgroup, and extranet partners. The principal feature of IPSec that enable it to support these applications is that it can encrypt and/or authenticate all traffics at the IP layer. Thus, all distributed applications, including remote login, client/server, e-mail, file transfer, web access, and so on, can be secured.

The establishment of a secure end to end communication generally requires IPSec to use IKE to negotiate and establish Security Association (SA) between two or more nodes. A SA is a relationship that describes how the nodes negotiate the security services such as encryption, hash algorithm, keys, and authentication mechanisms to the IP packets. IKE requires a mechanism to authenticate the nodes which involve in the exchange of SA in the negotiation process. The conventional IKE authentication outlined in [7] are pre-shared key and certificate-based asymmetric cryptosystems such as RSA/DSA digital signature and RSA public key encryption. The use of pre-shared key is only feasible in small scale network due to the difficulties in distributing the shared key to each pair of nodes. On the other hand, the use of certificate-based infrastructure requires a common certificate authority (CA) between two nodes.

Nazrul M. Ahmad and Asrul H. Yaacob are with Faculty of Information Science and Technology (FIST), Multimedia University (MMU), Jalan Ayer Keroh Lama, 75450 Melaka, Malaysia. email: (nazrul.muhaimin, asrulhadi.yaacob)@mmu.edu.my

Because of the inherent limitations of the existing implementation of IKE authentication, we propose a new IKE authentication scheme based on certificateless signature, which is derived from the concept of digital signature. Certificateless signature is implemented based on bilinear pairing on elliptic curve. Generally, cryptosystem based on elliptic curve such as Ellitic Curve Digital Signature Algorithm (ECDSA) offers equivalent security strength at smaller key sizes compared to the conventional cryptosystem such as RSA. This results in faster computations and efficient use of computing power, bandwidth, and storage. It makes ECDSA an attractive choice for many IKE implementations [6]. However, ECDSA authentication still requires the nodes to know and trust each other's public key. It's applied by exchanging certificates, possibly within the IKE Phase 1 negotiation. The proposed scheme offers significantly reduction in infrastructure complexity, eliminates the use of certificate to guarantee the authenticity of public keys, and reduces the cost for establishing and managing the certificates.

The rest of the paper is organized as follows. Section II gives a brief introduction to IKE authentication and also summarizes the RSA digital signature algorithm. In Section III, we describe background concepts on bilinear pairings and certificateless signature scheme. In Section IV, we present a new IKE authentication by using certificateless signature scheme. The general performance comparisons such as the size of public key and the size of actual signature block, and the efficiency analysis on signing and verification algorithm for certificateless signature and RSA are given in Section V. Finally, we conclude the paper with Section VI.

II. IKE Authentication

The conventional IKE authentication mechanisms outlined in [7] are pre-shared key and certificate-based techniques such as RSA public-key encryption and RSA/DSA digital signature. A pre-shared key is nontrivial string up to 128 characters long. The authentication is accomplished by assigning the identical key to each node. The pre-shared key authentication offers simplicity in its implementation. On the other hand, the use of certificate-based techniques provides more security and scalability than pre-shared key. In the traditional Public Key Infrastructure (PKI), certificates are deployed to provide an assurance of the binding between public keys and users' identities that hold the corresponding private keys. The binding process is performed by the CA. The public/private key pairs in PKI can either be generated by the CA for the node or the

node can generate the keys for itself and then verify with the CA for authenticity.

In addition to the conventional IKE authentication mechanisms, [6] proposed Elliptic Curve Digital Signature Algorithm (ECDSA) as the authentication method within the IKE and IKEv2. ECDSA is the elliptic curve analogue of the DSA. ECDSA signatures provide advantages including computational efficiency, small signature sizes and minimal communication bandwidth compared to RSA and DSA signatures. Like RSA public key encryption and RSA/DSA Signatures, ECDSA-based signatures still deploy the PKI to convey the elliptic curve domain parameters and public key in X.509 certificates.

Both public key encryption and digital signature require the use of digital certificates to validate the public/private key mapping. However, PKI faces many obstacles in practises such as expensive implementation, the scalability of the infrastructure and complex certificate management [8], [12]. In addition, it is not widely deployed. In contrast, the pre-shared key is not very secure and must be distributed out of band (e.g. telephone or registered email) before IKE negotiations. Moreover, in large-scale deployments a single key is assigned to an entire domain, this leads to compromise of security. Therefore, we propose a new IKE authentication scheme based on certificateless public key infrastructure. The proposed scheme offers significantly reduction in system complexity, eliminates the use of certificate to guarantee the authenticity of public keys, and reduces the cost for establishing and managing the certificates.

A. RSA Digital Signature

The idea of digital signature can be traced back in [3] where it is based on asymmetric public key cryptosystem. Digital signatures can guarantee message integrity and authenticity in unsecure network. It consists of three main algorithms: key generation, signing and verification. In key generation process, public key and private key are generated randomly. To generate the signature, the issuer first calculates a digest of his message using a hash function. Then he encrypts the digest with his private key to generate the signature. Both message and signature are transmitted to the responder. Upon receipt of the message and signature, the responder first decrypts the issuer's signature into a digest using the issuer's public key. Then the responder calculates the digest from the issuer's message. If the two digests match, it proves that the message is from the issuer and unaltered. The most widely known digital signature scheme is RSA.

RSA is named after the initials of the three creators: Ron Rivest, Adi Shamir and Leonard Adleman[13]. It is a public key cryptosystem which is widely used today. The security of RSA is based on the difficulty of factoring two large integers where these integers are the mathematical relationship between public and private keys. The algorithm can be summarized as follows.

1) Generate two prime numbers p and q such that $p \neq q$.
2) Compute $n = pq$ which has an equivalent length of key size, e.g. 1024 bits.
3) Compute the totient $\phi = (p\text{-}1)(q\text{-}1)$.
4) Choose an integer e such that $1 < e < \phi$ and GCD(e, ϕ) = 1.
5) Compute d such that $1 < d < \phi$ and $ed \equiv 1 \bmod \phi$.
6) The public key is (n,e) and the private key is (n,d) where the values of p,q and ϕ should be kept secret.
7) Signing Process: Given a message $M \in M = \{0, 1\}^*$, compute signature $s \equiv m^d \bmod n$.
8) Verification Process: Compute $v = s^e \bmod n$. The signature is valid if and only if $v = m$.

The strength of the signature depends on the size of the key used. An advance in cryptanalysis and computing power requires public key sizes to grow over time to ensure it is secured for particular period of time. Table 1 shows the equivalent key sizes of various public key cryptosystems.

TABLE I
COMPUTATIONALLY EQUIVALENT KEY SIZES

Security Level (bit)	ECC (bit)	RSA/DH/DSA (bit)
80	160	1024
112	224	2048
128	256	3072
192	384	7680

III. CERTIFICATELESS PUBLIC KEY INFRASTRUCTURE

In the certificateless public key infrastructure (CL-PKI), the authenticity of a public key is ensured by the Key Generator Centre (KGC). Unlike the trusted third party in PKI, the KGC does not have access to the node's secret key. Instead, the KGC generates a partial secret key and transmits it to the end node. Upon receipt of the partial key, the node combines partial secret key, KGC's public parameters, and node's secret value to form node's secret key. In this paper, we focus on the certificateless signature (CLS) scheme as only the authentication service is needed. In this section, we describe the concepts of bilinear pairing and related mathematical problems. We then present the model and algorithms used in CLS. The security and efficiency of the CLS can be referred to the original paper [15].

A. Preliminaries

In CL-PKI, a pairing function will give a mapping from a cyclic group to another cyclic group. Particularly in our case, we have a mapping from a subgroup of additive group of elliptic curve points to a subgroup of multiplicative group of the finite field. Let $E(\mathbb{F}_q)$ be an elliptic curve over finite field $\mathbb{F}_q$, with q a large prime

$$y^2 \equiv x^3 + ax + b \pmod{q} \tag{1}$$

Let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, \times)$ be two cyclic groups of prime order r. The pairing function is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ with the following properties [15]:

Bilinearity: $\forall P, Q, R, S \in \mathbb{G}_1$

$$e(P + Q, R + S) = e(P, R)e(P, S)e(Q, R)e(Q, S) \tag{2}$$

Consequently, $\forall a, b \in \mathbb{Z}_r^*$, we have

$$e(aP, bQ) = e(aP, Q)^b = e(P, bQ)^a = e(P, Q)^{ab} \quad (3)$$

Non-degeneracy: If P is the generator for $\mathbb{G}_1$, then

$$e(P, P) \neq 1 \in \mathbb{F}_{q^l}^* \quad (4)$$

where $F_{q^l}^*$ is a generator for $\mathbb{G}_2$ and l is an embedding degree.
Computability: For any $P, Q \in \mathbb{G}_1$, there is an efficient algorithm to compute $e(P, Q)$.

B. Certificateless Signature

A formal CLS structure is a 7-tuple polynomial time algorithms: **Setup, Partial-Secret-Key-Extract, Set-Secret, Set-Secret-Key, Set-Public-Key, Sign** and **Verify**. The KGC will execute **Setup** and **Partial-Secret-Key-Extract** whereas the node will execute **Set-Secret, Set-Secret-Key, Set-Public-Key**. The algorithms are based on the proposed CLS scheme by [15] which provide a better performance compared to [9],[18]. We recall the details of the algorithms here. The discussion on the security and performance can be found in the original papers [15],[9],[18].

1) Setup: This algorithm calculates the parameters of CLS $params$ and master secret key msk. This algorithm runs as follows:

i. Run a generator to output two cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order r and a bilinear pairing

$$e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2 \quad (5)$$

ii. Choose an arbitrary generator $G \in \mathbb{G}_1$ and generate

$$g = e(G, G) \quad (6)$$

iii. Select a random $s \in \mathbb{Z}_r^*$ and set master public key

$$K_{KGC} = sG \quad (7)$$

iv. Choose hash function

$$\mathrm{H} : \{0,1\}^* \rightarrow \mathbb{Z}_r^* \quad (8)$$

The outputs of this algorithm are:

- the domain parameter which is public

$$params =< e, g, G, K_{KGC}, \mathrm{H} > \quad (9)$$

- master private key which is private to KGC

$$msk = s \in \mathbb{Z}_r^* \quad (10)$$

2) Partial-Secret-Key-Extract: This algorithm is executed by KGC when a node requests its partial secret key. The inputs of this algorithm are $params$,msk and the node identity ID. Prior to the release of the partial secret key, the node must be authenticated by the system. The algorithm returns

$$partial\_key = (\mathrm{H}(ID) + msk)^{-1} G \in \mathbb{G}_1 \quad (11)$$

The key is transmitted over a secure channel to the node.

3) Set-Secret: This algorithm computes the secret value used in CLS. The node chooses a random $s_{ID} \in \mathbb{Z}_r^*$. This secret value will be used in the next algorithm to create the node's secret key.

4) Set-Secret-Key: This algorithm creates node's secret key. This key is kept safely by the node and it will be used in the signing process.

$$nsk = (partial\_key, s_{ID}) \in \mathbb{G}_1 \times \mathbb{Z}_r^* \quad (12)$$

5) Set-Public-Key: This algorithm takes as inputs the $params$ and s_{ID} to compute the node's public key. This public key will be used to verify the signature. The public key corresponding to the node is

$$npk = g^{s_{ID}} \in \mathbb{G}_2 \quad (13)$$

6) Signing Process: Given a message $M \in \mathcal{M} = \{0,1\}^*$, this algorithm uses the $params$, and ID as well as its npk and s_{ID} to compute the signature, σ. The steps are as follows:

i. Choose randomly $x \in \mathbb{Z}_r^*$
ii. Compute

$$t = g^x \in \mathbb{G}_2 \quad (14)$$

iii. Calculate the hash

$$h = H(M||ID||npk||t) \in \mathbb{Z}_r^* \quad (15)$$

iv. Compute S

$$S = (x + h \cdot s_{ID}) \cdot partial\_key \in \mathbb{G}_1 \quad (16)$$

v. The signature on M is

$$\sigma = (S, h) \in \mathbb{G}_1 \times \mathbb{Z}_r^* \quad (17)$$

7) Verification Process: To verify the authenticity of a message M with the signature $\sigma = (S, h)$, the other node computes

$$t' = e(S, H(ID) G + K_{KGC}) (npk)^{-h} \quad (18)$$

$$h' = H(M||ID||npk||t') \quad (19)$$

The message is authenticated if and only if $h' = h$.

IV. IKE Authentication using Certificateless Signature

In this section, we discuss a new IKE authentication scheme with certificateless signature based on bilinear pairing over elliptic curve. This scheme is called CL-PKI Type A. We employ pairing-based elliptic curve cryptosystem due to the small key size and low computational overhead. So far, there have been a total of eleven IANA-assigned attribute numbers for Phase 1 IKE authentication. Here, we suggest a twelveth option: CL-PKI Type A, specified by attribute value 12 in the SA. In conjunction with the scheme, we specify a set of elliptic curve group parameters $gparams$. These consist of the recommended choice of the curve itself and the field it is defined over.

TABLE II
PARAMETER *gparams* FOR ELLIPTIC CURVE

gparams	value (in Hexadecimal)
q	88EC1409 B09802F9 C8CA36B4 8E2BFB6F 6D86E09F 5B523FC7 26A3840D B9836B52 B05B3013 7923706B 87BB8012 037B4C76 3C2A3762 CBBFDCDB 074712B6 BAF4360B
r	FFFFFFFF FFFFFFFF FFBFFFFF FFFFFFFF FFFFFFFF

where*:*

- *q* prime order which satisfies 3 mod 4
- *r* subgroup order of Solinas prime

A. Pairing-friendly Elliptic Curve

Pairing-based cryptography is a cryptography technique in which bilinear map is constructed between two groups where the discrete logarithm problem is hard. The pairings are constructed based on Weil or Tate pairings on elliptic curve over finite fields. There is large number of constructions of pairing-friendly elliptic curves with prescribed embedding degree[5]. However, [11] showed that a supersingular elliptic curve must have embedding degree $l \leq 6$ with $\#E(\mathbb{F}_q) = q+1$ and $\#E(\mathbb{F}_{q^2}) = (q+1)^2$. In this work, a supersingular elliptic curve with an embedding degree 2 will be used to implement CL-PKI Type A scheme. The underlying of this proof of concept is based on C implementation of pairing-based cryptography available at [10]. Type A curve is used by applying modified Tate pairing on the curve $y^2 = x^3 + x$ over the field $\mathbb{F}_q$. The group parameters $gparams$ for the elliptic curve are defined in Table II.

B. KGC Setup

In CL-PKI, a KGC is involved in generation of public parameters and deriving a partial secret key from the node's identity whose is assumed to be unique in a trust domain. Initially, KGC executes **setup** algorithm to generate a list of public parameters $params$ and master secret key msk. This algorithm is executed only when a new set of $params$ needs to be generated or in case of KGC is under compromised attack (in practice very seldom). This algorithm takes as input the system security parameter *k*. Strength of public domain parameters depends on security *k* which corresponds to RSA modulus bit-length of comparable of security. In this implementation, we use *k* = 1024 bits which attains 80 bits of cryptographic strength [2]. This level of security is the minimum requirement to ensure any adversary to perform computationally-intensive cryptanalysis attack. The sample of parameters used for this implementation is listed in Table III. Apart from the parameters generated in section III-B, two additional parameters are listed. KGC_{ADDR} and KGC_{PORT} are IP address or hostname of the KGC and port on which the KGC is listening. These two parameters are used by the node to request partial secret key from the KGC.

C. Distribution of Shared, Public Paramaters

In certificateless public key infrastructure, all end nodes are required to know the domain public parameters *params* in order to generate their public/secret keys pair. The most

TABLE III
PARAMETER $params$ FOR TRUST DOMAIN

params	value
e	"modified Tate pairing"
g	75338D8E 3B51425D B8E1352E 8AF0563F EFE1A07C 9AAC8614 CED241D7 EE120776 DFABB10E 3A00EEA4 0FAD9165 DDCE5130 470AF6D9 83D8645F 0F5862C1 A500FC36, 29BF49F5 9CBBC692 47E41FDB 8B790189 CD0F8470 610AF6ED CD31BD9D 4121F9FF 08EC9103 1B65D9B9 990C3753 092600C4 81419A65 EBF3F2E0 76366F24 AC666A80
G	746C02FB 2B09F5B4 BB4C33BC 7DF83398 E7A4AA81 4DFC2E64 955C7DFE EB52BD8B 343574B7 261BB94B F4C510DF 9199F14F 2C783CD6 9D86D46D CBADE603 F162AE44, 3354F695 533E0A30 009231BF 87B12361 31740936 DC88F5FF 332BA46D 2D187594 99936B23 6D495138 54F062AA 8400ACDF 794809AA 6573971D F458271A 581450FA
K_{KGC}	5A595019 36F15AD4 82CB1209 30EFE324 F72DEA27 38B2C085 3148A1A0 D962A4B9 CED8531C 8047F337 EA66C34F 7BA9E439 44055EA9 4293F23C 31741EC0 0C18380E, 03DEBDEA 814D2D36 1EEC1CEF 88819F89 B790DB77 B2B25B65 83EC7763 DCF5D85D AC292BB7 13E8579C CFEF522E 57EBB5C1 12C53BC0 4D766A49 312F4742 1C94DF46
H	1.3.14.3.2.26 (SHA-1 OID)
KGC_{ADDR}	kgc.testbed.com
KGC_{PORT}	80

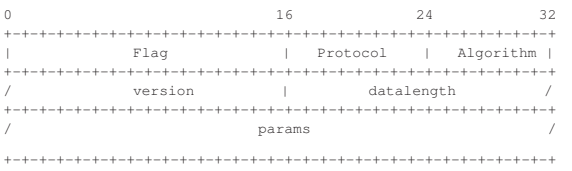

Fig. 1. KGCPARAMS DNS Resource Record

feasible solution to distribute the KGC's *params* is to use the existing network infrastructure. Smetters and Durfee proposed to piggyback this information to DNS reply [14]. Due to the simplicity of the mechanism, we enhance this distribution technique for our implementation. Initially, KGC executes **setup** algorithm to generate *params*. KGC is then responsible to publish the *params* to DNS server. To provide secure and authenticating update to DNS server, KGC deploys Transaction Signature (TSIG) [17], a cryptographically means of identifying each endpoint of a connection that involves in making or responding to a DNS update. Under compromise attack, KGC is forced to regenerate and republish *params* to DNS server. Presumably, KGC is on a closely-monitored and well-protected node, frequent dynamic update can be avoided for a reasonably long period of time.

To realize the distribution of KGC's *params*, we introduce a new Resource Record (RR) type to DNS. In this section, we discuss the format of RR fields associated to the new RR type, i.e. KGCPARAMS, as shown in Figure 1. RDATA portion of KGCPARAMS RR utilizes the same format as that of KEY RR [4]. RDATA comprises of flags, protocol, algorithms and two additional fields, version and params. Version field is an integer that defines the version of the *params* generated in a

single trust domain. Final field of RDATA represents a base64 encoding of the *params*.

To initiate the IPSec communication, the client initially sends KGCPARAMS query to DNS server by setting the QNAME of the query to Fully Qualified Domain Name (FQDN) of the responder. Once obtained the DNS response, the client extracts KGC_{ADDR} and KGC_{PORT} from *params* to contact KGC for requesting *partial_key*. Upon receiving the query from the client, KGC runs **partial_secret_key_extract** algorithm by taking the client's identity, *params*, and *msk* as inputs to generate IPv6 client's *partial_key*. This *partial_key* is transmitted to the client over a secure channel. Upon receipt of the *params* and *partial_key*, the client generates public/secret keys pair (*nsk*, *npk*). For the responder, the DNS query needs to be preformed once it received the first IKE message from the initiator. In this case, the responder is pre-configured with the FQDN of the KGC. As an example, the node could be configured with the name "kgc.testbed.com", where the testbed.com is the domain name. The responder generates a DNS query by setting the QNAME to the name of the KGC. The query has QTYPE set to KGCPARAMS so that the DNS server replies the *params* to the responder. Upon receipt of the $params$ from DNS server and *partial_key* from KGC, the nodes generate secret/public keys pair (*nsk*, *npk*) as shown in Table IV.

TABLE IV
CLIENT AND SERVER KEY PAIRS

(nsk, npk)	value
nsk_{client}	6720C35D 589137D8 65402209 40729D26 DF0FED11 40C87086 512CBEC0 CE6626BC B795D32A 9B8DF98B 6E684EB2 85088F08 A4C02BB5 7B791970 A4D9459F 5E7B00F8, 7F6FE7B5 7A520107 CB84F61B 62AFF33E 07443F1B E8A2FDA8 8A828021 AFAA9C87 3A6978E3 CA45A317 8101B1AF F044166B 497E2CD4 0DA69E99 AD302F4F C8989344, 19CAFF80 7E863C1A 0677588D E9200430 617E1AAE
npk_{client}	207B9091 7918DC9B 482F4BC4 192244C7 0DB1FE04 429670AA 44100626 B8962E85 3E47A90D 1B02BFD9 A0F225F5 1EA3AE50 BE1B2F4A 6DC49CAC 5BBA636C FC9DD35B, 612E1920 E55AD010 23B2FAAD 3AECD562 45383E11 6327E9BC C45221E6 3207E5AA E2A71CF1 0421BBE3 E722E1DE 225C296F 7412C4BC 1B6B87AD B2406AD4 7FA12120
nsk_{server}	378F3C29 93CD0500 CBB1ED77 880834F9 BEF07C5F F5217FAD BCA2BB27 17FC7ED0 29B0F942 27EDF517 4A500413 D549F0FD 000DFD7D C5EABAD7 B111B636 8BE1DDB1, 172A7952 261A56C8 3C7EA4FE A639E8DE FD558755 3FF465AD 77F05D12 6F847A43 0ED6CB9A FC340DBA 1F3F2515 915C8099 72315253 8518B6F9 A9F4C3CE 257BB7E8, B967AFB6 065EE2CE F5A6A8A0 160130DD 33B3D6D4
npk_{server}	7D70337C DCD122CA FD30A82B 6B071D9F 41431D3D 42380B7B F0A601F3 F6D33972 B023FD51 8DBC20EC 5BB6F989 1B850D25 2CB6389A 47358A18 5E6441D2 6E484DDD, 0CBE45EB 75ECAC53 A3D7164B 8DE864AF 05C1D046 F661869F 6EE7AABE 6C3BCFB1 8410721D BCB9E228 D22DD17B 400EF4CC BDE7DC23 BD0FD613 19F8E7C0 B326D930

D. New IKE Authentication

Because of the inherent limitations of the current implementation of IKE authentication by using pre-shared key and PKI certificate-based, we propose new IKE authentication scheme which has no impact on the existing network infrastructure, but it requires both end nodes to have the appropriate IPSec software. The framework CL-PKI Type A IKE authentication is inherited from IKE authentication with digital signature.

We propose to implement CL-PKI Type A in the Phase 1 of the IKE negotiation. This scheme is used to authenticate both end nodes and to provide the protection for the upcoming Phase 2 IPSec negotiation. To initiate IKE negotiation, the exchange of first two messages is identical to the existing scheme. The new scheme only has impact on the following subsequent exchanges. The nonce payloads N_{client} and N_{server} that are sent in third and fourth message normally are a large random number between 64 bit and 2048 bits. Instead of random number, this scheme transmits npk_{client} or npk_{server} to remote node. This public key is used to validate and verify the signature that will be sent in the later stage of IKE negotiation.

The encryption key $SKEYID_e$ is derived from messages 3 and 4 as described in [7]. This key is used to encrypt and to provide confidentially for IKE messages 5 and 6. These last two messages are strictly for exchanging the node identity, proving the identity and retroactively authenticating all the previous messages. The ID payload contains information about the identity of the client or server. This scheme proposes to populate client and server IP addresses in ID payload. The server then can use the *ID* as a lookup key for IKE SA policy and the client's public key npk_{client}, and as part of the security attributes that are related to the generation and verification of authenticators generated by the initiator and responder, $Hash_{client}$ and $Hash_{server}$, respectively.

Until the end of IKE message 4, all exchanges are not authenticated. To provide mutual authentication, both nodes need to provide a digital signature in the exchange that the other node will verify. In this case, the client and server authenticate their exchanges by sending signed $Hash_{client}$ and $Hash_{server}$ respectively, as defined in the Figure 2 at the last two exchange messages.

The client and server execute the signing algorithm **sign** that takes $Hash_{client}$ or $Hash_{server}$, client or server ID, $params$, and end node's secret key as input. Upon reception of IKE third exchange, the client and server decrypt the messages to obtain the ID payload and signed $Hash_{client}$ or $Hash_{server}$. Finally, the server's verification algorithm, **verify**, inputs $Hash_{client}$ generated by the client by using all the security attributes transmitted prior to this verification, client ID_{client}, *params*, client's public key npk_{client}, and signature *Sig-* $(Hash_{client})$. The algorithm returns output 1 if the signature is valid and 0 otherwise. Figure 2 summarizes the new IKE authentication by using CL-PKI Type A.

The signature is a combination of a point in group $\mathbb{G}_1$ and a number in $\mathbb{Z}_r^*$, noted as S and h respectively. As outlined in equations (18) and (19), the verification process involves in computing new values of t' and h' which are based on the

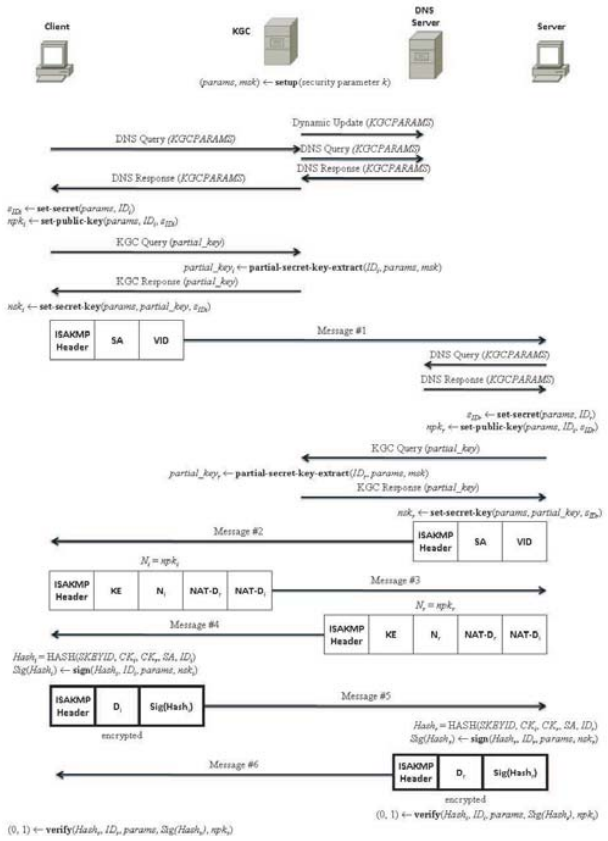

Fig. 2. IKE messages exchange

TABLE V
EXAMPLE OF SIGNATURE

variable	value (in hexadecimal)
$Hash_{Server}$	ECA0C540 86394CEE F8B0DB85 46AA9112 B985E777
h	32424446 31383439 31373231 44424345 33373633
S	83E84284 44F2EB3A 0C3FB086 9B0CF15C 0CA7C097 A8999D46 F29E4223 F233F1F4 178065E5 8574F01A D39FD968 C72996A9 4360CE34 F6F7734E 7E9FDF89 4A333793, 4D1DD10A F173DE8A 6BAD838E A4683580 0548D4F6 54685978 9B632722 FD08B1A4 2A3CFC25 6D689731 239D460E 7FD71DA2 81C3BED2 F751EFE3 F07AFA39 B6B61B15
$Sign(Hash_{Server})$	(S,h)

transmitted signature, and in comparing the values of h and h'. The sample of values related to signing and verification processes of the server are shown in Tables V and VI. The $Hash_{server}$ represents the resulting value from hashing function of a string derived from secret material known only to the active nodes *SKEYID*, both cookies of active nodes, SA, and ID_{server}. The signing and verification processes for the server took place before and after the transmission of IKE message 6. From Tables V and VI, we notice $h = h'$, which indicate the the verification of the authenticity of the server is successful.

TABLE VI
EXAMPLE OF VERIFICATION

variable	value (in hexadecimal)
t'	556F8401 D16E0939 75C87CE4 8B072861 5F9EFC7F EBBCAEB9 73AA82D8 3672EB05 3B9B6E46 2692B42F AE7AB8EC A1665276 FDD0EE53 5014006C 2C3A5941 3C20B597, 24CCEB5B 11C1AB11 7BC62CF7 57E1AEE8 FD41E73B A3EDB09C 6CF55C0C 106D07AA DADE9C98 98B30157 39A08D81 11BD880B B63FEFBA 1197B2EB 289286FD 3B56347C
h'	32424446 31383439 31373231 44424345 33373633

TABLE VII
PUBLIC KEY SIZE OF RSA AND CL-PKI

Algorithm	RSA 1024-bit	CL-PKI-160-bit
Public Key Size	131 bytes	128 bytes

V. PERFORMANCE ANALYSIS

We evaluate the performance of CL-PKI and make comparison with a well-known RSA cryptosystem in terms of the size of public key, the size of signature block, and the speed for generating or verifying the signature.

A. Public Key Size of RSA and CL-PKI

An RSA public key consists of the modulus n and the public exponent e. In a 1024-bit RSA system, the length of n is 1024 bits and a common value for the public exponent $e = 2^{16} + 1 = 65537$. By using a small value of e, the public key operation can be executed very fast. Thus, 1024-bit RSA public key requires 128 bytes for the modulus and 2 + 1 = 3 bytes for the public exponent field. The total size of the RSA public key is then 131 bytes.

The public key of CL-PKI is represented as an element in the finite field F_{q^l} and it is implemented as $F_q[X]/f(X)$, where l is the embedding degree and $f(X)$ is irreducible polynomial. To achieve 80-bit security level, the size of the extension field F_{q^l} should be at least 1024 bits long for finite field F_q size of 512 bits and the group order $E(F_q)[r]$ of 160 bits long. This is due to the fact that element in F_{q^l} is represented by using polynomial basis $(1, X, X^2, ..., X^{l-1})$ of degree at most l - 1, where X is a root of the irreducible polynomial over F_{q^l}. In this work, we implemented CL-PKI by using supersingular elliptic curve with embedding $l = 2$. For an element $A(X)$ in F_{q^2}, $A(X)$ is represented as $a_1X + a_0$ $(a_1, a_0 \in F_q)$. In our implementation, $A(X)$ is encoded as (a_1, a_0). Therefore, the CL-PKI public key consists of two 512-bit elements, giving a total key size of 1024 bits or 128 bytes.

B. Signature Block Size

RSA signature length is a function of modulus n in bytes. Hence, 1024-bit RSA signature can be represented in 128 bytes. CL-PKI signature block consists of (S, h), where S is a point on the elliptic curve $(S \in G1)$ and h is an 160-bit output of the pseudo-random function based on SHA-1. For 512-bit elliptic curve, the S is represented by two 64-byte numbers.

TABLE VIII
SIGNATURE BLOCK SIZE

Algorithm	RSA 1024-bit	CL-PKI-160-bit
Signature Block Size	128 bytes	85 bytes

TABLE IX
SIGNING AND VERIFICATION EXECUTION TIME (IN MILLISECONDS)

Sign		Verify	
RSA	CL-PKI	RSA	CL-PKI
0.662	2.975	0.041	4.112

Therefore, the length of the signature block for CL-PKI is 128 + 20 = 148 bytes. However, there is a standard compression technique that can be used to reduce the elliptic curve point size by a factor of 2. Elliptic curve point compression allows the y-coordinate of the point to be represented compactly using a single additional bit [1]. If we deploy the point compression in this scheme, S could be represented by using one 512-bit value and one additional bit. This would then require 64 + 1 = 65 bytes, giving a total signature size of 85 bytes. The bandwidth or storage used is considerably less than RSA if only one point is sent.

C. Signing and Verification Execution Time

In this section, we present the analysis on the signing and verification execution times for CL-PKI and RSA cryptosystem. The experiments were conducted on a 2.93 GHz Core i5 Linux PC. We have used a popular OpenSSL 0.9.85o crypto library [16] to implement the RSA cryptographic functions. For RSA signature generation, the scheme uses 1024-bit public modulus n and $e = 2^{16} + 1(65537)$ for public exponent. For CL-PKI, we implement the scheme in C using pairing-based cryptography (PBC) library [10] (version 0.5.8) for the elliptic-curve and pairing operations. Table IX shows the experimental results for an average execution time of signing or verification operation, where the amount time specified is for a single operation. We run each specific operation 100 times.

Based on the results shown in Table IX, RSA signing operation is generally slower than the verification operation. RSA operation is essentially a modular exponentiation. Technically, one modular exponentiation involves in each signing and verification operation. By using small public exponent value (e = 65537) for the public key, the verification operation can be performed faster than signing operation.

According to the observations, 1024-bit RSA signing operation is about 5 times faster than the one of CL-PKI. However, RSA verification operation totally outperforms CL-PKI. This is due to the fact that pairing is considered to be the most computationally expensive operation in certificateless signature. Therefore, it is important to efficiently implement the pairing computation. In CL-PKI, the pairing $e(G_1, G_1) = g$ can be pre-computed and published as the public parameter *params*. Thus, it eliminates the pairing computation in the signing operation, and it only needs to perform one computationally expensive pairing in the verification operation. There are two expensive operations underlying the CL-PKI scheme, namely, modular exponentiation and pairing. Unlike CL-PKI signing operation where there is only one modular exponentiation, the verification operation involves one modular exponentiation and one pairing computation. Therefore, CL-PKI verification operation is slower than signing operation.

VI. CONCLUSION

This paper proposed a new IKE authentication based on certificateless signature in order to eliminate the need of certificate in IPSec. Cryptosystem based on elliptic curve offers good security with smaller key size compared to RSA. Smaller key size results in faster computation and efficient use of computing power, bandwidth, and storage. However, elliptic curve-based certificateless signature does not inherit this advantage. In this paper, we implemented certificateless signature by using a supersingular elliptic curve with an embedding degree 2. If the size of an element of F_q is 512 bits and embedding degree l is 2, then the size of element of F_{q^l} and the size of $g = e(G_1, G_1)$ are 1024 bits. Since the public key of the proposed scheme is one of the elements in F_{q^l}, the key size is approximately the same length with 1024-bit RSA. Moreover, due to computationally expensive operation of bilinear pairing, the scheme introduces extra processing overhead in verification algorithm. Nevertheless, the proposed scheme offers significantly reduction in infrastructure complexity, eliminates the use of certificate to guarantee the authenticity of public keys, and reduces the cost for establishing and managing the certificates.

Acknowledgment.: The research is funded by Ministry of Science, Technology, and Innovation (MOSTI), Government of Malaysia, under e-ScienceFund grant No. 01-02-01-SF0170.

REFERENCES

[1] Certicom Research: Standards for efficient cryptography - SEC1: Elliptic curve cryptography (2000)
[2] Certicom Research: Standards for efficient cryptography - SEC2: Recommended elliptic curve domain parameters (2000)
[3] Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **IT-22**(6), 644–654 (1976)
[4] Eastlake 3rd, D.: Domain name system security extensions (1999)
[5] Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal Of Cryptology **23**(2), 224 – 280 (2010)
[6] Fu, D., Solinas, J.: IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA). RFC 4754 (Proposed Standard) (2007)
[7] Harkins, D., Carrel, D.: The Internet Key Exchange (IKE). RFC 2409 (Proposed Standard) (1998). Obsoleted by RFC 4306, updated by RFC 4109
[8] Jancic, A., M.J.Warren: PKI - advantages and obstacles. In: 2^{nd} Australian Information Security Management Conference (2004)
[9] Lifeng, G., Lei, H., Yong, L.: A practical certificateless signature scheme. Internation Symposium on Data, Privacy, and E-Commerce pp. 248–253 (2007)
[10] Lynn, B.: The pairing-based cryptography (PBC) library. http://crypto.stanford.edu/pbc/
[11] Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Transactions on Information Theory **39**(5), 1639 – 1646 (1993)
[12] Peyravian, M., Roginsky, A., Zunic, N.: Non-PKI methods for public key distribution. Computers & Security **23**(2), 97 – 103 (2004)
[13] Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**, 120–126 (1978)

[14] Smetters, D.K., Durfee, G.: Domain-Based Authentication of Identity-Based Cryptosystems for Secure Email and IPsec. In: 12th Usenix Security Symposium. Washington, D.C. (2003)
[15] Terada, R., Denise, H.G.: A certificateless signature scheme based in bilinear pairing functions. In: symposium on Cryptography and Information Security (2007)
[16] The OpenSSL Project: Openssl. http://www.openssl.org
[17] Vixie, P., Gudmundsson, O., 3rd, D., Wellington, B.: Secret key transaction authentication for DNS (TSIG) (2000)
[18] Wang, C., Huang, H., Tang, Y.: An efficient certificateless signature from pairings. Internation Symposium on Data, Privacy, and E-Commerce pp. 236–238 (2007)