An Efficient Certificateless Designated Verifier Signature Scheme

Debiao He and Jianhua Chen
School of Mathematics and Statistics, Wuhan University, China

Abstract: *To solve the key escrow problem in identity-based cryptosystem, Al-Riyami et al. introduced the CertificateLess Public Key Cryptography (CL-PKC). As an important cryptographic primitive, CertificateLess Designated Verifier Signature (CLDVS) scheme was studied widely. Following Al-Riyami et al. work, many certificateless Designated Verifier Signature (DVS) schemes using bilinear pairings have been proposed. But the relative computation cost of the pairing is approximately twenty times higher than that of the scalar multiplication over elliptic curve group. In order to improve the performance we propose a certificateless DVS scheme without bilinear pairings. With the running time of the signature being saved greatly, our scheme is more practical than the previous related schemes for practical application.*

Keywords: *CL-PKC, DVS, bilinear pairings, elliptic curve, random oracle model.*

Received October 31, 2010; accepted May 24, 2011; published online August 5, 2012

1. Introduction

To solve the problem of certificate management in traditional public key cryptography, Shamir [15] proposed the concept of the identity-based public key cryptography. However, identity-based public key cryptography needs a trusted Key Generation Centre (KGC) to generate a private key for an entity according to his identity. So, the key escrow problem arises. To solve the problem, CertificateLess Public Key Cryptography (CL-PKC) was proposed by Al-Riyami and Paterson [1].

Digital signature scheme, one of the general primitives of cryptography, has many applications in information security to provide authentication, data integrity, and non-repudiation. In an ordinary digital signature scheme, anyone can verify the validity of a signature using the signer's public key. However, in some scenarios, this public verification is not desired, if the signer does not want the recipient of a digital signature to show this signature to a third party at will. To address this problem above, Jakobsson *et al*. [9] proposed the concept of Designated Verifier Signature (DVS) schemes. A DVS scheme is special type of digital signature scheme which provides message authentication without non-repudiation. These signatures have several applications such as in E-voting, call for tenders and software licensing. Suppose Alice has sent a DVS to Bob. Unlike the conventional digital signatures, Bob cannot prove to a third party that Alice has created the signature. This is accomplished by the Bob's capability of creating another signature designated to himself which is indistinguishable from Alice's signature.

Following the pioneering work due to Jakobsson *et al*. [9], many PKC-based DVS [12, 14, 17] and ID-based DVS [10, 11, 16, 21] have been proposed. Huang *et al.* [8], presented the first CertificateLess Designated Verifier Signature (CLDVS) scheme. Unfortunately, their scheme is insecure against a malicious but passive KGC attack.

In order to improve the security, several CLDVS schemes [3, 13, 19, 20] have been proposed. All the above CLS schemes may be practical, but they are from bilinear pairings and the pairing is regarded as the most expensive cryptography primitive. The relative computation cost of a pairing is approximately twenty times higher than that of the scalar multiplication over elliptic curve group [4, 18]. Therefore, CLDVS schemes without bilinear pairings would be more appealing in terms of efficiency.

In this paper, we present a CLDVS scheme without pairings. The scheme rests on the Elliptic Curve Computational Diffie-Hellman Problem (ECDHP). With the pairing-free realization, the scheme's overhead is lower than that of previous schemes [3, 8, 13, 19, 20] in computation. The rest of the paper is organized as follows: in section 2, we recall the model for CLDVS and the security properties of CLDVS, we propose our scheme in section 4, security analysis and performance analysis of the proposed scheme are given in section 5. Finally, we conclude this paper.

2. Preliminaries

2.1. Background of Elliptic Curve Group

Let the symbol E/F_p denote an elliptic curve E over a prime finite field F_p, defined by an equation:

$$y^2 = x^3 + ax + b, \quad a, b \in F_p \tag{1}$$

and with the discriminant:

$$\Delta = 4a^3 + 27b^2 \neq 0 \quad (2)$$

The points on E/F_p together with an extra point O called the point at infinity form a group:

$$G = \{(x, y) : x, y \in F_p, E(x, y) = 0\} \bigcup \{O\} \quad (3)$$

Let the order of G be n. G is a cyclic additive group under the point addition "+" defined as follows: Let $P,Q \in G$, l be the line containing P and Q (tangent line to E/F_p if $P=Q$), and R, the third point of intersection of l with E/F_p. Let l' be the line connecting R and O. Then P "+" Q is the point such that l' intersects E/F_p at R and O and P "+" Q. Scalar multiplication over E/F_p can be computed as follows:

$$tP = P + P + \ldots + P(t \text{ times}) \quad (4)$$

The following problems defined over G are assumed to be intractable within polynomial time. Computational Eliptic curve Diffie-Hellman problem: For P the generator of G, given $Q_1=a.P$ and $Q_2=b.P$ to compute $Q=ab.P$.

2.2. Certificateless Designated Verifier Signatures

A CLDVS scheme consists of eight algorithms [8]: Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Sign, Verify and Transcript-Simulation.

- *Setup:* Taking security parameter k as input and returns the system parameters params and master key.
- *Partial-Private-Key-Extract:* It takes params, master key and a user's identity ID as inputs. It returns a partial private key d_{ID}.
- *Set-Secret-Value:* Taking as inputs params and a user's identity ID, this algorithm generates a secret value s_{ID}.
- *Set-Private-Key:* This algorithm makes params, a user's partial private key d_{ID} and his secret value s_{ID} as inputs, and outputs the full private key $sk_{ID}=\{d_{ID},s_{ID}\}$.
- *Set-Public-Key:* Taking as inputs *params* and a user's secret value s_{ID} and d_{ID}, and generates a public key pk_{ID} for the user.
- *Sign:* It takes as inputs params, a message m, the signer A's identity ID_A, and A's private key sk_{ID}, the designated verifier's B's identity ID_B public key pk_{ID} and outputs a signature S.
- *Verify:* It takes as inputs params, a public key pk_{ID}, a message m, the signer A's public key pk_{ID}, the designated verifier's B's identity ID_B, B's private key sk_{ID} and a signature S, and returns 1 means that the signature is accepted. Otherwise, 0 means rejected.
- *Transcript-Simulation:* An algorithm that is run by the designated verifier B to produce identically distributed transcripts that are indistinguishable from the original signer A.

2.3. Security Properties of Certificateless Designated Verifier Signatures

The CLDVS scheme must satisfy the following properties:

- *Correctness:* If the signer properly produces a CLDVS by the CLDVS-Sign algorithm, then the verifying algorithm must accept the produced signature.
- *Unforgeability:* It is computationally infeasible to construct a valid CLDVS without the knowledge of the private key of either the signer or the designated verifier [8].
- *Source hiding:* Given a message m and a CLDVS on m, it is infeasible to determine who created this signature either the original signer or the designated verifier, even if one knows all the private keys [8].
- *Non-delegatability:* Given any indirect form of the private key of the signer, it is infeasible to construct a valid CLDVS to any designated verifier [8].

2.4. Security Model for Certificateless Designated Verifier Signatures

In CLDVS, as defined in [8], there are two types of adversaries with different capabilities, we assume Type 1 Adversary, $\mathcal{A}1$ acts as a dishonest user while Type 2 Adversary, $\mathcal{A}2$ acts as a malicious KGC:

- *CLDVS Type 1 Adversary:* Such an adversary $\mathcal{A}1$ represents a third party attacks against the CLDVS scheme. $\mathcal{A}1$ does not have access to the master key nor the user partial private key, but $\mathcal{A}1$ can compromise users' secret values or replace users' public keys at will, because of the uncertified nature of the public keys generated by the users.
- *CLDVS Type 2 Adversary:* Such an Adversary $\mathcal{A}2$ represents a malicious-but-passive KGC who is assumed malicious at the very beginning of the Setup stage of the system. $\mathcal{A}2$ can access to the master key, but cannot obtain the user secret value nor replace the user public key.
- *Definition 1.* A CLDVS scheme is existential unforgeable against adaptively chosen message and identity attacks if and only if it is secure against both types of adversaries.

3. Our Scheme

3.1. Scheme Description

In this section, we present an ID-based signature scheme without pairing. Our scheme consists of eight

algorithms: Setup, Partial-Private-Key-Extract, Set-Secret-Value, Set-Private-Key, Set-Public-Key, Sign, Verify and Transcript-Simulation.

- *Setup:* Takes a security parameter *k*, returns system parameters and a master key. Given *k*, KGC does as follows:
  1. Choose a *k*-bit prime *p* and determine the tuple $\{F_P, E/F_P, G, P\}$ as defined in section 2.1.
  2. Choose the master private key $x \in Z_n^*$ and compute the master public key $P_{pub}=x \cdot P$.
  3. Choose two cryptographic secure hash functions $H_1 : \{0,1\}^* \to Z_n^*$ and $H_2 : \{0,1\}^* \to Z_n^*$.
  4. Publish $params=\{F_P, E/F_P, G, P, P_{pub}, H_1, H_2\}$ as system parameters and keep the master key *x* secretly.
- *Set-Secret-Value:* The user with identity ID picks randomly $s_{ID} \in Z_n^*$, computes $P_{ID}=s_{ID} \cdot P$ and sets s_{ID} as his secret value.
- *Partial-Private-Key-Extract:* This algorithm takes system parameters, master key, a user's identifier and $P_{ID}=s_{ID} \cdot P$ as input and returns the user's ID-based private key. With this algorithm, KGC works as follows for each user with identifier ID_U:
  1. Choose at random $r_{ID} \in Z_n^*$, compute $R_{ID}=r_{ID} \cdot P$ and $h_{ID}=H_1(ID, R_{ID}, P_{ID})$.
  2. Compute $d_{ID}=r_{ID}+h_{ID} \; x \bmod n$.

  The user's s partial private key is the tuple $\{d_{ID}, R_{ID}\}$ and he can validate her private key by checking whether the equation $d_{ID} \cdot P=R_{ID}+h_{ID} \cdot P_{pub}$ holds. The private key is valid if the equation holds and vice versa.
- *Set-Private-Key:* Given *params*, the user's partial private key d_{ID} and his secret value s_{ID}, and output a pair $sk_{ID}=\{d_{ID}, s_{ID}\}$ as the user's private key.
- *Set-Public-Key:* This algorithm takes *params*, the user's secret value s_{ID} as inputs and as inputs, computes $P_{ID}=s_{ID} \cdot P$ and generates the user's public key $pk_{ID}=\{R_{ID}, P_{ID}\}$
- *Sign:* This algorithm takes system parameters, the signer *A*'s identity ID_A, and *A*'s private key sk_{ID_A}, the designated verifier's *B*'s identity ID_B public key pk_{ID_B} and a message *m* as input and returns a signature of the message *m*. The user does as the follows:
  1. Compute $h_{ID_B} = H_1(ID_B, R_{ID_B}, P_{ID_B})$.
  2. Choose at random $l \in Z_n^*$ to compute $c = l \cdot (P_{ID_B} + R_{ID_B} + h_{ID_B} \cdot P_{pub})$.
  3. Compute $r=H_2(m,c)$.
  4. Choose at random $t \in Z_n^*$ and compute $s = lt^{-1} - r(d_{ID_A} + s_{ID_A}) \bmod n$.
  5. The resulting signature is (*r, s, t*).
- *Verify:* This algorithm takes system parameters, the signer *A*'s identity ID_A, and *A*'s public key pk_{ID_A}, the designated verifier's *B*'s identity ID_B private key sk_{ID_B}, a message *m* and a signature (*r, s, t*) as inputs. Then return 1 means that the signature is accepted. Otherwise, 0 means rejected. The user does as the follows:
  1. Compute $h_{ID_A} = H_1(ID_A, R_{ID_A}, P_{ID_A})$.
  2. Compute $c' = t(d_{ID_B} + s_{ID_B})(s \cdot P + r \cdot (P_{ID_A} + R_{ID_A} + h_{ID_A} \cdot P_{pub}))$.
  3. Check whether $r=H_2(m,c')$ holds. Return 1 if it is equal. Otherwise return 0.
- *Transcript-Simulation:* This algorithm takes system parameters, the signer *A*'s identity ID_A, and *A*'s public key pk_{ID_A}, the designated verifier's *B*'s identity ID_B private key sk_{ID_B} and a message *m* as inputs. Then generate a signature (*r, s, t*). The user does as the follows:
  1. Compute $h_{ID_A} = H_1(ID_A, R_{ID_A}, P_{ID_A})$.
  2. Choose at random $s', r' \in Z_n^*$, compute $c = s' \cdot P + r' \cdot (P_{ID_A} + R_{ID_A} + h_{ID_A} \cdot P_{pub})$, $r=H_2(m,c)$, $l=r'l^{-1} mod\ n$, $s=s'l^{-1} mod\ n$ and $t=l(d_{ID_B}+s_{ID_B})^{-1} mod\ n$.
  3. The resulting signature is (*r, s, t*). Since

$$\begin{aligned} c' &= t(d_{ID_B} + s_{ID_B})(s \cdot P + r \cdot (P_{ID_A} + R_{ID_A} + h_{ID_A} \cdot P_{pub})) \\ &= l(s \cdot P + r \cdot (P_{ID_A} + R_{ID_A} + h_{ID_A} \cdot P_{pub})) \\ &= sl \cdot P + rl \cdot (P_{ID_A} + R_{ID_A} + h_{ID_A} \cdot P_{pub})) \\ &= s' \cdot P + r' \cdot (P_{ID_A} + R_{ID_A} + h_{ID_A} \cdot P_{pub})) = c \end{aligned} \tag{5}$$

Then the designated verifier can generate the same transcripts in an indistinguishable way.

3.2. Security Analysis

In the section, we give security analysis of our proposed scheme and show that the security of our proposed scheme is secure under the difficulty of solving the ECCDH problem.

1. *Correctness:* The correctness of proposed scheme can be verified by the following equations.
   Since $(d_{ID_A} + s_{ID_A}) \cdot P = P_{ID_A} + R_{ID_A} + h_{ID_A} \cdot P_{pub}$ and $(d_{ID_B} + s_{ID_A}) \cdot P = P_{ID_B} + R_{ID_B} + h_{ID_B} \cdot P_{pub}$, we have:

$$\begin{aligned} c' &= t(d_{ID_B} + s_{ID_B})(s \cdot P + r \cdot (P_{ID_A} + R_{ID_A} + h_{ID_A} \cdot P_{pub})) \\ &= t(d_{ID_B} + s_{ID_B})((lt^{-1} - r(d_{ID_A} + s_{ID_A})) \cdot P \\ &+ r \cdot (P_{ID_A} + R_{ID_A} + h_{ID_A} \cdot P_{pub})) \\ &= t(d_{ID_B} + s_{ID_B})lt^{-1} \cdot P \\ &= l(d_{ID_B} + s_{ID_B}) \cdot P \\ &= l(P_{ID_B} + R_{ID_B} + h_{ID_B} \cdot P_{pub}) = c \end{aligned} \tag{6}$$

Then the correctness of our scheme is proved.

2. *Unforgeability:* We will prove our scheme can provide the unforgeability property by the following two theorems. The proof of the two theorems are given in the appendixes.
   - *Theorem 1:* If there is a type 1 adversary $\mathcal{A}1$ that breaks our proposed CLDVS scheme, then there exists an algorithm $\mathcal{F}$ which solves the ECDHP problem with non-negligible probability.
   - *Theorem 2:* If there is a type 2 adversary $\mathcal{A}2$ that breaks our proposed CLDVS scheme, then there exists an algorithm $\mathcal{F}$ which solves the EDCDH problem with non-negligible probability.
3. *Source Hiding:* Given a DVS (*r, s, t*) on a message *m*, even if a third party knows the signer *A*'s private key pair (d_{ID_A}, s_{ID_A}) and the verifier *B*'s private key pair (d_{ID_B}, s_{ID_B}), he cannot identify whether (d_{ID_A}, s_{ID_A}) or (d_{ID_B}, s_{ID_B}) has been used in the construction of the term *s* because he does not have the knowledge of the random numbers *l* used during the signing process. Hence, it is infeasible to determine whether the original signer or the designated verifier creates the signature.
4. *Non-Delegatability:* The problem of delegatability does not exist in our scheme. Because the construction of the term *s* requires the signer's private key pair (d_{ID_A}, s_{ID_A}), and it is impossible for the signer to delegate his signing capability to any third party without disclosing his private keys, a third party can not generate a valid signature (*r, s, t*) on a message *m*.

4. Comparison with Previous Scheme

In this section, we will compare the efficiency of our new scheme with the latest CLDVS schemes, i.e., Huang *et al*. scheme [8], Ming *et al*. scheme [13], Chen *et al*. scheme [3], Yang *et al*. scheme [20] and Xiao *et al*. scheme [19]. For the convenience of evaluating the computational cost, we define some notations as follows:

- TG_{exp}: The time of executing a modular exponentiation operation.
- TG_{pair}: The time of executing a pairing operation.
- TG_{pbsm}: The time of executing a pairing-based scalar multiplication operation of point.
- TG_{ebsm}: The time of executing an ECC-based scalar multiplication operation of point.
- TG_{padd}: The time of executing an addition operation of points
- TG_{inv}: The time of executing a modular inversion operation.
- TG_{mul}: The time of executing a general multiplication operation.
- TG_{add}: The time of executing a general addition operation.
- TG_{mtph}: The time of executing a map-to-point hash function.
- TG_h: The time of executing a one-way hash function.

In Table 1, we summarize the performance results of different CLDVS schemes. From the Table 1, we know that the client side requires only $3TG_{pair}+1TG_{pbsm}+TG_{mtph}+TG_h+1TG_{padd}$.

For the pairing-based scheme, to achieve the 1024-bit RSA level security, we have to use the Tate pairing defined over some supersingular elliptic curve on a finite field F_q, where the length of q is 512 bits at least [2]. For the ECC-based schemes, to achieve the same security level, we employ some secure elliptic curve on a finite field F_p or F_{2^m}, where the length of p is 160 bits at least [2]. Table 1 shows the results of the performance comparison.

Table 1. Performance evaluation of our protocol.

	Sign	Verify
Huang *et al.*'s scheme [8]	$1TG_{pair}+1TG_{pbsm}+1TG_{inv}+1TG_{mtph}+1TG_h+1TG_{padd}$	$3TG_{pair}+1TG_{pbsm}+TG_{mtph}+TG_h+1\ TG_{padd}$
Ming *et al.*'s scheme [13]	$1TG_{pair}+3\ TG_{pbsm}+TG_{mtph}+TG_h$	$3TG_{pair}+2TG_{pbsm}+1\ TG_{padd}+TG_h$
Chen *et al.*'s scheme [3]	$1TG_{pair}+1TG_{pbsm}+TG_{mtph}+TG_h$	$1TG_{pair}+1TG_{pbsm}+TG_{mtph}+TG_h$
Yang *et al.*'s scheme [20]	$1TG_{pair}+4\ TG_{pbsm}+TG_{mtph}+TG_h+1\ TG_{padd}$	$1TG_{pair}+2TG_{pbsm}+TG_{mtph}+TG_h+1\ TG_{padd}$
Xiao *et al.*'s scheme [19]	$3TG_{pbsm}+TG_{inv}+TG_{mtph}+TG_h$	$1TG_{exp}+2TG_{pair}+1TG_{pbsm}+TG_{mtph}+TG_h$
Our scheme	$2TG_{ebsm}+1TG_{inv}+3TG_{padd}+2\ TG_h+2TG_{mul}+2\ TG_{add}$	$2TG_{ebsm}+3TG_{padd}+2TG_h+1\ TG_{mul}+1TG_{add}$

As the main computational overheads, we only consider the bilinear pairing operation, the modular exponentiation, the scale multiplication and the pairing-based scale multiplication. From the theoretical analysis [4] and the experimental result [2, 5, 6, 7], we know the relative computation cost of the bilinear pairing operation, the modular exponentiation and the pairing-based scale multiplication are at least 19, 3 and 3 times that of the scalar multiplication separately. The running time of the sign algorithm of our scheme is 9.09% of Huang *et al*. scheme [8], 7.14% of Ming *et al*. scheme [13], 9.09% of Chen *et al*. scheme [3], 6.45% of Yang *et al*. scheme [20] and 22.22% of Xiao *et al*. scheme [19], the running time of the verify algorithm of our scheme is 3.33% of Huang *et al*. scheme [8], 3.18% of Ming *et al*. scheme [13], 9.09% of Chen *et al*. scheme [3], 8.00% of Yang *et al*. scheme [20] and 4.54% of Xiao *et al*. scheme [19]. Thus our

scheme is more useful and efficient than the previous schemes.

5. Conclusions

In this paper, we have proposed an efficient CLDVS scheme without bilinear pairings. We also prove the security of the scheme under random oracle. Compared with previous scheme, the new scheme reduces both the running time. Therefore, our scheme is more practical than the previous related schemes for practical application.

Acknowledgements

The authors thank the anonymous reviewers and the editors for their valuable comments. This research was supported by the Fundamental Research Funds for the Central Universities and the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20110141120003).

References

[1] Al-Riyami S. and Paterson G., "Certificateless Public Key Cryptography," *in Proceedings of the 9th International Conference on the Theory and Application of Cryptology and Information Security*, Taiwan, pp. 452-473, 2003.

[2] Cao X. and Kou W., "A Pairing-Free Identity-Based Authenticated Key Agreement Scheme with Minimal Message Exchanges," *Information Sciences*, vol. 180, no. 15, pp. 2895-2903, 2010.

[3] Chen H., Song S., Zhang T., and Song G, "An Efficient Certificateless Short Designated Verifier Signature Scheme," *in Proceedings of the 4th International Conference on Wireless Networking and Mobile Computing*, Dalian, pp. 1-6, 2008.

[4] Chen L., Cheng Z., and Smart N., "Identity-Based Key Agreement Protocols from Pairings," *International Journal Information Security*, vol. 6, no. 4, pp. 213-241, 2007.

[5] He D., Chen J., and Hu J., "An ID-Based Proxy Signature Schemes without Bilinear Pairings," *Annals of Telecommunications*, vol. 66, no. 11-12, pp. 657-662, 2011.

[6] He D., Chen J., and Hu J. "A Pairing-Free Certificateless Authenticated Key Agreement Protocol," *International Journal of Communication Systems*, vol. 25, no. 2, pp. 221-230, 2012.

[7] He D., Chen J., and Zhang R., "An Efficient Identity-Based Blind Signature Scheme without Bilinear Pairings," *Computers & Electrical Engineering*, vol. 37, no. 4, pp. 444-450, 2011.

[8] Huang X., Susilo W., Yi M., and Zhang F., "Certificateless Designated Verifier Signature Schemes," *in Proceedings of Advanced Information Networking and Application*, Vienna, pp. 15-19, 2006.

[9] Jakobsson M., Sako K., and Impagliazzo R., "Designated Verifier Proofs and their Applications," *in Proceedings of Advances in Cryptology: EUROCRYPT-International Conference on the Theory and Application of Cryptographic Techniques*, Spain, pp. 143-154, 1996.

[10] Kang B., Boyd C., and Dawson E., "A Novel Identity-Based Strong Designated Verifier Signature Scheme," *The Journal of Systems and Software*, vol. 82, no. 2, pp. 270-273, 2009.

[11] Kumar K., Shailaja G., and Saxena A., "Identity Based Strong Designated Verifier Signature Scheme," *Informatica*, vol. 18, no. 2, pp. 239-252, 2007.

[12] Lipmaa H., Wang G., and Bao F., "Designated Verifier Signature Schemes: Attacks, New Security Notions and a New Construction," *in Proceedings of the 32nd International Colloquium Automata, Languages and Programming*, Berlin, pp. 459-471, 2005.

[13] Ming Y., Shen X., and Wang Y., "Certificateless Universal Designated Verifier Signature Schemes," *Journal of China Universitles Posts and Telecommunications*, vol. 14, no. 3, pp. 85-91, 2007.

[14] Saeednia S., Kremer S., and Markowitch O., "An Efficient Strong Designated Verifier Signature Scheme," *in Proceedings of Information Security and Cryptology*, Berlin, pp. 40-54, 2003.

[15] Shamir A., "Identity-Based Cryptosystems and Signature Schemes," *in Proceedings of CRYPTO on Advances in Cryptology*, USA, pp. 47-53, 1985.

[16] Susilo W., Zhang F., and Mu Y., "Identity-Based Strong Designated Verifier Signature Schemes," *in Proceedings of Information Security and Privacy*, Berlin, pp. 313-324, 2004.

[17] Tso R., Okamoto T., and Okamoto E., "Practical Strong Designated Verifier Signature Schemes Based on Double Discrete Logarithms," *in Proceedings of the 1st SKLOIS Conference on Information Security and Cryptology*, Berlin, pp. 113-127, 2005.

[18] Wu S. and Zhu Y., "Improved Two-Factor Authenticated Key Exchange Protocol," *The International Arab Journal of Information Technology*, vol. 8, no. 4, pp. 430-439, 2011.

[19] Xiao Z., Yang B., and Li S., "Certificateless Strong Designated Verifier Signature Scheme," *in Proceedings of 2nd International Conference on e-Business and Information System Security*, Wuhan, pp. 1-5, 2010.

[20] Yang B., Hu Z., and Xiao Z., "Efficient Certificateless Strong Designated Verifier Signature Scheme," *in Proceedings of*

International Conference on Computational Intelligence and Security, Beijing, pp.432-436, 2009.

[21] Zhang J. and Mao J., "A Novel ID-Based Designated Verifier Signature Scheme," *Information Science*, vol. 178, no. 3, pp. 766-773, 2008.

Debiao He received his PhD degree in applied mathematics from School of Mathematics and Statistics, Wuhan University in 2009. Currently, he is a lecturer in Wuhan University. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Jianhua Chen received his PhD degree in applied mathematics from Wuhan University, Wuhan, China, in 1994. Currently, he is a professor in Wuhan University. His current research interests include number theory, information security and network security.

Appendixes

- *Proof for Theorem 1*: Suppose $\mathcal{F}$ is challenged with a ECCDH instance (P, $Q_1=aP$, $Q_2=bP$) and is tasked to compute $Q_3=ab \cdot P$. To do so, $\mathcal{F}$ picks two identity ID_I and ID_J at random as the challenged ID in this game, and gives $\{F_p, E/F_p, G, P, P_{pub}=Q_1, H_1, H_2\}$ to $\mathcal{A}1$ as the public parameters. Then $\mathcal{F}$ answers $\mathcal{A}1$'s queries as follows.
- *H1-Queries:* $\mathcal{F}$ maintains a hash list L_{H_1} of tuple $(ID_i, R_{ID_i}, P_{ID_i}, d_{ID_i}, h_{ID_i})$ as explained below. The list is initially empty. When $\mathcal{A}1$ makes a hash oracle query on ID_i, if the query ID_j has already appeared on L_{H_1}, then the previously defined value is returned. Otherwise, $\mathcal{F}$ acts as described in the partial private key extraction queries.
- *H2-Queries:* $\mathcal{F}$ maintains a hash list L_{H_2} of tuple (m_j, c_j, h_j). When $\mathcal{A}1$ makes $H2$ queries for identity ID_i on the message m_j, $\mathcal{F}$ chooses a random value $h_j \in Z_n^*$, sets $h_j=H_2(m_j, c_j)$ and adds (m_j, c_j, h_j) to L_{H_2}, and sends h_j to $\mathcal{A}1$.
- *Partial Private Key Extraction Queries:* $\mathcal{A}1$ is allowed to query the extraction oracle for an identity ID_i. $\mathcal{F}$ query H_1 oracle, ID_i is on L_{H_1}, then $\mathcal{F}$ response with $(ID_i, R_{ID_i}, P_{ID_i}, d_{ID_i}, h_{ID_i})$. Otherwise, if simulates the oracle as follows. It chooses $a_i, b_i \in Z_n^*$ at random, sets $R_{ID_i}=a_i \cdot P_{pub}+b_i \cdot P$, $d_{ID_i}=b_i$, $h_{ID_i}=H_1(ID_i, R_{ID_i}) \leftarrow -a_i \bmod n$, response with $(ID_i, R_{ID_i}, P_{ID_i}, d_{ID_i}, h_{ID_i})$, and inserts $(ID_i, R_{ID_i}, P_{ID_i}, d_{ID_i}, h_{ID_i})$ into L_{H_1}. Note that $(R_{ID_i}, d_{ID_i}, h_{ID_i})$ generated in this way satisfies the equation $d_{ID} \cdot P=R_{ID}+h_{ID} \cdot P_{pub}$ in the partial private key extraction algorithm. It is a valid secret key.
- *Public Key Extraction Queries:* $\mathcal{F}$ maintains a list L_{pk} of tuple $(ID_i, s_{ID_i}, pk_{ID_i})$ which is initially empty. When $\mathcal{A}1$ queries on input ID_i, $\mathcal{F}$ checks whether L_{pk} contains a tuple for this input. If it does, the previously defined value is returned. Otherwise, if $ID_i \neq ID_J$, $\mathcal{F}$ picks a random value $s_{ID} \in Z_n^*$, computes $P_{ID_i}=s_{ID_i} \cdot P$. If $ID_i=ID_J$, $\mathcal{F}$ sets $P_{ID_i}=b \cdot P$ and $s_{ID_i}=\perp$. $\mathcal{F}$ queries Partial Private Key Extraction Queries with ID_i and P_{ID_i} and get response R_{ID_i}. At last $\mathcal{F}$returns $pk_{ID_i}=\{P_{ID_i}, R_{ID_i}\}$ and adds $(ID_i, s_{ID_i}, pk_{ID_i})$ to the L_{pk}.
- *Private Key Extraction Queries:* For query on input ID_i, If $ID_i=ID_I$ or $ID_i=ID_J$, $\mathcal{F}$ stops and outputs "failure". Otherwise, $\mathcal{F}$ performs as follows:
  If the L_{H_1} and the L_{pk} contain the corresponding tuple $(ID_i, R_{ID_i}, s_{ID_i}, h_{ID_i})$ and the tuple $(ID_i, s_{ID_i}, pk_{ID_i})$ respectively, $\mathcal{F}$ sets $sk_{ID_i}=\{d_{ID_i}, s_{ID_i}\}$ and sends it to $\mathcal{A}1$. Otherwise, $\mathcal{F}$ makes a partial private key extraction query and a public key extraction query on ID_i, then simulates as the above process and sends $sk_{ID_i}=\{d_{ID_i}, s_{ID_i}\}$ to $\mathcal{A}1$.
- *Public Key Replacement:* When $\mathcal{A}1$ queries on input $(ID_i, \widetilde{pk_{ID_i}})$, $\mathcal{F}$ checks whether the tuple $(ID_i, s_{ID_i}, pk_{ID_i})$ is contained in the L_{pk}. If it does, F sets $pk_{ID_i}=\widetilde{pk_{ID_i}}$ and adds the tuple $(ID_i, \perp, pk_{ID_i})$ to the L_{pk}.
- *Signing Queries:* When a message m, the signer A's identity ID_A, the designated verifier's B's identity ID_B is coming, F acts as follows: F first checks that whether tuple $(ID_i, R_{ID_i}, d_{ID_i}, h_{ID_i})$ and $(ID_i, s_{ID_i}, pk_{ID_i})$ are in L_{H_1} and L_{pk} separately. If yes, it just retrieves $(d_{ID_A}, s_{ID_A}, P_{ID_B}, R_{ID_B})$ from the tables and uses these values to sign for the message according to the signing algorithm described in the scheme. It outputs the signature (r, s, t) for the message m and stores the value $H_2(m, c)$ in L_{H_2} for consistency. If ID_A or ID_B has not been queried to

the partial private extraction oracle and the private extraction oracle, $\mathcal{F}$ executes the simulation of the partial private extraction oracle and the private extraction oracle, then uses the corresponding secret key to sign the message.

Finally, $\mathcal{A}1$ stops and outputs a signature S={*r, s, t*} on the message *m* with the signer *A*'s identity ID_A, and *A*'s private key sk_{ID_A}, the designated verifier's *B*'s identity ID_B public key pk_{ID_B}, which satisfies the following equation $Verify(params, m, ID_A, sk_{ID_B}, pk_{ID_A}, S) = 1$. If $ID_A \neq ID_J$ or $ID_B \neq ID_J$, $\mathcal{F}$ outputs "failure" and aborts. Otherwise, $\mathcal{F}$ recovers the tuple $(ID_I, R_{ID_I}, d_{ID_I}, h_{ID_I})$ and $(ID_J, R_{ID_J}, d_{ID_J}, h_{ID_J})$ from L_{H_1}, the tuple $(ID_I, s_{ID_I}, pk_{ID_I})$ and $(ID_J, s_{ID_J}, pk_{ID_J})$ from L_{pk} and the tuple (*m, c, h*) from L_{H_2}. Then, we have:

$$c = t(d_{ID_J} + s_{ID_J})(s \cdot P + r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub})) \quad (7)$$

Since $P_{IDI}=s_{IDI} \cdot P$, $R_{IDJ}=b \cdot P$ and $P_{pub}=a \cdot P$, we could have:

$$\begin{aligned} c &= t(d_{ID_J} + s_{ID_J})(s \cdot P + r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub})) \\ &= t \cdot s \cdot Q_2 + t \cdot s \cdot s_{ID_J} \cdot P + t \cdot r \cdot s_{ID_I} \cdot Q_2 \\ &+ t \cdot r \cdot d_{ID_I} \cdot Q_2 + t \cdot r \cdot h_{ID_I} \cdot Q_3 \\ &+ t \cdot s_{ID_J} \cdot r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub}) \end{aligned} \quad (8)$$

Then we have

$$\begin{aligned} t \cdot r \cdot h_{ID_I} \cdot Q_3 &= c - t \cdot s \cdot Q_2 - t \cdot s \cdot s_{ID_J} \cdot P - \\ & t \cdot r \cdot s_{ID_I} \cdot Q_2 - t \cdot r \cdot d_{ID_I} \cdot Q_2 \\ & - t \cdot s_{ID_J} \cdot r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub}) \end{aligned} \quad (9)$$

and

$$\begin{aligned} Q_3 &= (t \cdot r \cdot h_{ID_I})^{-1}(c - t \cdot s \cdot Q_2 - \\ & t \cdot s \cdot s_{ID_J} \cdot P - t \cdot r \cdot s_{ID_I} \cdot Q_2 - t \cdot r \cdot d_{ID_I} \cdot Q_2 - \\ & t \cdot s_{ID_J} \cdot r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub})) \end{aligned} \quad (10)$$

Obviously, the probability of $ID_A=ID_I$ or $ID_B=ID_J$ is $\frac{1}{q_s(q_s-1)}$. Thus, we can obtain $Q_3=ab \cdot P$ with the probability $\frac{2\varepsilon}{q_s(q_s-1)}$. In other words, given (*P*, $Q_1=aP$, $Q_2=bP$), F can solve the ECCDH problem with non-negligible probability $\frac{2\varepsilon}{q_s(q_s-1)}$, which is in contradiction with the ECCDH assumption.

- *Proof for Theorem 2:* Suppose that there is a type 2 Adversar $\mathcal{A}2$ who can breaks our scheme with probability ε, when making q_e extraction queries, q_s signing queries and q_h hashing queries respectively. Then, we will show how to use the ability of $\mathcal{A}2$ to construct an algorithm $\mathcal{F}$ solving the ECCDH. Suppose $\mathcal{F}$ is challenged with a ECCDH instance (*P*, $Q_1=aP$, $Q_2=bP$) and is tasked to compute $Q_3=ab \cdot P$. To do so, $\mathcal{F}$ chooses a random $x \in Z_n^*$, computes $Q=xP$, picks two identity ID_I and ID_J at random as the challenged *ID* in this game, and gives public parameters $\{F_p, E/F_p, G, P, P_{pub}=Q_1, H_1, H_2\}$ and the master key *s* to $\mathcal{A}2$. Then $\mathcal{F}$ answers $\mathcal{A}2$'s queries as follows.
  - *H1-Queries:* $\mathcal{F}$ maintains a hash list L_{H_1} of tuple $(ID_i, R_{ID_i}, P_{ID_i}, d_{ID_i}, h_{ID_i})$ indexed by ID_i. The list is initially empty. When $\mathcal{A}2$ makes a hash oracle query on ID_i, if the query ID_i has already appeared on L_{H_1}, then the previously defined value is returned. Otherwise, $\mathcal{F}$ makes the partial private key extraction query with ID_i, and sends h_{ID_i} to $\mathcal{A}2$.
  - *H2-Queries:* $\mathcal{F}$ maintains a hash list L_{H_2} of tuple (m_j, c_j, h_j). When $\mathcal{A}2$ makes *H2* queries for identity ID_i on the message m_j, $\mathcal{F}$ chooses a random value $h_j \in Z_n^*$, sets $h_j=H_2(m_j, c_j)$ and adds (m_j, c_j, h_j) to L_{H_2}, and sends h_j to $\mathcal{A}2$.
  - *Partial Private Key Extraction Queries:* $\mathcal{A}2$ is allowed to query the extraction oracle for an identity ID_i. $\mathcal{F}$ query H_1 oracle, ID_i is on L_{H_1}, then $\mathcal{F}$ response with $(ID_i, R_{ID_i}, P_{ID_i}, d_{ID_i}, h_{ID_i})$. Otherwise, if simulates the oracle as follows. It chooses r_{ID_i} at random, sets $R_{ID_i} = r_{ID_i} \cdot P$, $h_{ID_i} = H_1(ID_i, R_{ID_i}, P_{ID_i})$ and $d_{ID_i} = r_{ID_i} + x \cdot h_{ID_i}$, response with $(ID_i, R_{ID_i}, P_{ID_i}, d_{ID_i}, h_{ID_i})$, and inserts $(ID_i, R_{ID_i}, P_{ID_i}, d_{ID_i}, h_{ID_i})$ into L_{H_1}.
  - *Private Key Extraction Queries:* $\mathcal{F}$ maintains a list L_{sk} of tuple $(ID_i, d_{ID_i}, s_{ID_i})$ which is initially empty. For query on input ID_i, $\mathcal{F}$ performs as follows:
    1. If the query ID_i has already appeared on L_{sk}, then the previously defined value is returned.
    2. Else if $ID_i=ID_I$, F sets $s_{ID_i} = \perp$, $P_{ID_i} = Q_1$.
    3. Else if $ID_i=ID_J$, F sets $s_{ID_i} = \perp$, $P_{ID_i} = Q_2$.
    4. Else if $ID_i \neq ID_I$ and $ID_i \neq ID_J$, $\mathcal{F}$ generates a random number $s_{ID_i} \in Z_n^*$, compute $P_{ID_i} = s_{ID_i} \cdot P$.

    Then $\mathcal{F}$ looks up L_{H_1} to get the tuple $(ID_i, R_{ID_i}, d_{ID_i}, h_{ID_i})$, and add the tuple

$(ID_i, d_{ID_i}, s_{ID_i}, P_{ID_i})$ to L_{sk}, and sends $sk_{ID_i} = \{d_{ID_i}, s_{ID_i}\}$ to $\mathcal{A}2$.

- *Public Key Extraction Queries:* $\mathcal{F}$ maintains a list L_{pk} of tuple (ID_i, pk_{ID_i}) which is initially empty. When $\mathcal{A}2$ queries on input ID_i, $\mathcal{F}$ checks whether L_{pk} contains a tuple for this input. If it does, the previously defined value is returned. Otherwise, $\mathcal{F}$ looks up the tables L_{H_1} and L_{sk} and gets the tuple $(ID_i, R_{ID_i}, d_{ID_i}, h_{ID_i})$ and $(ID_i, d_{ID_i}, s_{ID_i}, P_{ID_i})$ separately. At last $\mathcal{F}$ returns $pk_{ID_i} = \{P_{ID_i}, R_{ID_i}\}$ and adds (ID_i, pk_{ID_i}) to the L_{pk}.
- *Signing Queries:* When a message *m*, the signer *A*'s identity ID_A, the designated verifier's *B*'s identity ID_B is coming, $\mathcal{F}$ acts as follows: If $ID_A=ID_I$ or $ID_A=ID_J$, $\mathcal{F}$ terminates the simulation. Otherwise, $\mathcal{F}$ first checks that whether tuple $(ID_i, R_{ID_i}, d_{ID_i}, h_{ID_i})$ and (ID_i, pk_{ID_i}) are in L_{H_1} and L_{pk} separately. If yes, it just retrieves $(d_{ID_A}, s_{ID_A}, P_{ID_B}, R_{ID_B})$ from the tables and uses these values to sign for the message according to the signing algorithm described in the scheme. It outputs the signature (*r, s, t*) for the message *m* and stores the value $H_2(m, c)$ in L_{H_2} for consistency. If ID_A or ID_B has not been queried to the partial private extraction oracle and the private extraction oracle, $\mathcal{F}$ executes the simulation of the partial private extraction oracle and the private extraction oracle, then uses the corresponding secret key to sign the message.

  Finally, $\mathcal{A}2$ stops and outputs a signature $S=\{r, s, t\}$ on the message *m* with the signer *A*'s identity ID_A, and *A*'s private key sk_{ID_A}, the designated verifier's *B*'s identity ID_B public key pk_{ID_B}, which satisfies the following equation $Verify(params, m, ID_A, sk_{ID_B}, pk_{ID_A}, S) = 1$. If $ID_A \neq ID_I$ or $ID_B \neq ID_J$, $\mathcal{F}$ outputs "failure" and aborts. Otherwise, $\mathcal{F}$ recovers the tuple $(ID_I, R_{ID_I}, d_{ID_I}, h_{ID_I})$ and $(ID_J, R_{ID_J}, d_{ID_J}, h_{ID_J})$ from L_{H_1}, the tuple $(ID_I, s_{ID_I}, pk_{ID_I})$ and $(ID_J, s_{ID_J}, pk_{ID_J})$ from L_{pk} and the tuple (*m, c, h*) from L_{H_2}. Then, we have:

$$c = t(d_{ID_J} + s_{ID_J})(s \cdot P + r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub})) \quad (11)$$

Since $P_{ID_I} = Q_1 = a \cdot P$, $R_{ID_J} = Q_2 = b \cdot P$ and $P_{pub}=x \cdot P$, we could have:

$$\begin{aligned} c &= t(d_{ID_J} + s_{ID_J})(s \cdot P + r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub})) \\ &= t \cdot s \cdot R_{ID_J} + t \cdot d_{ID_J} \cdot r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub}) + t \cdot s \cdot P_{ID_J} \\ &+ t \cdot r \cdot Q_3 + t \cdot d_{ID_I} \cdot r \cdot Q_2 + t \cdot r \cdot h_{ID_I} \cdot x \cdot Q_2 \end{aligned} \quad (12)$$

Then we have:

$$\begin{aligned} t \cdot r \cdot Q_3 &= c - t \cdot s \cdot R_{ID_J} - \\ & t \cdot d_{ID_J} \cdot r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub}) - \\ & t \cdot s \cdot P_{ID_J} - t \cdot d_{ID_I} \cdot r \cdot Q_2 - t \cdot r \cdot h_{ID_I} \cdot x \cdot Q_2 \end{aligned} \quad (13)$$

and

$$\begin{aligned} Q_3 &= (t \cdot r)^{-1}(c - t \cdot s \cdot R_{ID_J} - \\ & t \cdot d_{ID_J} \cdot r \cdot (P_{ID_I} + R_{ID_I} + h_{ID_I} \cdot P_{pub}) - \\ & t \cdot s \cdot P_{ID_J} - t \cdot d_{ID_I} \cdot r \cdot Q_2 - t \cdot r \cdot h_{ID_I} \cdot x \cdot Q_2) \end{aligned} \quad (14)$$

Obviously, the probability of $ID_A=ID_I$ or $ID_B=ID_J$ is $\frac{1}{q_s(q_s-1)}$. Thus, we can obtain $Q_3=ab \cdot P$ with the probability $\frac{2\varepsilon}{q_s(q_s-1)}$. In other words, given (P, $Q_1=aP$, $Q_2=bP$), $\mathcal{F}$ can solve the ECCDH problem with non-negligible probability $\frac{2\varepsilon}{q_s(q_s-1)}$, which is in contradiction with the ECCDH assumption.