A More Efficient 1–Checkable Secure Outsourcing Algorithm for Bilinear Maps

Öznur Kalkar, Mehmet Sabir Kiraz, İsa Sertkaya, Osmanbey Uzunkol

A More Efficient 1−Checkable Secure Outsourcing Algorithm for Bilinear Maps

Öznur Kalkar[1], Mehmet Sabir Kiraz[1], İsa Sertkaya[1], and Osmanbey Uzunkol[2]

[1] Mathematical and Computational Sciences
TÜBİTAK BİLGEM, Turkey
{oznur.arabaci,mehmet.kiraz,isa.sertkaya}@tubitak.gov.tr
[2] FernUniversität in Hagen, Faculty of Mathematics and Computer Science, Germany
osmanbey.uzunkol@gmail.com

Abstract. With the rapid advancements in innovative technologies like cloud computing, internet of things, and mobile computing, the paradigm to delegate the heavy computational tasks from trusted and resource-constrained devices to potentially untrusted and more powerful services has gained a lot of attention. Ensuring the verifiability of the outsourced computation along with the security and privacy requirements is an active research area. Several cryptographic protocols have been proposed by using pairing-based cryptographic techniques based on bilinear maps of suitable elliptic curves. However, the computational overhead of bilinear maps forms the most expensive part of those protocols. In this paper, we propose a new 1−checkable algorithm under the one-malicious version of a two-untrusted-program model. Our solution is approximately twice as efficient as the single comparably efficient 1−checkable solution in the literature, and requires only 4 elliptic curve point additions in the preimage and 6 field multiplications in the image of the bilinear map.

Keywords: Outsourcing computation, bilinear maps, security, privacy

1 Introduction

The flexibility of cloud computing comes with innovative and cost-effective solutions for both individuals and enterprises. The main advantages include ubiquitous and on-demand network access, using resources in a pay-per manner, scalability of the underlying services, location independence, and rapid elasticity [21]. Accordingly, the proposals to delegate the mostly expensive and sometimes energy consuming computations from trustful resource-constrained devices (e.g., sensors, RFID cards, SIM cards) to untrusted or even malicious external applications and services (e.g., cloudlets, large scale cloud service providers) have been increasing dramatically. The demand of weak clients to outsource costly computational tasks to external powerful services while preserving security and privacy becomes more and more crucial in accordance with the interplay of cloud computing paradigm with the new advancements and novel solutions in internet

of things (IoT) and mobile technologies. In order to assure security and privacy in such scenarios, the most crucial part of the underlying cryptographic techniques is not to get the private cryptographic values out of the secure memory. Therefore, one can only apply masking tecniques in such a way that it can be outsourced to a (potentially) untrusted server to perform further computations. However, one also has to verify simultaneously that the computations are indeed correct without having an additional significant overhead. Hence, efficiently verifiable outsourcing to untrusted or even malicious services while ensuring the desired level of security and privacy has become a subject of many recent research activities, and forms a new challenge especially in the presence of malicious services [18,9].

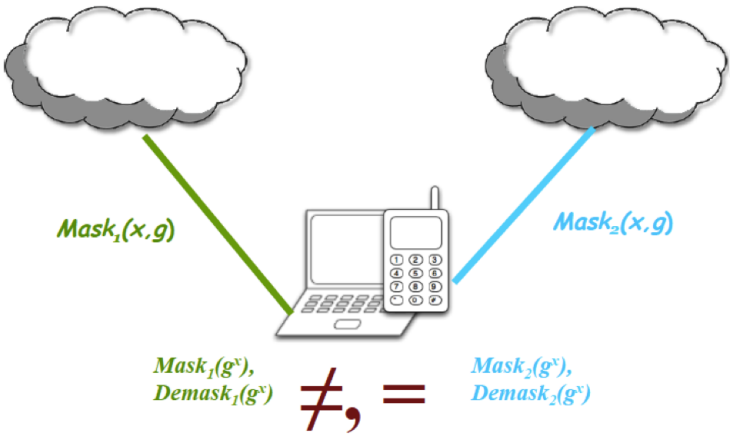

Fig. 1. A typical verifiable outsourcing mechanism with two servers

Many cryptographic applications realizing novel security and privacy solutions demand to use pairing-based primitives in order to overcome different challenges in the realm of cloud computing, internet of things and mobile technologies. Pairing-based cryptography is realized by using bilinear maps of carefully chosen elliptic curves [15,6,7]. Computing such bilinear maps is the most inefficient and expensive part of those solutions. Although, there are plenty of research results aiming to reduce the computational cost of the computation of bilinear maps [3,4,7,13,17], these computations are still far away of being feasible for resource-constrained environments. Even worse, new research results imply that the key sizes must be increased to achieve the desired security level [2].

1.1 Related work

Hohenberger and Lysyanskaya gave the first formal security model for secure and verifiable outsourcing of cryptographic computations in the presence of malicious powerful devices or services [14]. Subsequently, Chevallier-Mames *et al.*

proposed the first cryptographic protocol for outsourcing the computation of bilinear maps [11]. However, this delegation process brought some additional challenges. Firstly, no sensitive informations has to be revealed to the untrusted parties. However, one needs to establish a mechanism in which the validity of the outsourced computation can effectively be verified by the client, since the external device can otherwise cheat the client by deliberately manipulating the desired result.

Chen *et al.* classified the security models based on the number of servers being involved in the outsourcing process [10]:

- One-Untrusted Program (OUP): There exists a single malicious server performing the delegated computation.
- One-Malicious version of a Two-Untrusted Program (OMTUP): There exist two untrusted servers performing the delegated computation but exactly one of them may also behave maliciously.
- Two-Untrusted Program (TUP): There exist two untrusted servers performing the delegated computation and both of them may simultaneously behave maliciously, but they do not maliciously collude.

Tian *et al.* improves the checkability result of the work of Chen *et al.*, [10] with some additional computations under the TUP assumption in [25]. Both studies have two main phases; offline computation and online computation. In the offline phase, the client prepares some precomputed values. The online phase consists of masking the sensitive values and using the precomputed values from the offline phase to prepare the necessary requests. Both algorithms suffer from being only probabilistically checkable implementations (both are $1/2-$checkable implementations). However, as outlined by Canard *et al.* [8], an outsourcing algorithm not ensuring sufficient checkability property could lead to severe security issues especially in the case of authentication protocols. Hence, it is highly desirable to have an efficient outsourcing mechanism having $1-$checkable property, i.e. potentially malicious external services can never cheat the clients since the verification of the outsourced computation works with probabilty 1.

Chevallier-Mames *et al.* proposed the first $1-$checkable outsourcing mechanism for bilinear maps in the OUP setting [11] which subsequently was improved by Kang *et al.* [16] and Canard *et al.* [8]. These mechanisms suffer unfortunately from being inefficient since directly embedding the computation of bilinear maps inside the trusted devices are more efficient than utilizing their outsourcing mechanism.

Chen *et al.* used two servers under the OMTUP assumption with an offline phase to propose a $1/2-$checkable outsourcing mechanism using 5 point additions in the preimage of the bilinear map and 4 multiplications in the image of the bilinear map. A similar $1/2-$checkable result was later proposed by Tian *et al.* [25] improving the online phase with 4 point additions in the preimage of the bilinear map and 3 multiplications in the image of the bilinear map. Subsequently, Lin *et al.* also proposed another $1/2-$checkable outsourcing mechanism improving the tuple-sizes and offline phase of the previous mechanisms [19].

Recently, Luo *et al.* proposed the first efficient $1-$checkable outsourcing mechanism for bilinear maps under the OMTUP assumption requiring only 8 point additions in the preimage of the bilinear map and 6 multiplications in the image of the bilinear map [20].

1.2 Our contributions

In this paper, we propose a new and highly efficient outsourcing mechanism for bilinear maps following the steps of Chen *et al.* and Tian *et al.*'s work. The main advantage our mechanism is to provide $1-$checkability property which ensures that external servers never can cheat the users. Our $1-$checkable outsourcing mechanism is given under the OMTUP assumption for bilinear maps. Compared to the only efficient solution of Luo *et al.* [20], our mechanism is approximately twice more efficient requiring only 4 point additions in the preimage of the bilinear map and 6 multiplications in the image of the bilinear map. Furthermore, our mechanism is almost as efficient as existing $1/2-$checkable solutions. In particular, our mechanism only needs 3 more multiplications in the image of the bilinear map in order to achieve a $1-$checkable outsourcing algorithm.

2 Security Model

We follow the security model proposed by Hohenberger and Lysyanskaya [14], and we use one-malicious version of a two-untrusted program (OMTUP) model. More concretely, there are two untrusted cloud servers in this model performing the outsourced computation where only one of them is assumed to be malicious.

Definition 1. *A pair of algorithms (T, U_1, U_2) are an α-efficient implementation of an algorithm* Alg *if* (1) *they are an outsource-secure implementation of* Alg*, and* (2) $\forall$ *inputs x, the running time of T is $\leq$ an α-multiplicative factor of the running time of* $\mathsf{Alg}(x)$*.*

Definition 2. *A pair of algorithms (T, U_1, U_2) are an β-checkable implementation of an algorithm* Alg *if* (1) *they are an outsource-secure implementation of* Alg*, and* (2) $\forall$ *inputs x, if $U_i', i = 1, 2$ deviates from its advertised functionality during the execution of $T^{(U_1', U_2')}(x), T$ will detect the error with probability $\geq \beta$.*

Definition 3. *A pair of algorithms (T, U_1, U_2) are an (α, β)-outsource secure implementation of an algorithm* Alg *if they are both α-efficient and β-checkable.*

3 Verifiable Secure Outsourcing of Bilinear Maps

3.1 Bilinear Maps

Definition 4. *Let $(\mathbb{G}_1, +)$ and $(\mathbb{G}_2, +)$ be two additive cyclic groups of order q and $(\mathbb{G}_3, \cdot)$ be a multiplicative cyclic group of order q, where q is a prime number. Let also P_1 and P_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Assume that Discrete Logarithm Problem (DLP) is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$. A bilinear map is a map $e: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$ satisfying the following properties [5,24,12]:*

- ***Bilinearity:*** *For all $P, P' \in \mathbb{G}_1, Q, Q' \in \mathbb{G}_2$, e is a group homomorphism in each component, i.e.*
  1. $e(P + P', Q) = e(P, Q) \cdot e(P', Q)$,
  2. $e(P, Q + Q') = e(P, Q) \cdot e(P, Q')$.
- ***Non-degeneracy:*** *e is non-degenerate in each component, i.e.,*
  1. *For all $P \in \mathbb{G}_1^*$, there is an element $Q \in \mathbb{G}_2$ such that $e(P, Q) \neq 1$,*
  2. *For all $Q \in \mathbb{G}_2^*$, there is an element $P \in \mathbb{G}_1$ such that $e(P, Q) \neq 1$.*
- ***Computability:*** *There exists an algorithm which computes the bilinear map e efficiently.*

3.2 Precomputations

In order to speed up the algorithms, some computations are required to be performed offline. For this, we will use BPV+ method proposed by Wang et al. [26] and applied by Tian et al. [25]. BPV+ uses two tables; a static table ST, and a dynamic table DT. Values in the static table are used to construct the values in the dynamic table. After each invocation of $Rand()$, values in the dynamic table needs to be refreshed in an idle time of the device. Dynamic table consists of the following tuple:

$$(\alpha P_1, xP_1, yP_1, \alpha P_1 - xP_1, \alpha P_1 - yP_1, mP_2, nP_2, \beta P_2, e(P_1, P_2)^{\alpha\beta}).$$

Rand. We now describe the precomputation algorithm called $Rand()$. At each invocation of $Rand()$, a tuple is returned and removed from the dynamic table. Then, at some convenient time, a new tuple is created from the values in the static table and stored in the dynamic table. Below, we describe the creation process of static and dynamic tables.

- `Preprocessing`:
  1. Generate n random elements $\alpha_1, \cdots, \alpha_n \in \mathbb{Z}_q$.
  2. For $j = 1, \cdots, n$ compute $\beta_{j_1} = \alpha_j P_1$, $_{j_2} = \alpha_j P_2$.
  3. Compute $e(P_1, P_2)$.
  4. Store the values of α_j, β_{j_1}, β_{j_2} and, $e(P_1, P_2)$ in the static table ST.
- `Point generation`:
  1. Randomly generate $S \subset \{1, \cdots, n\}$ such that $|S| = k$.
  2. For each $j \in S$, randomly select $K_j \in \{1, \cdots, h-1\}$, where $h > 1$ is a small integer.
  3. Compute
  $$\alpha = \sum_{j \in S} \alpha_j K_j \bmod q.$$
  If $\alpha = 0 \bmod q$, start again. Otherwise, compute
  $$\alpha P_1 \equiv \sum_{j \in S} K_j \cdot \beta_{j_1} \bmod q.$$
  4. Following the above procedure, compute similarly the elements $xP_1, yP_1, mP_2, nP_2, \beta P_2$.
  5. Compute $\alpha P_1 - xP_1, \alpha P_1 - yP_1$.
  6. Calculate $e(P_1, P_2)^{\alpha\beta}$.
  7. Store $(\alpha P_1, xP_1, yP_1, \alpha P_1 - xP_1, \alpha P_1 - yP_1, mP_2, nP_2, \beta P_2, e(P_1, P_2)^{\alpha\beta})$ in the dynamic table DT.

3.3 Our Algorithm: `OutPair`

Let T denote the trusted device with limited computational power which wants to outsource the pairing computation, $U_i(M,N) \to e(M,N), \quad i \in \{1,2\}$ denotes untrusted party U_i taking (M,N) as input and returning $e(M,N)$.

Our algorithm `OutPair` takes $A \in \mathbb{G}_1, B \in \mathbb{G}_2$ as inputs and produces $e(A,B)$ as output.

1. T runs $Rand()$ in order to get the following values:
$$(\alpha P_1, xP_1, yP_1, \alpha P_1 - xP_1, \alpha P_1 - yP_1, mP_2, nP_2, \beta P_2, e(\alpha P_1, \beta P_2)).$$
2. T computes
$$A - \alpha P_1, B - nP_2, B - mP_2, B - \beta P_2,$$
and queries U_1 in random order as follows:
   - $U_1(A - \alpha P_1, mP_2) \to A_{11}$,
   - $U_1(A - \alpha P_1, B - nP_2) \to A_{12}$,
   - $U_1(\alpha P_1 - xP_1, B - \beta P_2) \to A_{13}$,
   - $U_1(yP_1, B - \beta P_2) \to A_{14}$.

   Then similarly, T queries U_2 in random order as follows:
   - $U_2(A - \alpha P_1, nP_2) \to A_{21}$,
   - $U_2(A - \alpha P_1, B - mP_2) \to A_{22}$,
   - $U_2(\alpha P_1 - yP_1, B - \beta P_2) \to A_{23}$,
   - $U_2(xP_1, B - \beta P_2) \to A_{24}$.
3. Upon receiving computation results from both servers, T checks if
   - $A_{11}A_{22} \stackrel{?}{=} A_{21}A_{12}$,
   - $A_{13}A_{24} \stackrel{?}{=} A_{23}A_{14}$.

   If the check is not successful, then T outputs an "error". Otherwise, T computes and outputs
$$A_{11}A_{22}A_{13}A_{24}e(P_1,P_2)^{\alpha\beta} \quad .$$

3.4 Security Analysis

Theorem 1. *Under the OMTUP assumption, the algorithms $(T,(U_1,U_2))$ are an outsource-secure $1-$checkable implementation of `OutPair` for calculating the bilinear map $e(A,B) = OutPair(A,B)$, where the input $(A \in G_1, B \in G_2)$ may be honest secret, honest protected, or adversarial protected.*

Proof. The correctness of $(T,(U_1,U_2))$ is straight forward. If T wants to evaluate $e(A,B)$, at the last step of `OutPair` T needs to compute $D_1D_2e(P_1,P_2)^{\alpha\beta}$. Taking D_1, D_2, α, β as defined in OutPair:

$$\begin{aligned}
D_1D_2e(P_1,P_2)^{\alpha\beta} &= A_{11}A_{22}A_{13}A_{24}e(P_1,P_2)^{\alpha\beta} \\
&= e(A - \alpha P_1, mP_2)e(A - \alpha P_1, B - mP_2)e(\alpha P_1 - xP_1, B - \beta P_2) \\
&\quad e(xP_1, B - \beta P_2)e(P_1,P_2)^{\alpha\beta} \\
&= e(A - \alpha P_1, B)e(\alpha P_1, B - \beta P_2)e(P_1,P_2)^{\alpha\beta} \\
&= e(A,B)e(\alpha P_1, -\beta P_2)e(P_1,P_2)^{\alpha\beta} \\
&= e(A,B).
\end{aligned}$$

Next, we prove the security of the algorithm as follows. First of all, let (E, U_1', U_2') be a probabilistic polynomial-time (PPT) adversary that interacts with a PPT algorithm T in the OMTUP model, and S_1, S_2 be the simulators in Pair One, Pair Two, respectively.

- **Pair One:** We below show that $EVIEW^i_{real} \sim EVIEW_{ideal}$ to ensure that the external adversary E learns nothing.
  If the input (A, B) is honest protected, or adversarial protected, then the simulator S_1 behaves the same way as in the real execution. There is no secret input, so S_1 does not require to access secret inputs. Hence, suppose that (A, B) is an hones secret input. Then the simulator S_1 behaves as follows: Upon receiving the input in round i, S_1 ignores it, selects a series of random points of the form $((a - a_1)P_1, b_1P_2, (b - b_2)P_2, (a_1 - a_2)P_1, b_3P_2, a_3P_1, b_2P_2, (b - b_1)P_2, (a_1 - a_3)P_1, a_2P_1)$, and makes the following queries to U_1' and U_2'.
  - $U_1((a - a_1)P_1, b_1P_2) \rightarrow A_{11}$,
  - $U_1((a - a_1)P_1, (b - b_2)P_2) \rightarrow A_{12}$,
  - $U_1((a_1 - a_2)P_1, b_3P_2) \rightarrow A_{13}$,
  - $U_1(a_3P_1, b_3P_2) \rightarrow A_{14}$.
  - $U_2((a - a_1)P_1, b_2P_2) \rightarrow A_{21}$,
  - $U_2((a - a_1)P_1, (b - b_1)P_2) \rightarrow A_{22}$,
  - $U_2((a_1 - a_3)P_1, b_3P_2) \rightarrow A_{23}$,
  - $U_2(a_2P_1, b_3P_2) \rightarrow A_{24}$.

  If $A_{11}A_{22} = A_{21}A_{12}$ and $A_{13}A_{24} = A_{23}A_{14}$, S_1 sets $Y^i_p = \emptyset, Y^i_u = \emptyset, rep^i = 0$; otherwise S_1 sets $Y^i_p = error, Y^i_u = \emptyset, rep^i = 1$. For both cases, S_1 saves the appropriate states. In the real and ideal experiments, the input distributions are computationally indistinguishable for U_1' and U_2'. In the ideal experiment, inputs are chosen uniformly random. In the real experiment, inputs that are generated by *Rand* are re-randomized. In the $i-$th round, there are two possibilities:
  - U_1' and U_2' performs honestly: S_1 gives the correct output which is the same as the output of $(T, (U_1, U_2))$.
  - One of U_1' and U_2' gives an incorrect output: In the ideal experiment, S_1 detects the mistake and result in "error". In the real experiment, T does the same thing.

  Thus, $EVIEW^i_{real} \sim EVIEW^i_{ideal}$ and by the hybrid argument, we conclude that $EVIEW_{real} \sim EVIEW_{ideal}$.
- **Pair Two:** We now show that $UVIEW_{real} \sim UVIEW_{ideal}$ which ensures that the untrusted software (U_1, U_2) learns no information.
  Upon receiving an input on round i, S_2 behaves exactly like S_1, generates random inputs and saves the appropriate states. Also, In the ith round of the real experiment, T re-randomizes the inputs to (U_1', U_2'). Hence, for each round i, we have $UVIEW^i_{real} \sim UVIEW^i_{ideal}$. Hence, by the hybrid argument, we conclude that $UVIEW_{real} \sim UVIEW_{ideal}$.

□

4 Comparison

For the comparison table in Figure 4, we use the following abbreviations:

- **SM**: Scalar multiplication in $\mathbb{G}_1, \mathbb{G}_2$
- **ME**: Modular exponentiation in $\mathbb{G}_3$
- **PA**: Point addition in $\mathbb{G}_1, \mathbb{G}_2$
- **PC**: Pairing computation
- **FM**: Field multiplication in $\mathbb{G}_3$

Also, k is the size of the set used at step 1 of point generation, and h is the size of the set used at step 2 of point generation. For a more detailed explanation of k, h we refer to [22].

	Chen[10]	**Tian**[25]	**AKSU**[1]	**Lin**[19]	**Luo**[20]	**Ren**[23]	`OutPair`
Checkability							
	1/2	1/2	1/2	1/2	1	1	1
Client's Workload							
PA	5	4	4	4	8	8	4
FM	4	3	3	3	6	14	6
Servers's Workload							
PC	8	6	4	6	4	6	8
ME	0	0	0	0	0	4	0
Precomputations							
SM	9	3	2	2	0	8	0
ME	0	2	2	2	1	0	1
PA	0	5(k+h-3)	4(k+h-3)	4(k+h-3)	2(k+h-3) + 2	0	6(k+h-3) + 2
PC	3	0	0	0	0	6	0

Fig. 2. Comparison of `OutPair` with the state-of-the-art schemes

As one can see from the comparisons, we achieve the $1-$checkability property by reducing the client's workload in the online phase at the expense of adding small compuational overhead to the offline phase.

5 Conclusion

In this paper, we propose a highly efficient outsourcing algorithm for bilinear maps with $1-$checkability property. Our $1-$checkable outsourcing mechanism basically ensures that external servers never can cheat the honest clients. Compared to the state-of-the-art, our mechanism is approximately twice more efficient requiring only 4 elliptic curve point additions in the preimage of the bilinear map and 6 field multiplications in the image of the bilinear map. Furthermore, our mechanism is almost as efficient as existing $1/2-$checkable solutions. In particular, our mechanism only needs 3 more multiplications in the image of the

bilinear map. For future work, it is interesting and hihgly desirable to propose such efficient algorithms in the presence of only one untrusted server, i.e. in the One-Trusted Program (OUP) assumption.

References

1. Arabacı, ., Kiraz, M.S., Sertkaya, I., Uzunkol, O.: More efficient secure outsourcing methods for bilinear maps. Cryptology ePrint Archive, Report 2015/960 (2015), http://eprint.iacr.org/2015/960
2. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. Cryptology ePrint Archive, Report 2017/334 (2017), http://eprint.iacr.org/2017/334
3. Barreto, P., Galbraith, S., higeartaigh, C., Scott, M.: Efficient pairing computation on supersingular abelian varieties. Designs, Codes and Cryptography 42(3), 239–271 (2007), http://dx.doi.org/10.1007/s10623-006-9033-6
4. Beuchat, J.L., Gonzlez-Daz, J., Mitsunari, S., Okamoto, E., Rodrguez-Henrquez, F., Teruya, T.: High-speed software implementation of the optimal ate pairing over barretonaehrig curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing-Based Cryptography - Pairing 2010, Lecture Notes in Computer Science, vol. 6487, pp. 21–39. Springer Berlin Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-17455-1_2
5. Blake, I., Seroussi, G., Smart, N.: Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series). Cambridge University Press, New York, NY, USA (2005)
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) Advances in Cryptology CRYPTO 2001, Lecture Notes in Computer Science, vol. 2139, pp. 213–229. Springer Berlin Heidelberg (2001), http://dx.doi.org/10.1007/3-540-44647-8_13
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. Journal of Cryptology 17(4), 297–319 (2004), http://dx.doi.org/10.1007/s00145-004-0314-9
8. Canard, S., Devigne, J., Sanders, O.: Delegating a pairing can be both secure and efficient. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) Applied Cryptography and Network Security, Lecture Notes in Computer Science, vol. 8479, pp. 549–565. Springer International Publishing (2014), http://dx.doi.org/10.1007/978-3-319-07536-5_32
9. Chen, X.: Introduction to secure outsourcing computation. Synthesis Lectures on Information Security, Privacy, & Trust 8(2), 1–93 (2016)
10. Chen, X., Susilo, W., Li, J., Wong, D., Ma, J., Tang, S., Tang, Q.: Efficient algorithms for secure outsourcing of bilinear pairings. Theor. Comput. Sci. 562, 112–121 (2015), http://dblp.uni-trier.de/db/journals/tcs/tcs562.html#ChenSLWMTT15
11. Chevallier-Mames, B., Coron, J.S., McCullagh, N., Naccache, D., Scott, M.: Secure delegation of elliptic-curve pairing. Cryptology ePrint Archive, Report 2005/150 (2005)
12. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. 156(16), 3113–3121 (2008)
13. Hess, F., Smart, N., Vercauteren, F.: The eta pairing revisited. Information Theory, IEEE Transactions on 52(10), 4595–4602 (Oct 2006)

14. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3378, pp. 264–282. Springer (2005), http://www.iacr.org/cryptodb/archive/2005/TCC/3678/3678.pdf
15. Joux, A.: A one round protocol for tripartite diffiehellman. Journal of Cryptology 17(4), 263–276 (2004), http://dx.doi.org/10.1007/s00145-004-0312-y
16. Kang, B.G., Lee, M.S., Park, J.H.: Efficient delegation of pairing computation (2005)
17. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N. (ed.) Cryptography and Coding, Lecture Notes in Computer Science, vol. 3796, pp. 13–36. Springer Berlin Heidelberg (2005), http://dx.doi.org/10.1007/11586821_2
18. Kumar, K., Liu, J., Lu, Y.H., Bhargava, B.: A survey of computation offloading for mobile systems. Mobile Networks and Applications 18(1), 129–140 (Feb 2013), http://dx.doi.org/10.1007/s11036-012-0368-0
19. Lin, X.J., Qu, H., Zhang, X.: New efficient and flexible algorithms for secure outsourcing of bilinear pairings. Cryptology ePrint Archive, Report 2016/076 (2016), http://eprint.iacr.org/2016/076
20. Luo, Y., Fu, S., Huang, K., Wang, D., Xu, M.: Securely outsourcing of bilinear pairings with untrusted servers for cloud storage. In: Trustcom/BigDataSE/ISPA (2016)
21. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. NIST Special Publication 800-145 (2011)
22. Nguyen, P.Q., Shparlinski, I.E., Stern, J.: Distribution of modular sums and the security of the server aided exponentiation (2000)
23. Ren, Y., Ding, N., Wang, T., Lu, H., Gu, D.: New algorithms for verifiable outsourcing of bilinear pairings. Science China Information Sciences 59(9), 99103 (2016)
24. Shacham, H.: New Paradigms in Signature Schemes. Ph.D. thesis, Stanford, CA, USA (2006)
25. Tian, H., Zhang, F., Ren, K.: Secure bilinear pairing outsourcing made more efficient and flexible. In: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security. pp. 417–426. ASIA CCS '15, ACM, New York, NY, USA (2015), http://doi.acm.org/10.1145/2714576.2714615
26. Wang, Y., Wu, Q., Wong, D.S., Qin, B., Chow, S.S.M., Liu, Z., Tan, X.: Securely Outsourcing Exponentiations with Single Untrusted Program for Cloud Storage, pp. 326–343. Springer International Publishing, Cham (2014), http://dx.doi.org/10.1007/978-3-319-11203-9_19