

# Secure computation outsourcing

MOEZ BEN MBARKA

Cryptolog, Paris - CNRS Labri Lab, Bordeaux  
benmbark@labri.fr

## Abstract

Computation outsourcing is used when an entity needs to execute a task but does not have the appropriate computation power to perform it. In this paper, we will survey the approach of delegation: instead of doing the computation locally, the entity outsources it to one or more external helpers which have the appropriate resources. Although this can safely be done for most everyday life tasks, it is a more complicated question when dealing with computations which involve security requirements. Indeed, the entity owner may wish that the computation input to be kept secret from the helper. Furthermore, the involved helpers may be dishonest or corrupted and thus the task owner have to check that the returned result is correct. We will first survey some research works related to secure computation outsourcing. Then we will show some outsourcing models and related issues.

**Keywords:** secure outsourcing, cryptographic processors.

## 1 Introduction

Computation outsourcing is used when an entity (the client) needs to execute a task but does not have the appropriate computation power to perform it. The task is then delegated to external helpers (the servers) which have the sufficient calculation power. Secure outsourcing refers to an outsourcing in which security requirements are involved. The main issue of a secure outsourcing is to delegate a computation to a set of untrusted helpers without revealing neither the input nor the output.

The key idea is for the client to do some pre-processing over the input before sending it to the helpers, and then to do some post-processing over the helpers result to recover the final computation output. There are two main pre-processing methods to hide the secret input: disguise and encryption. The disguise is to apply some functional or mathematical transformations over the input. The encryption is a particular disguise method which consists in encrypting the input using a secret key. The main difference between the two methods, is that the disguise method can be broken provided the helpers know some information about the disguise operations, while, the encryption can not be broken even if the encryption algorithm is known (provided that the encryption key is kept secret).

The next section will survey main secure outsourcing research works. In the section 3, we will propose an outsourcing model and try to formalize the secure outsourcing requirements.

## 2 Related work

Many works have been done in the framework of secure outsourcing in the scope of many research areas.

In [19] [14] [27] [21] [15] [11] [23] [20] [26] [9], issues related to outsource the modular exponentiation are evoked. The main objective is to define server-aided protocols for public key algorithms, allowing for example fast generation of RSA signatures into low cost cryptographic processors like smart cards. Several protocols have been proposed to solve this problem, and many have been broken. In [26], the authors proposed a server-aided RSA computation protocol which has been broken using passive and active attacks in [23] [31] [28].

In [9], the authors proposed a protocol to outsource the RSA key generation. The protocol was then improved in [13] to support collusion attacks (namely the attacks in which the servers collude to share information about the computation). However, the protocol has been broken in [12].

In [1], the problem of secure outsourcing is addressed by encrypting the computation input. Basically, the client computes  $y = E(x)$  where  $x$  is the input to hide and  $E$  is an encryption function, then gives  $y$  to the untrusted server. The server performs the computation and returns  $y' = F(y)$  to the client. Then, the owner recovers the final output by computing  $F(x) = D(y')$ . The author proposed encryption schemes for some well-known functions (like the discrete logarithm problem). However, he didn't address the problem of the correctness since he assumed that the server always perform the computation he is expected to perform.

In [7], Blaze proposed a protocol to outsource secret-key encryption and decryption. Luck found a serious weakness in this protocol and specified a new one in [25]. This new protocol was later broken by Blaze in [8]. In the proposed protocols, the server knows for each encryption request both the plain-text and the cipher-text but only the tamper-resistant device is trusted with the key. The authors didn't address the problem of hiding the input or the output from the server.

Solutions to securely outsource scientific computations (matrix multiplications, differential equations, String matching...) have been discussed in [2] [3] [5] [6]. Most of these solutions are based on mathematical disguise methods in the client side. Many disguise techniques suitable for scientific computations have been proposed by Mikhail in [5].

The papers [29] [4] [24] have addressed many issues related to the security of mobile code. This class of problems is similar to the problem of secure outsourcing since the mobile code has to be protected against cheating and potentially curious and/or malicious hosts. Additionally, the code owner (called the originator) may want that the computation input and output have to be hidden from the host. Some research works have attempted to provide generic solutions using function hiding techniques. The aim of function hiding is to provide confidentiality and integrity to the functions being executed under an untrusted host. "Function confidentiality" means that the host does not learn anything about the function. "Function integrity" means that the code owner can check the correctness of the result. [24] proposed an efficient solution to hide any function that can be represented as a Boolean circuit. Other hiding schemes have been proposed in [29] and [30]. For usual secure outsourcing use-cases, "Function confidentiality" is not needed since the untrusted helper may know which function to execute.

On the other hand, mobile code may need more security requirements. The host manager may want to prevent the execution of malicious mobile code by checking the code integrity to prove that it comes from a registered originator. This issue has also been widely addressed through containment mechanisms like sandbox and applet-firewall.

In [22] and [17], another related problem is addressed. The objective is to overcome the current memory limitations of trusted handheld devices. The main issue is to provide the same level of security to the externally stored data as if it was protected inside the trusted environment. The key idea is to encrypt the data using a secret key stored into the trusted device. Then the data can be stored in an external memory resource. The trusted device simply acts as a key to access the

data providing data privacy.

### 3 An outsourcing model

In this section we will try to formalize a simple outsourcing scheme. Lets say we have a tamper-resistant trusted device  $A$  storing a secret key  $S$  owned by a user  $U$ .  $U$  wants to execute an algorithm  $F$  using this key. Since,  $A$  does not have an appropriate computation power to execute  $F$ , we need to outsource the most part of the computation to an external helper  $B$  which have sufficient computation resources. If  $B$  is trusted, then  $A$  can simply pass  $S$  and the input of the computation to  $B$ . However, we are going to consider  $B$  as an untrusted and potentially malicious helper.

First lets introduce some notions [19] [18]. A communication channel is said private if the content of the communication can not be observed by anyone who is neither the sender nor the receiver. The channel is trusted, if the receiver can have confidence on the origin of the received data. If a channel is both private and trusted, we will say it is secure. If it is neither trusted nor private, then we say it is unsecure. In our model, we are going to suppose that the communication channel between  $A$  and  $B$  is unsecure.

In addition to  $S$ ,  $F$  may need an external input  $X$  which will be given to  $A$  by the  $A$ 's owner through a secure channel. An input/output can have one of the following properties:

- **secret:** if it is only available to  $A$  (e.g : the secret key).
- **protected:** if it is only available to  $A$  and  $A$ 's owner.
- **unprotected:** is available to  $A$ ,  $A$ 's owner and  $B$ .

We will denote the input as  $X = \{X_s, X_p, X_u\}$  where  $X_s = S$  denotes the secret input and  $X_p, X_u$  denote respectively the protected and unprotected input. Similarly, we will denote:  $Y = \{Y_p, Y_u\}$  the computation output such as  $Y_p$  and  $Y_u$  represent respectively the protected and the unprotected parts.

The figure 1 shows a typical outsourcing scheme:

1.  $A$  performs a pre-processing function  $A_{pre}$  over  $S$  and  $X_p$ .
2.  $B$  executes the outsourced part  $B_F$  of the computation and returns the outputs  $Y_u$  and  $Y'_p$  to  $A$ .
3.  $A$  performs a post-processing function over  $Y'_p$  to get  $Y_p$ .
4.  $A$  computes a validation boolean value  $V(Y)$  to check the correctness of the output.

We will denote  $O_{A,B}(F)$  a protocol to outsource a computation  $F$  from  $A$  to  $B$ . A typical secure outsourcing scheme should have the following properties:

**Property 1 Security:**  $O_{A,B}(F)$  is secure, if  $B$  does not learn anything about  $S$  or  $X_p$  during the actual computation.

Formally:  $O_{A,B}(F)$  is secure, if computing  $S$  and  $X_p$  using  $A_{pre}(S, X_p) = X'_p$  is unfeasible.

The definition of “unfeasible” depends on the wished security degree of the scheme. Some outsourcing protocols cited in the section 2 define it as the fact that  $B$  can not deduce  $S$  or  $X_p$  from  $X'_p$  in a polynomial time.

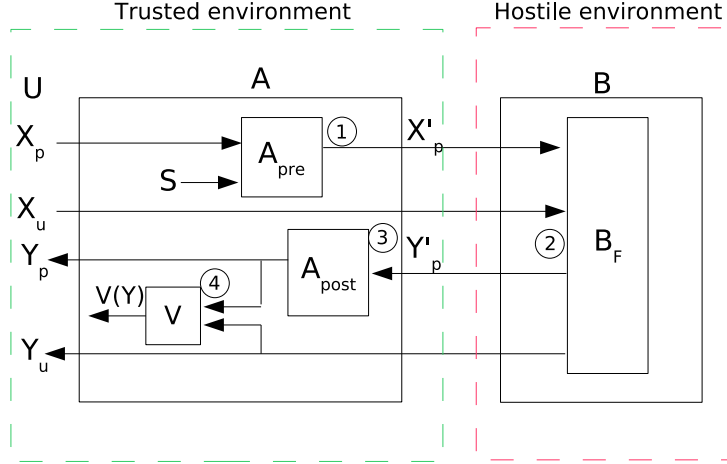


Figure 1: A typical outsourcing scheme.

**Property 2 Correctness:**  $O_{A,B}(F)$  is correct, if  $A$  can efficiently check that the output returned by  $B$  is actually the correct output.

Formally:  $O_{A,B}(F)$  is correct, if there is a polynomial time function  $V(Y)$  that returns true if the output verifies  $Y = F(X)$ , false otherwise.

Some of the solutions cited in section 2 define the notion of probabilistic correctness. It means that,  $A$  is only able to detect  $B$  failures with a probability  $P$ .

**Property 3 Efficiency:**  $O_{A,B}(F)$  is efficient, if the time taken by :  $A$  pre- and post-processing,  $A$ - $B$  communication latencies and  $B$  computations is less than the time that would be needed by  $A$  to perform the full computation.

Formally,  $O_{A,B}(F)$  is efficient if  $T_A(A_{pre}) + T_B(F) + T_A(A_{post}) + T_A(V) + T_{A,B} < T_A(F)$ . Where  $T_A(F)$  denotes the execution time of a function  $F$  within  $A$  and  $T_{A,B}$  denotes the communication latencies between  $A$  and  $B$ .

### 3.1 Possible settings

In the model shown in the figure 1, we assumed that the user  $U$  have a secure input/output channel with the trusted device  $A$ . In usual cases,  $A$  may be a HSM\* connected to a server  $B$  which has to route all  $U$  requests to  $A$ . In the figure 2,  $U$  can only send requests to  $A$  through an unsecure communication channel with  $B$ .

If  $U$  is a human user, then obviously he can not provide any protected input to  $A$ , since any disguise method which could not be broken by  $B$  is out of the ability of a normal human being. Moreover,  $U$  will not be able to check the correctness of the computation output and thus he can completely be duped by  $B$ .

However, if  $U$  is a machine, then the model can be adapted to provide the three basic outsourcing properties: security, correctness and efficiency. The adaptation consists in that  $U$  will perform the pre- and post-processing over the protected input/output and will also compute the function  $V(Y)$  to check the correctness of the result.  $A$  will only need to disguise the secret key to hide it from  $B$ .

\*HSM: Hardware Security Module

The model can be simplified (figure 3), if we assume that  $U$  does not need to hide any input or output from  $B$ . Thus,  $U$  will not perform any disguise method which may lead to more efficiency. However, in this case, the user is more vulnerable to attacks based on the knowledge of the input and the corresponding output. Moreover, if an attacker can control  $B$  (can see any computation input and output) and at the same time can behave as a user  $U$  (we assumed that  $U/B$  channel is not secure), then he can carry out attacks based on chosen input. For example, if  $F$  is a secret key encryption, an attacker can carry out chosen plain-text attacks to compromise the secret key [16] [10].

The model can still be simplified by removing the need to check the correctness of the output. However, to keep the security and correctness properties, we need to make more security assumptions about  $B$ . First,  $U$  must trust the code being executed by  $B$ . It is the case for example, if  $U$  is also the code owner. Thus,  $U$  can trust that  $B$  will not maliciously return wrong results. On the other hand, the communication channel between  $B$  and  $U$  should be at least trusted. It means that  $U$  should be sure that he is talking with the code he trust within  $B$ . Otherwise, he can be duped by a malicious code which tries to behave as the trusted code.

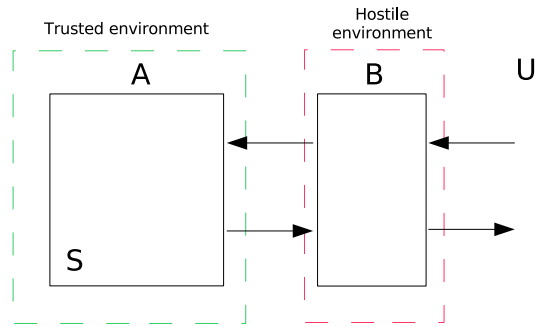


Figure 2: An outsourcing scheme where  $U$  can only communicate with  $B$ .

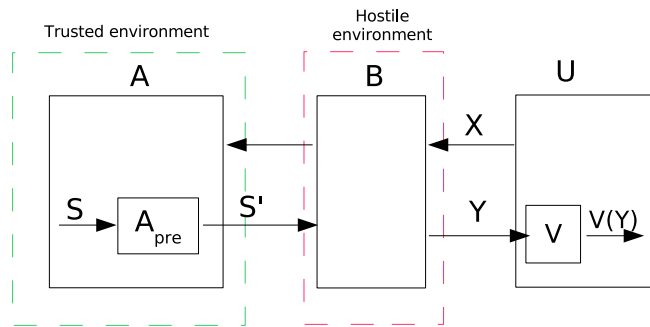


Figure 3: A simplified outsourcing scheme.

## 4 Conclusion

As described in the previous section, the outsourcing scheme relies heavily on the trust parameters. More the server is trusted, less the client have computations to perform and more the scheme is

efficient. Furthermore, the outsourcing scheme depends on whether the user can directly interact with the tamper-resistant device or whether all user requests are routed through the untrusted server.

Although many research works have been done in the scope of secure outsourcing, it seems that no general and generic solution exists. All of the proposed solutions (section 2) are only appropriate for specific situations. One of the main issues that have to be evoked if such generic solution exists is how to formally prove that the scheme verifies the three secure outsourcing properties (security, correctness and efficiency) for any situation.

## References

- [1] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 195–203. ACM, 1987.
- [2] R. Akimana, O. Markowitch, and Y. Roggeman. Grids confidential outsourcing of string matching. The 6th WSEAS Int. Conf. on Software Engineering, Parallel and Distributed Systems, 2007.
- [3] R. Akimana, O. Markowitch, and Y. Roggeman. Secure outsourcing of dna sequences comparisons in a grid environment.
- [4] J. Algesheimer, C. Cachin, J. Camenisch, and G. Karjoth. Cryptographic security for mobile code. In *SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy*, page 2, Washington, DC, USA, 2001. IEEE Computer Society.
- [5] Atallah, Pantazopoulos, Rice, and Spafford. Secure outsourcing of scientific computations. In *Advances in Computers*, ed. by Marshall C. Yovits, Academic Press, volume 54. 2001.
- [6] M. J. Atallah and J. Li. Secure outsourcing of sequence comparisons. *Int. J. Inf. Secur.*, 4(4):277–287, 2005.
- [7] M. Blaze. High-bandwidth encryption with low-bandwidth smartcards. In *Proceedings of the Third International Workshop on Fast Software Encryption*, pages 33–40. Springer-Verlag, 1996.
- [8] M. Blaze, J. Feigenbaum, and M. Naor. A formal treatment of remotely keyed encryption. In *SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 868–869, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics.
- [9] D. Boneh, N. Modadugu, and M. Kim. Generating rsa keys on a handheld using an untrusted server. In *INDOCRYPT '00: Proceedings of the First International Conference on Progress in Cryptology*, pages 271–282, London, UK, 2000. Springer-Verlag.
- [10] E. Brickel and A. Odlyzko. Cryptanalysis: A survey of recent results.
- [11] J. Burns and C. J. Mitchell. Parameter selection for server-aided rsa computation schemes. *IEEE Trans. Comput.*, 43(2):163–174, 1994.
- [12] T. Cao, X. Mao, and D. Lin. Security analysis of a server-aided rsa key generation protocol. In K. Chen, R. H. Deng, X. Lai, and J. Zhou, editors, *ISPEC*, volume 3903 of *Lecture Notes in Computer Science*, pages 314–320. Springer, 2006.

- [13] Y. Chen, R. Safavi-Naini, J. Baek, and X. Chen. Server-aided rsa key generation against collusion attack. In M. Burmester and A. Yasinsac, editors, *MADNES*, volume 4074 of *Lecture Notes in Computer Science*, pages 27–37. Springer, 2005.
- [14] M. Dijk, D. Clarke, B. Gassend, G. E. Suh, and S. Devadas. Speeding up exponentiation using an untrusted computational resource. *Des. Codes Cryptography*, 39(2):253–273, 2006.
- [15] A.-M. Ernvall and K. Nyberg. On server-aided computation for RSA protocols with private key splitting. In S. Knapskog, editor, *Proceedings of Nordsec 2003*. Department of Telematics, NTNU, 2003.
- [16] F. X. S. et al. Cryptanalysis of block ciphers: A survey.
- [17] P. B. Gemplus. How smartcards can benefit from internet technologies to break memory constraints.
- [18] H. Gobiuff, S. Smith, J. D. Tygar, and B. Yee. Smart cards in hostile environments. In *WOEC'96: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 3–3. USENIX Association, 1996.
- [19] S. Hohenberger and A. Lysyanskaya. *How to securely outsource cryptographic computations*, volume 3378/2005 of *Lecture Notes in Computer Science*, pages 264–282. Springer Berlin / Heidelberg, 2005.
- [20] S.-M. Hong, J.-B. Shin, H. Lee-Kwang, and H. Yoon. A new approach to server-aided secret computation. In *Information Security and Cryptology*, pages 33–45, 1998.
- [21] M. Jakobsson and S. Wetzel. Secure server-aided signature generation. In *PKC '01: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, pages 383–401. Springer-Verlag, 2001.
- [22] V. Jean-Jacques. *Projet OSMOSE: modélisation et implémentation pour l'interopérabilité de services carte à microprocesseur par l'approche orientée objet*. PhD thesis, Université des Sciences et Technologies de Lille, 1997.
- [23] C. H. Lim and P. J. Lee. Security and performance of server-aided rsa computation protocols. In *CRYPTO '95: Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, pages 70–83. Springer-Verlag, 1995.
- [24] S. Loureiro, L. Bussard, and Y. Roudier. Extending tamper-proof hardware security to untrusted execution environments. In *CARDIS'02: Proceedings of the 5th conference on Smart Card Research and Advanced Application Conference*, pages 12–12, Berkeley, CA, USA, 2002. USENIX Association.
- [25] S. Lucks. On the security of remotely keyed encryption. In *FSE '97: Proceedings of the 4th International Workshop on Fast Software Encryption*, pages 219–229. Springer-Verlag, 1997.
- [26] T. Matsumoto, H. Imai, C.-S. Lai, and S.-M. Yen. On verifiable implicit asking protocols for rsa computation. In *ASIACRYPT '92: Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques*, pages 296–307. Springer-Verlag, 1993.
- [27] T. Matsumoto, K. Kato, and H. Imai. Speeding up secret computations with insecure auxiliary devices. In *CRYPTO '88: Proceedings on Advances in cryptology*, pages 497–506. Springer-Verlag, 1990.

- [28] B. Pfitzmann and M. Waidner. Attacks on protocols for server-aided RSA computation. *Lecture Notes in Computer Science*, 658:153–162, 1993.
- [29] T. Sander and C. Tschudin. Towards mobile cryptography. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1998. IEEE Computer Society Press.
- [30] T. Sander and C. F. Tschudin. On software protection via function hiding. In *Proceedings of the Second International Workshop on Information Hiding*, pages 111–123, London, UK, 1998. Springer-Verlag.
- [31] Shimbo and Kawamura. Factorization attack on certain server-aided secret computations. *Electronics Letters*, 26(17):1387–1388, 1990.