Publicly Auditable Inner Product Evaluation from Different Data Sources under On-time generated Multiple Keys

[1]Suganya.S, [2]Dr. D. Sharmila
ajitharajanjan18@gmail.com
III MCA, Lord Jegannath College of Engineering & Technology
Assistant Professor, Department of Computer Applications,
Lord Jegannath College of Engineering & Technology
shamishenai@rediffmail.com

Abstract: Uploading data streams to a resource-rich Moreover, the experimental result also shows cloud server for inner product estimation, an necessary building block in many popular stream applications (e.g., statistical monitoring), is appealing to many companies and individuals. On the other hand, verifying the result of the remote computation plays a crucial role in addressing the issue of trust. My project is meeting point on the multi-key scenario where data streams are uploaded by multiple data sources with distinct keys. We first present a novel homomorphism verifiable tag technique to publicly verify the outsourced inner product computation on the dynamic data streams, and then extend it to support the verification of matrix product computation.

INTRODUCTION

The rise of the cloud compute paradigm requires that users can securely outsource their data to a remote service dealer while allowing it to consistently perform computations over the data. The recent ground-breaking improvement of fully homomorphism encryption [1] allows us to maintain confidentiality/privacy of outsourced data in this setting. I am look at the analogous but orthogonal question of given that integrity/authenticity for computations over outsourced data. Toward this goal, we define and instantiate a new primitive, called a fully homomorphism message authenticator. This primitive can be seen as a symmetric-key version of fully homomorphism signatures, which were denied by Bone and Freeman [2], but whose construction remains an open problem. We will return to survey the related work on partially homomorphism signatures and authenticators, as well as related work on delegating memory and computation.

Even clients with very limited storage capacity (e.g., smart phones) can have access on demand" to very large amounts of data. Having access to the outsourced data does not necessarily mean only to retrieve such data. Indeed, a user may wish to perform a computation on (a subset of) the outsourced data and this too can be delegated to the service provider. These and other bent it's the key success of Cloud Computing. The paradigm, however, raises security concerns essentially because cloud providers cannot always be trusted. One problem is related to preserving the privacy of the outsourced data.

REALATED WORK

The problem of *verifying the outsourced algebraic computation* has attracted extensive attention in the past few years. These schemes can be divided into two categories:

- Single key setting
- Multi key setting

Single-key Setting. Fully homomorphic message authenticators [3], [4], [5] allow the holder of a public evaluation key to perform computations on previously authenticated data, in such a way that the produced proof can be used to certify the correctness of the computation. More precisely, with the knowledge of the secret key used to authenticate the original data, a client can verify the computation by checking the proof. For the asymmetric setting, Bone and Freeman [6] proposed a realization of homomorphic signatures for bounded constant degree polynomials based on hard problems on ideal lattices. Although not all the above schemes are explicitly presented in the context of streaming data, they can be applied there under a *single-key setting*. In this scenario, the data source continually generates and outsources authenticated data values to a third-party server. Given the public key, the server can compute over these data and produce a proof, which enables the client to privately [3], [4], [6] or publicly [8] verify the computation result.

Multi-key Setting. Recently, a multi-key non-interactive confirmable computation method was future in [7], followed by a stronger security guarantee scheme [8]. However, these schemes may not be useful to the stream setting since sources lost data control after the outsourcing and thus cannot produce the matching secrets for the confirmation. Besides, both of them based on FHE are not practically efficient. As shown in [9], it takes at least 30 seconds to run one bootstrapping operation of FHE for weaker security parameter on a high performance machine. I am considering publicly provable designation of inner product computation over dynamic data streams under the *multi-key* setting. The proposed scheme is extremely lightweight for both data sources and clients.

ALGORITHM FORMULATION

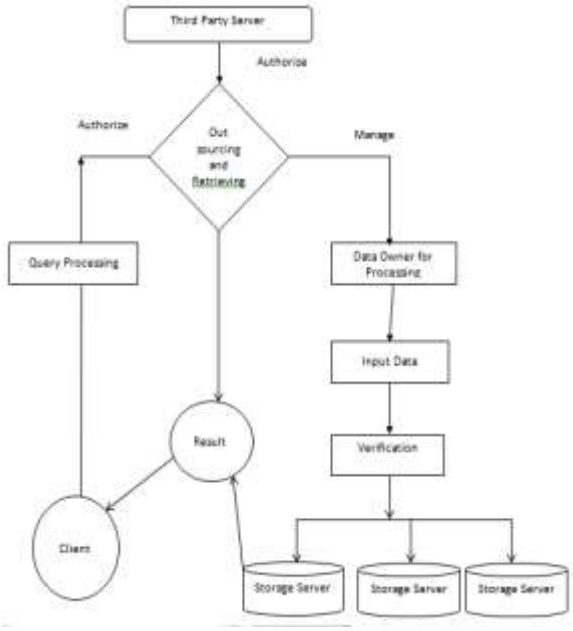
In this subsection, we provide the formal algorithm definition of our proposed scheme.

Our public verifiable inner product computation scheme includes a tuple of algorithms as follows:

• KeyGen(1_) → (pkj , skj): A probabilistic algorithm run by each machine Mj takes a security parameter _ as input, and outputs a public key pkj and a secret key skj .

• TagGen (ski, i,Xj,i) → _j,i: A (possibly)probabilistic algorithm run by machine Mj,

ARCHITECTURE DESIGN



Takes as input its secret key skj , the current discrete time i and data Xj,i, and outputs a publicly verifiable tag _j,i.

• Evaluate(FIP,Xi,Xj) → res: Let Xi = {Xi,1,Xi,2, ...,Xi,n} and Xj = {Xj,1,Xj,2, ...,Xj,n}z

KeyGen (1_) :

1. **For**j = 1 to l **do**
2. Choose a random number skj = sj∈ Z_ q as
the secret key
3. Compute pkj = gsj
4. output (pkj , skj)
5. **end for**

TagGen(skj , i,Xj,i) :
1. compute _j,i = (gh1(Mj ,i)
1 gh2(Mj ,i)
2 g
Xj,i
3)skj
2. output _j,i

Evaluate(FGS,Xj) :
1. compute res = Pi2_ Xj,i
2. output res

GenProof (FGS, _j ,Xj):
1. compute _ = Qi2_ _j,i
2. output _

CheckProof (FGS, pkj , res, _) :
1. set S_ = (S1, S2)
2. compute S1 = Pi2_ h1(Mj, i) and S2 = Pi2_ h2(Mj , i)
3. **if**(e(_, g) = e(gS1
1 gS2
2 gres
3 , pkj)) **then**
4. output 1
5. **else**
6. output 0
7. end if

MATRIX PRODUCT QUERY EXTENSION

In this section, we extend the publicly verifiable inner product evaluation scheme to support *matrix product query* under the multi-key setting.

SECURITY ANALYSIS

We prove the security of the proposed scheme in the random oracle model.

Proof: The security definition of the publicly verifiable computation scheme for group-by sum query is similar to definition except that adversary AC forges a result res 6= Pi2_ Xj,i and passes the verification C. Now, we show how to construct an adversary
B that uses A to solve the CDH problem. That is, C given a CDH tuple (g, gsj , g3), the adversary B is able C to compute gsjC3 with non-negligible probability. CB simulates a publicly verifiable computation C scheme with group-by sum query for A as follows.

Setup: The adversary B sets machine Mj 's public key pkj = gsj , g1 = g · g_C3 and g2 = (g · g3)_,C where _ and _ are two random numbers in Z_ q . The system parameters and the public key are given to the adversary A.

Query: The adversary A adaptively queries B for tags on the discrete time and data of its choice.
Specifically, A sends a tuple (Mj,Xj,1, 1) to B. The algorithm B generates a tag _j,1 and sends it back to A.

EVALUATION

This section evaluates the practical performance of our scheme. We conduct the computation at client side by using JPBC library [10] in Eclipse 4.2 on a Windows 7 machine with 2.30 GHz Intel Core I7-3615QM. The cloud-side computation overhead is evaluated on an IBM System x3550 M4 machine. We choose type-A (symmetric) pairings with 80-bit security in our simulation, which results in the element in G1 and Z_ q to be 512-bit and 160-bit, respectively. Note that our scheme can also be implemented under the asymmetric pairings.

1. Storage:

In our scheme, data sources store their public/private keys and system parameters locally while outsourcing all the data along with the corresponding tags to a third-party server. The size of a public and private key pair (pkj ∈ G1, skj∈ Z_ q) is 84 bytes. The size of system parameters {G, g, g1, g2, h1(), h2()} is constant, regardless of data streams' size. The public keys of data sources dominate the client's storage. Assuming that there are 100 data sources in

the system, the total storage on the client side is 6400 bytes. Thus, we observe that the storage overhead on data owners and clients are much smaller than the outsourced data streams.

2. Communication:

We do not take the communication cost of query and the computation result into account, since they also occurs in the scenario without outsourcing. On receiving a computation query from the client, the cloud evaluates the corresponding function and generates a proof to ensure the validity of the computation result. Thus, the communication cost is constant in our scheme, regardless of the input size of the evaluated function.

3. Computation:

Data source side. Generating a tag for a data value needs three exponentiation operations in G1, two modular multiplications in G1 and two hashes, which takes about 2.25 ms.

Client side. The verification cost for group-by sum and inner product queries, respectively. Note that the auxiliary information S_ in the verification can be pre-computed, because they are only determined by S_, i.e., independent of the outsourced data.

CONCLUSION

In this paper, launch a novel homomorphism verifiable tag technique, and drawing an efficient and publicly provable inner product calculation scheme on the dynamic outsourced data streams under multiple keys. I am also extended the inner product scheme to support matrix product. Compared with the existing works under the single-key setting, our scheme aims at the more demanding *multi-key* scenario, i.e., it allows several data sources with dissimilar secret keys to upload their *nonstop data streams* and entrust the same computation to a third party server, while the traceability can still be offer on demand.

Furthermore, any *keyless* client is able to *widely* verify the validity of the return calculation result. Protection study shows that our scheme is verifiable secure under the CDH assumption in the arbitrary oracle model. Experimental results display that our protocol is practically well-organized in terms of both communication and calculation cost.

REFERENCE

1. C. Gentry. Fully homomorphic encryption using ideal lattices. In M. Mitzenmacher, editor, 41st ACM STOC, pages 169{178. ACM Press, May / June 2009.
2. D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In K. G. Paterson, editor, EUROCRYPT 2011, volume 6632 of LNCS, pages 149{168. Springer, May 2011.
3. D. Catalano and D. Fiore, "Practical homomorphic macs for arithmetic circuits," in *Advances in Cryptology–EUROCRYPT*. Springer, 2013, pp. 336–352.
4. R. Gennaro and D. Wichs, "Fully homomorphic message authenticators," in *Advances in Cryptology-ASIACRYPT*. Springer, 2013, pp. 301–320.
5. M. Bakes, D. Fiore, and R. M. Reischuk, "Verifiable delegation of computation on outsourced data," in *ACM conference on Computer and communications security*. ACM, 2013, pp. 863–874.
6. D. Boneh and D. M. Freeman, "Homomorphic signatures for polynomial functions," in *Advances in Cryptology–EUROCRYPT*. Springer, 2011, pp. 149–168.
7. S. G. Choi, J. Katz, R. Kumaresan, and C. Cid, "Multi-client non-interactive verifiable computation," in *Theory of Cryptography*. Springer, 2013, pp. 499–518.
8. S. D. Gordon, J. Katz, F.-H. Liu, E. Shi, and H.-S. Zhou, "Multi-client verifiable computation with stronger security guarantees," in *Theory of Cryptography*. Springer, 2015, pp. 144–168.
9. C. Gentry and S. Halevi, "Implementing gentry's fully homomorphic encryption scheme," in *Advances in Cryptology–EUROCRYPT*. Springer, 2011, pp. 129–148.
10. A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *IEEE Symposium on Computers and Communications*. IEEE, 2011, pp. 850–855.