A Note on "Outsourcing Large Matrix Inversion Computation to a Public Cloud"

Zhengjun Cao and Lihua Liu

Abstract. We remark that the Lei et al.'s scheme [IEEE Transactions on Cloud Computing, 1 (1), 78-87, 2013] fails, because the verifying equation does not hold over the infinite field $\mathbb{R}$. For the field $\mathbb{R}$, the computational errors should be considered seriously.

Keywords. Cloud computing, outsourcing computation, matrix inversion, malicious server.

1 Introduction

Recently, Lei et al. [1] have proposed a scheme for outsourcing matrix inversion computation over infinite field $\mathbb{R}$. In this note, we remark that the Lei et al.'s scheme fails because the verifying equation does only hold over any finite field, instead of the infinite field $\mathbb{R}$. For the field $\mathbb{R}$, the computational errors, especially rounding errors, should be considered carefully.

2 Computational errors

Suppose that a floating-point number system is characterized by four integers [2]: base χ, precision p, exponent range $[L, U]$. Then its accuracy can be characterized by a quantity known as *machine precision*, ϵ. If a given real number x is not exactly representable as a floating-point number, then it must be approximated by some "nearby" floating-point number. The process of choosing $\mathrm{fl}(x)$ to approximate x is called rounding, and the error introduced by such an approximation is called rounding error. Consider the simple computation $x(y+z)$. In floating-point arithmetic we have $\mathrm{fl}(y+z) = (y+z)(1+\theta_1)$ with $|\theta_1| \le \epsilon$, so that

$$\begin{aligned}\mathrm{fl}(x(y+z)) &= (x((y+z)(1+\theta_1)))(1+\theta_2), \text{ with } |\theta_2| \le \epsilon \\ &= x(y+z)(1+\theta_1+\theta_2+\theta_1\theta_2) \\ &= x(y+z)(1+\theta), \text{ with } |\theta| = |\theta_1+\theta_2| \le 2\epsilon.\end{aligned}$$

• Z. Cao is with the Department of Mathematics, Shanghai University, Shanghai, China.

• L. liu is with the Department of Mathematics, Shanghai Maritime University, China. liulh@shmtu.edu.cn

3 Review of the Lei et al.'s scheme

Let $\mathbf{X}(i,j), x_{i,j}$ or x_{ij} denote the entry in ith row and jth column in matrix $\mathbf{X}$. Define $\delta_{x,y} = 1$, if $x = y$; $\delta_{x,y} = 0$, if $x \neq y$. Given a matrix $\mathbf{X} \in \mathbb{R}^{n\times n}$, the resource-constrained client wants to securely outsource the computation of $\mathbf{X}^{-1}$ to the cloud. The scheme can be described as follows (see Table 1).

Table 1: The Lei et al.'s scheme for outsourcing matrix inversion computation

Client	Server
Setup. Pick two sets of random numbers $\{\alpha_1, \cdots, \alpha_n\},\ \{\beta_1, \cdots, \beta_n\}$. Generate two random permutations: π_1, π_2 over $\{1, \cdots, n\}$, Set them as the secret key.	
Input $\mathbf{X} \in \mathbb{R}^{n\times n}$. *Transformation* $\mathbf{Y}(i,j) = (\alpha_i/\beta_j)\mathbf{X}(\pi_1(i), \pi_2(j))$, *Outsourcing* Send $\mathbf{Y}$ to the server.	*Compute* $\mathbf{R}' = \mathbf{Y}^{-1}$ and return it to the client.
Composition $\mathbf{R}(i,j) = \left(\alpha_{\pi_1^{-1}(j)}/\beta_{\pi_2^{-1}(i)}\right)\mathbf{R}'(\pi_2^{-1}(i), \pi_1^{-1}(j))$. *Verification* Pick an $n \times 1$ random 0/1 vector $\mathbf{r}$. Check that $\mathbf{R}(\mathbf{X}\mathbf{r}) - \mathbf{I}\mathbf{r} \overset{?}{=} (0, \cdots, 0)^T$. Repeat the process l times. *Output* $\mathbf{R}$.	

The correctness of the scheme can be argued as follows. Set the matrixes $\mathbf{P}_1, \mathbf{P}_2$ as

$$\mathbf{P}_1(i,j) = \alpha_i \delta_{\pi_1(i),j}, \quad \mathbf{P}_2(i,j) = \beta_i \delta_{\pi_2(i),j}.$$

Then

$$\mathbf{P}_1^{-1}(i,j) = (\alpha_j)^{-1}\delta_{\pi_1^{-1}(i),j}, \quad \mathbf{P}_2^{-1}(i,j) = (\beta_j)^{-1}\delta_{\pi_2^{-1}(i),j}.$$

Hence,

$$\mathbf{Y} = \mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1},\ \mathbf{R} = \mathbf{P}_2^{-1}\mathbf{R}'\mathbf{P}_1 = \mathbf{P}_2^{-1}(\mathbf{P}_1\mathbf{X}\mathbf{P}_2^{-1})^{-1}\mathbf{P}_1 = \mathbf{X}^{-1}.$$

Unfortunately, the above reasoning process is true only in some symbolic computing environments. But in a practical floating-point number system, computational errors involved in the above procedure should be considered seriously.

4 The checking mechanism in the Lei et al.'s scheme fails

Let $\mathbf{X} = (x_{ij})_{n\times n}$, $\mathbf{R} = (\lambda_{ij})_{n\times n}$, $\mathbf{r} = (r_1, \cdots, r_n)^T$. Then the first entry of $\mathbf{R}(\mathbf{X}\mathbf{r})$ is

$$\Sigma_{i=1}^{n}\lambda_{1i}(\Sigma_{j=1}^{n} x_{ij}r_j) = r_1\Sigma_{j=1}^{n} \lambda_{1j}x_{j1} + \cdots + r_n\Sigma_{j=1}^{n} \lambda_{1j}x_{jn}.$$

Since $\lambda_{ij} = \left(\alpha_{\pi_1^{-1}(j)}/\beta_{\pi_2^{-1}(i)}\right) \mathbf{R}'(\pi_2^{-1}(i), \pi_1^{-1}(j))$, we have

$$\begin{aligned}\Sigma_{i=1}^{n}\lambda_{1i}(\Sigma_{j=1}^{n} x_{ij}r_j) \;=\; & r_1\Sigma_{j=1}^{n} \left(\alpha_{\pi_1^{-1}(j)}/\beta_{\pi_2^{-1}(1)}\right) \mathbf{R}'(\pi_2^{-1}(1), \pi_1^{-1}(j))x_{j1} \\ & + \cdots \\ & + r_n\Sigma_{j=1}^{n} \left(\alpha_{\pi_1^{-1}(j)}/\beta_{\pi_2^{-1}(1)}\right) \mathbf{R}'(\pi_2^{-1}(1), \pi_1^{-1}(j))x_{jn}\end{aligned}$$

Since $\{\alpha_1, \cdots, \alpha_n\}$, $\{\beta_1, \cdots, \beta_n\}$ are randomly chosen in $\mathbb{R}$, the total rounding error in the above equation approximates to $\bar{x}_1\bar{y}_1n^2\epsilon$, where $\bar{x}_1 = \frac{1}{n}\Sigma_{j=1}^{n}x_{j1}$, $\bar{y}_1 = \frac{1}{n}\Sigma_{j=1}^{n}\mathbf{R}'(\pi_2^{-1}(1), \pi_1^{-1}(j))$, and ϵ is the machine precision. Therefore, the practical computational result is

$$\mathbf{R}(\mathbf{X}\mathbf{r}) - \mathbf{I}\mathbf{r} = (\bar{x}_1\bar{y}_1n^2\epsilon, \cdots, \bar{x}_n\bar{y}_nn^2\epsilon)^T \neq (0, \cdots, 0)^T.$$

Thus, the original checking mechanism in the Lei et al.'s scheme fails. The authors did not pay more attentions to the differences between the arithmetic over the infinite field $\mathbb{R}$ and that over any finite field.

To fix the scheme, one has to check that

$$|u_i - v_i| \leq \psi n^2\epsilon, \quad i = 1, \cdots, n$$

where $\mathbf{R}(\mathbf{X}\mathbf{r}) = (u_1, \cdots, u_n)^T$, $\mathbf{I}\mathbf{r} = (v_1, \cdots, v_n)^T$, and ψ is a fault-tolerant parameter. If all n inequalities are true, then output $\mathbf{R}$.

4.1 The revised version of Lei et al.'s scheme is insecure

Though the computational errors have been considered in the revised version of Lei et al.'s scheme, we show it is insecure because the malicious server can cheat the client to accept a wrong result. For example, to do this, the malicious server computes $\mathbf{R}' = \mathbf{Y}^{-1}$ and returns $\widehat{\mathbf{R}'}$ to the client, where

$$\widehat{\mathbf{R}'} = \mathbf{R}' + \begin{pmatrix} \rho & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix},$$

$\rho = (\chi - 1)\epsilon$, χ is the base of the underlying floating-point number system. In such case, we have

$$\mathbf{R}(\pi_2(1), \pi_1(1)) = (\alpha_1/\beta_1)\,\widehat{\mathbf{R}'}(1, 1) = (\alpha_1/\beta_1)\,\mathbf{R}'(1, 1) + (\alpha_1/\beta_1)\,\rho.$$

Hence,

$$\left|v_{\pi_1(1)} + (\alpha_1/\beta_1)\rho\sum_{j=1}^{n} r_j x_{\pi_1(1),j} - u_{\pi_1(1)}\right| \le \left|v_{\pi_1(1)} - u_{\pi_1(1)}\right| + \left|(\alpha_1/\beta_1)\rho\sum_{j=1}^{n} r_j x_{\pi_1(1),j}\right|.$$

The probability of the event that

$$\left|v_{\pi_1(1)} - u_{\pi_1(1)}\right| + \left|(\alpha_1/\beta_1)\sum_{j=1}^{n} r_j x_{\pi_1(1),j}\right| \cdot (\chi - 1)\epsilon \le \psi n^2 \epsilon,\ r_j = 1 \,\text{or}\, 0, j = 1, \cdots, n$$

is not negligible, because n is supposed to be sufficiently large. Thus, the malicious server can cheat the client to accept a wrong result $\mathbf{R}(i,j) = \left(\alpha_{\pi_1^{-1}(j)}/\beta_{\pi_2^{-1}(i)}\right)\widehat{\mathbf{R}'}(\pi_2^{-1}(i), \pi_1^{-1}(j))$.

5 Conclusion

We show that the Lei et al.'s scheme for outsourcing matrix inversion computation over the infinite field $\mathbb{R}$ is insecure. We think, the problem that what computations are worth delegating privately by individuals and companies to an untrusted cloud remains open.

References

[1] X. Lei, et al., "Outsourcing Large Matrix Inversion Computation to a Public Cloud", IEEE Transactions on Cloud Computing, 1 (1), 78-87, 2013.

[2] M. Heath, Scientific computing, an introductory survey, second edition, 2001.