JAM CM Journal of Applied Mathematics and Computational Mechanics 2016, 15(1), 39-47
www.amcm.pcz.pl p-ISSN 2299-9965
DOI: 10.17512/jamcm.2016.1.04 e-ISSN 2353-0588

SELECTED APPLICATION OF THE CHINESE REMAINDER THEOREM IN MULTIPARTY COMPUTATION

Artur Jakubski

Institute of Computer and Information Sciences, Czestochowa University of Technology
Częstochowa, Poland
artur.jakubski@icis.pcz.pl

Abstract. In this paper we present protocols checking the equality of two distributed numbers and calculation of the product in such a way that the distributed numbers are unknown to anyone. The presented protocols use the Chinese Remainder Theorem. As a result, the obtained protocols have many interesting cryptographic features.

Keywords: *multiparty cryptography, secret sharing, distributed algorithms, modular arithmetic*

1. Introduction

This paper discusses the multiparty computation protocols with the use of the Chinese Remainder Theorem. Of the many protocols with such application, we discuss two selected protocols. We adopt the honest-but-curious framework of multiparty computations, executed collectively in a fully (completely) decentralized environment of cooperating agents (participants, players, computers, servers, processes, devices, mobile devices), in which $k \geq 3$ parties collectively generate a protocol result. Because of the cryptographic application, the typical length of the participants' shares should be greater than 500 bits.

The integer is never revealed to any party. We assume that our protocol is run without any trusted party (server, dealer, or central authority).

The participants are assumed to communicate over secure channels, meaning that the messages sent from one participant to the other are private and no one can interfere along the way. We assume that all participants follow the protocol honestly.

Fully distributed or fully decentralized means: with no trusted party whatsoever and with all the parameters shared as secret.

The main idea is that both the input and output integer parameters are never revealed. Instead, they are shared by participants using an appropriate secret sharing. We are going to use the elementary additive secret sharing but also secret sharing

based on Shamir secret sharing as in [1, 2] and the BGW method [3, 4]. We use also the Chinese Remainder Theorem (see [2] and [5]). In the area of multiparty computation, five papers [6-10] deserve special attention. The papers deal with distributed primality testing.

Our paper is organized as follows. In Section 2 we discuss Shamir's secret sharing. Section 3 presents the BGW method, which is a protocol for distributed product computing. In Section 4 we introduce two new protocols, using the Chinese Remainder Theorem. Finally, Section 5 concludes the paper.

2. Shamir's secret sharing

The first threshold schemes were initiated by two papers published almost simultaneously in 1979, A. Shamir *How to share a secret* [1] and G. Blakley *Safeguarding cryptographic keys* [11].

To build a threshold scheme, Adi Shamir [1] uses a polynomial over a finite field. First, a prime number has to be selected (let's call it p), which is greater than the number of protocol parties n and greater than a password (secret, key) K. The password will be distributed among the parties. If the threshold is t, it is generated a polynomial $f(x)$ of degree $t-1$ with coefficients derived from the field $\mathbb{Z}_p$ $(GF(p))$ and free term equal K. A prime number p is public, and the coefficients of the polynomial must be secret and known only to the person choosing them, i.e. the dealer. The shares of the distribution, also known as the shadows, are the values of this polynomial for n different arguments.

Let's denote a share belonging to the party i by K_i. It is often assumed that $K_i = f(i) \bmod p$. t or more participants can reconstruct the password K. The reconstruction is to restore the polynomial based on at least t its values and computing the value at zero of this polynomial. Generally, we often use here Lagrange interpolation formula.

A more general approach to the presented problem can be found by the reader in [12-14].

2.1. Secret sharing

A dealer (a trusted third party) has to generate a random password K ($K \in \mathbb{N}_+$). Then the dealer shares the key among n parties (participants). He executes this as follows; he randomly selects a prime number p of a feature that $p > \max(K, n)$ and assumes that $a_0 = K$. Now the dealer randomly chooses $t-1$ coefficients $a_1, a_2, \ldots, a_{t-1}$ belonging to the field $\mathbb{Z}_p$ and he denotes $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$. The dealer computes $K_i = f(i) \bmod p$ for $1 \leqslant i \leqslant n$, and he distributes a pair of numbers (i, K_i) to each participant.

Distribution of password K

1. The dealer generates a prime number p greater than $\max(K, n)$ and defines $a_0 = K$.
2. He denotes in field $\mathbb{Z}_p, t-1$ random coefficients $a_1, a_2, \ldots, a_{t-1}$ and considers polynomial over field Z_p as $f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{t-1} x^{t-1}$.
3. He computes $K_i = f(i) \bmod p$, for $1 \leqslant i \leqslant n$ and sends the share (i, K_i) to the party i.

2.2. Password reconstruction

t or more parties (the participants) can appoint password K. There are several options that can be proposed here. The most common way of obtaining a key K is the Lagrange interpolation formula. Suppose, we have shares t of participants $\left(i_1, \mathrm{K}_{i_1}\right), \left(i_2, \mathrm{K}_{i_2}\right), \ldots, \left(i_t, \mathrm{K}_{i_t}\right)$.

Now, we get K from $\mathrm{K} = f(0) \bmod p = \sum_{\mathrm{j}=1}^{t} K_{i_j} \prod_{1 \leqslant \mathrm{k} \leqslant \mathrm{t},\, \mathrm{k} \neq j} \frac{i_k}{i_k - i_j} \bmod p$.

3. The product of two unknown to anyone (distributed) factors - the BGW method

This section discusses the protocol of distributed product computing - the BGW protocol [3, 4] (acronym of authors' last names Ben-Or M., Goldwasser S., Wigderson A.). The party in the submitted protocols is equated with the participant (computer or server) of protocol. In the submitted protocols, three or more parties will take part. We assume that each party may communicate with any other party. Messages sent from one participant to the other are private, and no one can interfere along the way. Furthermore, we assume that all parties of the protocol are fair.

In consequence of this, presented protocol is $\frac{n-1}{2}$ private. This means that any coalition of $\frac{n-1}{2}$ (or less) parties does not recognize the factors of the computed product. However, a coalition of more than $\frac{n-1}{2}$ parties may recover the factors of the mentioned product. The reason for this is privacy of the used BGW method. Replacing the BGW method by the Cocks method [15], the presented method can achieve the greatest possible privacy, i.e. $n-1$.

In our protocol there are n participants involved, and N is the product of two numbers p and q. The party i has secret numbers p_i and q_i, called its shares.

The sum of the shares is p and q in such a way that $p = \sum_{\mathrm{i}=1}^{n} p_i$, $q = \sum_{\mathrm{i}=1}^{n} q_i$. This method shows a distributed computing $N = \left(\sum_{\mathrm{i}=1}^{n} p_i\right)\left(\sum_{\mathrm{i}=1}^{n} q_i\right) \bmod P$ so that none of the parties could know both p or q, and only a coalition of more than $\frac{n-1}{2}$ parties can recognize factors p and q. So, in order to know the factorization of N, you have to bribe at least half of the protocol participants. The protocol is derived

from the work of Ben-Or, Goldwasser and Wigderson, in which the authors describe an elegant protocol to compute N, for three or more parties. Practically, we take a prime number P greater than N. Let also $s = \left\lfloor \frac{n-1}{2} \right\rfloor$.

The BGW method for distributed computation $N = (\sum p_i) \cdot (\sum q_i) \bmod P$

1. Each party chooses two random polynomials of degree s, i.e. party i chooses $f_i, g_i \in \mathbb{Z}_P[X]$, such that $f_i(0) = p_i$, $g_i(0) = q_i$ and a random polynomial $h_i \in \mathbb{Z}_P[X]$ of degree $2s$, such that $h_i(0) = 0$.
2. Each party computes $3n$ values, i.e. party i computes: $\forall_{1 \leqslant j \leqslant n}\, p_{i,j} = f_i(j)$, $q_{i,j} = g_i(j)$, $h_{i,j} = h_i(j)$. Party i sends to party j: $p_{i,j}, q_{i,j}, h_{i,j}$.
3. Party i computes: $N_i = \left(\sum_{j=1}^{n} p_{j,i}\right)\left(\sum_{j=1}^{n} q_{j,i}\right) + \left(\sum_{i=1}^{n} h_i\right) \bmod P$. The result is given to the public or transmitted to other parties.
4. Parties, having N_i with using Lagrange interpolation formula, receive polynomial $\alpha(x) = \left(\sum_{j=1}^{n} f_j(x)\right)\left(\sum_{j=1}^{n} g_j(x)\right) + \left(\sum_{j=1}^{n} h_j(x)\right) \pmod P$.

In view of the equality $\alpha(0) = N$ parties receive N.

4. Proposed protocols

In this section we will present original protocols checking the equality of two distributed numbers and distributed multiplication, using the Chinese Remainder Theorem. We will indicate the correctness of these protocols and we will present their bit complexity. Although such solutions are known so far, we have not cognized such a type of approach for the considered issues.

4.1. Chinese Remainder Theorem

Chinese Remainder Theorem is a theorem that is widely applicable in cryptography. It can often be used to speed up the computations. In addition, it is used to construct a number of libraries for computations on large integers. Libraries of this type find application in the aforementioned cryptography.

Theorem (Chinese Remainder Theorem).
If the integers $n_1, n_2, \ldots, n_k$ are pairwise relatively prime, then the system of simultaneous congruences

$$\begin{aligned} x &\equiv a_1 \bmod n_1 \\ x &\equiv a_2 \bmod n_2 \\ &\vdots \\ x &\equiv a_k \bmod n_k \end{aligned}$$

has a unique solution modulo $N = n_1 \cdot n_2 \ldots n_k$.

Gauss's Algorithm. The solution x to the simultaneous congruences in the Chinese Remainder Theorem may be computed as $x = \sum a_i N_i M_i \bmod N$, where $N_i = N/n_i$ and $M_i = N_i^{-1} \bmod n_i$ [2, Ch. 2]. It is easy to notice that the algorithm can use a mechanism of parallelization or distribution. We use the algorithm in the following protocols. If we know the values a_i, n_i, N, we can determine the value x in a fast $O(\log^2 N)$ operations) and simple way.

4.2. The protocol verifying equality of two distributed numbers

The following procedure involves a group of k participants of the protocol, in which the participant i holds the shares P_i and Q_i.

In the procedure given below, the result of subtraction of Q and P does not leak any information if the coalition of conspiring parties is less than k.

The procedure returns true, when $\sum P_i = \sum Q_i$, and false, when $\sum P_i \neq \sum Q_i$. We assume that all parties know a collection of k primes $p_1, p_2, ..., p_k$ (or, at least, pairwise relative primes) such that $\sum(Q_i - P_i) < \prod p_i$. To improve efficiency, the public primes $p_1, p_2, ..., p_k$ should be as small as possible and at the same time they should be close to each other.

The main idea of this procedure is based on the Chinese Remainder Theorem. Step 1 gives a new distributed number $S = Q - P$. Next, each party i produces shares w_j^i for the integer $S_i = Q_i - P_i$ using the Chinese Remainder Theorem. Shares are sent via secure channels to the other parties. In Step 4, all parties check the equality and broadcast their results. In this chapter, we assume that the value of λ_i is $(-1)^{k+1}\binom{k}{i}$.

equal (P_i, Q_i, k, p_1, p_2, ..., p_k; $\sum P_i = \sum Q_i$ *is true*)
INPUT: of party i: P_i, Q_i, k and the set $p_1, p_2, ..., p_k$ of public primes.
OUTPUT: of party i: *true*/*false* an answer to the question: whether $\sum P_i = \sum Q_i$?

1. Each party computes its own share in S, which is the result of subtraction of P and Q; i.e., party i computes $S_i = Q_i - P_i$.
2. Each party i computes k values $w_1^i, w_2^i, ..., w_k^i$ where $w_j^i = S_i \bmod p_j$.
3. Party i generates k polynomials $Q_1(x), Q_2(x), ..., Q_k(x)$ of degree $k-1$ over the fields $\mathbb{Z}_{p_1}, \mathbb{Z}_{p_2}, ..., \mathbb{Z}_{p_k}$ with the free coefficients $w_1^i, w_2^i, ..., w_k^i$, respectively.
4. Party i computes k^2 values $Q_m(n)$, for m, n from 1 to k.
5. Party i sends $v_j^i = Q_m(j)$ to party j.
6. Party i computes $b_i = \lambda_i \sum v_i^j \bmod p_i$.
7. Party i generates the random value r_i and computes $r_i \cdot b_i$.
8. Party i generates polynomial $Z_q(x)$ with integer coefficients of degree k with free coefficient $r_i \cdot b_i$.

9. Party i computes k values $Z_q(1), Z_q(2), ..., Z_q(k)$.
10. Party i sends to party j the computed values $z_j^i = Z(j)$.
11. Party i computes $s_i = \lambda_i \sum z_i^j \bmod q$.
12. Parties reveal their values s_i, if the sum equals zero, they return *true*, otherwise *false*.

Analysis of complexity

In the procedure analysed here, we assume that the length of all the shares in the distributed number does not exceed n bits and the number of participants is k. The communication overload complexity (the number of bits that each participant must send to the others) of the procedures is $O(k \cdot n)$.

The procedure **equal** ($P_i, Q_i, k, p_1, p_2, ..., p_k; \sum P_i = \sum Q_i$ *is true*) needs $O(n \cdot k^3 \cdot \log k$ bit operations and their communication complexity is $Ok{\cdot}n$.

Correctness of the distributed equality checking

The protocol checking the equality of two distributed numbers is **equal**(P_i,Q_i, k ,p_1, p_2, ..., p_k; $\sum P_i = \sum Q_i$ *is true*). Each party involved in this procedure inputs, in addition to its shares in the two distributed integers, k (relative) primes. The product of those primes is supposed to be bigger than the difference between the two numbers that we are comparing; i.e. $Q - \mathrm{P} < \prod p_i$. The protocol returns *true* when $\left(\sum w_j^i \bmod p_j\right) = \left(\sum v_j^i \bmod p_j\right)$, for each $1 \leqslant j \leqslant k$. We continue our analysis for a fixed prime p_j. The last equality entails that in step 3 of the procedure we have $\left(\sum S_i^1 \bmod p_j\right) \mathrm{mod} p_j = \left(\sum S_i^2 \bmod p_j\right) \bmod p_j$. Since $S_i = \mathrm{S}_i^1 - S_i^2$, we get $(\sum S_i) \bmod p_j = 0$. Hence, for each $1 \leqslant j \leqslant k$, we have $(Q - P) \bmod p_j = 0$. Finally, by the fact that $Q - \mathrm{P} < \prod p_i$ we get the equality of P and Q.

4.3. The protocol computing the product of two distributed factors with use of the Chinese Remainder Theorem

This section discusses the protocol of distributed product computing with the use of the Chinese Remainder Theorem. As in the previously submitted protocols, there are three or more parties participating in the protocol. We assume that each party may communicate with any other party. Messages sent from one participant to the other are private, and no one can interfere along the way. Furthermore, we assume that all parties of the protocol are fair.

In consequence of this, the presented protocol is $k - 1$ private. There are k participants involved in this protocol and N is the product of P and Q. The party i has two secret numbers P_i and Q_i called its shares. The sum of the parts is P and Q, such that $P = \sum P_i$, $Q = \sum Q_i$. This method presents distributed computation

$N = (\sum P_i)(\sum Q_i) \bmod M$, so that no party could learn neither P nor Q and only a coalition of more than $k-1$ parties can get to know factors P and Q. So, in order to know factorization N, you have to bribe at least half of the protocol participants. Practically, you have to take a prime number M greater than N.
Below we present the protocol of distributed computing of $N = (\sum P_i)(\sum Q_i) \bmod M$.

multiplication (P_i, Q_i, k, p_1,p_2, ..., p_k; $N = (\sum P_i)(\sum Q_i) \bmod M$)
INPUT: of party i: P_i,Q_i,k and the set p_1,p_2,...,p_k of public primes.
OUTPUT: $N = (\sum P_i)(\sum Q_i) \bmod M$

1. Party i computes $2 \cdot k$ values $v_i = P_i \bmod p_j$ and $w_i = Q_i \bmod p_j$ for $1 \leqslant j \leqslant k$.
2. For each party, the participants come together to compute the values of the two sums, i.e. for party i they compute $s_i^1 = \sum v_j \bmod p_i$ and $s_i^2 = \sum w_j \bmod p_i$ [see below, sum].
3. Party i computes: $N_i = \prod_{j \neq i} p_j$, $M_i = N_i^{-1} \bmod p_i$ and values $X_i = N_i \cdot M_i \cdot s_i^1$ and $Y_i = N_i \cdot M_i \cdot s_i^2$.
4. Using the BGW method, parties compute the value: $N = (\sum X_i)(\sum Y_i) \bmod M$.

The value computed in step 4 is the value of the product $N = (\sum P_i)(\sum Q_i) \bmod M$. The second step of this protocol is implemented on the basis of common computing of the distributed values sum, for a chosen party. Below we present an example of implementation of protocol for multiparty computing the sum (example implementation for step 2).

sum (P_i, k, j; $\sum P_i$)
INPUT: of party i: P_i, k, j.
OUTPUT: of party j: $\sum P_i$.

1. Each party generates a random polynomial of degree $k-1$ over a finite field with a value of zero equal to its share. Let party i generate the polynomial $f_i(x) = a_{k-1} x^{k-1} + a_{k-2} x^{k-2} + ... + a_1 x + P_i$.
2. Each party computes k values of our polynomial e.g. values for 1, 2, ... , k.
3. Each party sends the value for argument j to the party of number j.
4. Party of number i adds the values obtained in step 3.
5. Party i multiplies the sum obtained in step 4. by the value $(-1)^{k+1} \binom{k}{i}$.
6. Parties send the value obtained in step 5. to party j.
7. Party j computes the sum s_j of values obtained in step 6.

The protocol presented above for computing the sum is $k-1$ private. So, any coalition of $k-1$ or less parties (excluding the party of number j) does not know the computed secret.

The Chinese Remainder Theorem used here increases the cryptographic power of protocol for ordinary multiplication of two distributed numbers.

5. Conclusions

In this paper we focus on the applications of the Chinese Remainder Theorem in protocols for verifying the equality of two distributed numbers and computing the product of two distributed numbers. The discussed protocols form the beginning of our research on applications in the field of multilateral Chinese Remainder Theorem computations. We intend to expand our research into other protocols, among others, protocols for comparing distributed numbers (Millionaires' problem) or protocols for remainder computation etc.

Presented protocols can be used in more advanced protocols, including distributed primality testing. The problem will be described in the next article which will be published soon. In this article I will be co-authored. Another proposed research will be application of the proposed protocols.

References

[1] Shamir A., How to share a secret, Communications of the ACM 1979, 22(11), 612-613.

[2] Menezes A., van Oorschot P., Vanstone S.A., Handbook of Applied Cryptography, CRC Press, 2013.

[3] Ben-Or M., Goldwasser S., Wigderson A., Completeness theorems for noncryptographic fault-tolerant distributed computation, Proceeding STOC '88, Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, 1988, 1-10.

[4] Boneh D., Franklin M., Efficient generation of shared RSA key, Advances in Cryptology - CRYPTO'97, Springer, 1997, 1296, 425-439.

[5] Mignotte M., How to share a secret, Advances in Cryptology, Eurocrypt'82, LNCS, Springer-Verlag, 1983, 149, 371-375.

[6] Frankel Y., MacKenzie P.D., Yung M., Robust efficient distributed RSA-key generation, Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC'98, ACM, New York 1998.

[7] Algesheimer J., Camenisch J., Shoup V., Efficient computation modulo a shared secret with application to the generation of shared safe-prime products, Advances in Cryptology, Proceedings of CRYPTO 2002, LNCS, 2002, 2442, 417-432.

[8] Kiltz E., Unconditionally secure constant round multi-party computation for equality, comparison, bits and exponentiation, Proceedings of the Third Theory of Cryptography Conference 2005.

[9] Damgård I., Fitzi M., Kiltz E., Nielsen J.B., Toft T., Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation, Theory of Cryptography Lecture Notes in Computer Science 2006, 3876, 285-304.

[10] Chao Ning, Qiuliang Xu, Constant-rounds, linear multi-party computation for exponentiation and modulo reduction with perfect security, Proceedings of Asiacrypt'11, Springer-Verlag, 2011, 572-586.

[11] Blakley G.R., Safeguarding cryptographic keys, Proc. AFIPS 1979, National Computer Conference, AFIPS, 1979, 313-317.

[12] Chun-Pong Lai, Cunsheng Ding, Several generalizations of Shamir's secret sharing scheme, Internat. J. Found. Comput. Sci. 2004, 15, 445-458.

[13] Schinzel A., Spież S., Urbanowicz J., Elementary symmetric polynomials in Shamir's scheme, Journal of Number Theory 2010, 1572-1580.

[14] Spież S., Srebrny M., Urbanowicz J., Remarks on the classical threshold secret sharing schemes, Fundamenta Informaticae 2012, 114 (3-4), 345-357.

[15] Cocks C., Split knowledge generation of RSA parameters in cryptography and coding, 6th IMA Conference, LNCS, Springer-Verlag, 1997, 1355, 89-95.