Reducing the Complexity in the Distributed Multiplication Protocol of Two Polynomially Shared Values

Peter Lory
Universität Regensburg
Institut für Wirtschaftsinformatik
D-93040 Regensburg, Germany
Peter.Lory@wiwi.uni-regensburg.de

Abstract

The multiparty multiplication of two polynomially shared values over $\mathbb{Z}_q$ with a public prime number q is an important module in distributed computations. The multiplication protocol of Gennaro, Rabin and Rabin (1998) is considered as the best protocol for this purpose. It requires a complexity of $O(n^2k\log n + nk^2)$ bit-operations per player, where k is the bit size of the prime q and n is the number of players. The present paper reduces this complexity to $O(n^2k + nk^2)$ by using Newton's classical interpolation formula. The impact of the new method on distributed signatures is outlined.

1. Introduction

The last two decades have seen an exciting development of techniques for secure multiparty computations. Classical theoretical results [2, 6, 10, 16] show that any multiparty computation can be performed securely, if the number of corrupted participants does not exceed certain bounds. However already Gennaro, Rabin and Rabin [9] point out, that these generic secure circuit techniques are too inefficient in the area of practical feasibility, which might render them impractical. Thus, it is a high priority to optimize such techniques.

The present paper focuses on the multiparty multiplication of two polynomially shared values over $\mathbb{Z}_q$ with a public prime number q. Polynomial sharing refers to the threshold scheme originally proposed by Shamir [14]. It is assumed that n players share a secret α in a way that each player P_i ($1 \leq i \leq n$) owns the function value $f_\alpha(i)$ of a polynomial f_α with maximum degree t and $\alpha = f_\alpha(0)$. Then any subset of $t+1$ participants can retrieve the secret α (for example by Lagrange's interpolation formula) but no subset of, at most, t participants can do so. At the beginning of the multiplication protocol each player P_i holds as input the function values $f_\alpha(i)$ and $f_\beta(i)$ of two polynomials f_α and f_β with maximum degree t and $\alpha = f_\alpha(0), \beta = f_\beta(0)$. At the end of the protocol each player owns the function value $H(i)$ of a polynomial H with maximum degree t as his share of the product $\alpha\beta = H(0)$. It is assumed that the distributed multiplication takes place under the so-called "honest-but-curious" model. This means, that an adversary is passive and only tries to deduce information but follows the protocol honestly. In the protocols considered in this paper each participant P_i does not learn any information about the inputs of the other players except for what is revealed by his function value $H(i)$ (information theoretic security). Multiplication protocols of this type are important cryptographic primitives. In particular, they play a decisive role in the shared generation of an RSA [13] modulus by a number of participants such that none of them knows the factorization (see [1, 4]).

A first multiplication protocol of the above type has been presented by Ben-Or, Goldwasser and Wigderson [2]. A considerable improvement was proposed by Gennaro, Rabin and Rabin [9]. Presently, their approach is considered as the most efficient protocol (see [1, 4]). It requires $O(n^2k\log n + nk^2)$ bit operations per player. Here, k is the bit size of the prime q and n is the number of players. In the present paper, this complexity is reduced to $O(n^2k + nk^2)$. Remarkably, the key idea for this success is the application of a rather old technique, namely Newton's interpolation formula (*Methodus Differentialis*, 1676).

The paper is organized as follows: Section 2 presents the the protocol of Gennaro, Rabin and Rabin [9] for the reader's convenience and investigates its complexity in detail. Section 3 gives basic material on Newton's scheme of divided differences for reference in Section 4. In this section the new protocol is presented and its complexity is studied. In Section 5 the impact of the new technique on distributed signatures is outlined.

2. The Protocol of Gennaro, Rabin and Rabin

The protocol in [9] assumes that two secrets α and β are shared by polynomials $f_\alpha(x)$ and $f_\beta(x)$ respectively and the players would like to compute the product $\alpha\beta$. Both polynomials are of maximum degree t. Denote by $f_\alpha(i)$ and $f_\beta(i)$ the shares of player P_i on $f_\alpha(x)$ and $f_\beta(x)$ respectively. The product of these two polynomials is

$$f_\alpha(x)f_\beta(x) = \gamma_{2t}x^{2t} + \ldots \gamma_1 x + \alpha\beta \overset{def}{=} f_{\alpha\beta}(x).$$

It is easy to see (cf. [9]) that

$$\alpha\beta = \lambda_1 f_{\alpha\beta}(1) + \ldots + \lambda_{2t+1} f_{\alpha\beta}(2t+1)$$

with known non-zero constants λ_i. Let $h_1(x), \ldots, h_{2t+1}(x)$ be polynomials of maximum degree t which satisfy that $h_i(0) = f_{\alpha\beta}(i)$ for $1 \le i \le 2t+1$. Define

$$H(x) \overset{def}{=} \sum_{i=1}^{2t+1} \lambda_i h_i(x)\,,$$

then this function is a polynomial of maximum degree t with the property

$$H(0) = \lambda_1 f_{\alpha\beta}(1) + \ldots + \lambda_{2t+1} f_{\alpha\beta}(2t+1) = \alpha\beta\,.$$

Clearly, $H(j) = \sum_{i=1}^{2t+1} \lambda_i h_i(j)$. Thus, if each of the players P_i $(1 \le i \le 2t+1)$ shares his share $f_{\alpha\beta}(i)$ with the other participants using a polynomial $h_i(x)$ with the properties as defined above, then the product $\alpha\beta$ is shared by the polynomial $H(x)$ of maximum degree t. These ideas are the basis of the protocol given in Figure 1.

Please note, that the protocol implicitly assumes that the number n of players obeys $n \ge 2t+1$.

For the investigation of the complexity the basic assumption is made, that the bit-complexity of a multiplication of a k-bit-integer and an l-bit-integer is $O(kl)$. This is a reasonable estimate for realistic values (e.g. $k = l = 1024$). Step 1 of the protocol of Figure 1 requires n evaluations of the polynomial $h_i(x)$ of degree t. If Horner's scheme (c.f. Stoer and Bulirsch [15]) is used for this purpose, one evaluation requires t multiplications of a k-bit integer and an integer with at most $\log n + 1$ bits. In step 2 of the protocol each player has to compute $2t+1$ multiplications of two k-bit numbers. Taking into account that $t < 2t+1 \le n$, a complexity of $O(n^2 k \log n + nk^2)$ bit-operations per player follows. This is consistent with the corresponding propositions in Algesheimer, Camenish and Shoup [1] and Catalano [4].

3. Newton's Scheme of Divided Differences

Newton's interpolation formula on the basis of the concept of divided differences is described in many textbooks

Input of player P_i: The values $f_\alpha(i)$ and $f_\beta(i)$.

1. Player P_i $(1 \le i \le 2t+1)$ computes $f_\alpha(i)f_\beta(i)$ and shares this value by choosing a random polynomial $h_i(x)$ of maximum degree t, such that

$$h_i(0) = f_\alpha(i)f_\beta(i)\,.$$

   He gives player P_j $(1 \le j \le n)$ the value $h_i(j)$.

2. Each player P_j $(1 \le j \le n)$ computes his share of $\alpha\beta$ via a random polynomial H, i.e. the value $H(j)$, by locally computing the linear combination

$$H(j) = \sum_{i=1}^{2t+1} \lambda_i h_i(j)\,.$$

on numerical analysis. For later reference, this section presents the basic facts following the notation in Stoer and Bulirsch [15].

Let the support abscissas x_i and corresponding support ordinates f_i $(0 \le i \le m)$ be given. The *divided differences* are defined recursively by

$$f_{i_0,i_1,\ldots,i_l} \overset{def}{=} \frac{f_{i_1,i_2,\ldots,i_l} - f_{i_0,i_1,\ldots,i_{l-1}}}{x_{i_l} - x_{i_0}} \qquad (1)$$

and can be arranged in a tableau, the so-called *divided-difference scheme* (see Figure 2).

$$\begin{array}{c|ccccc}
 & l=0 & l=1 & l=2 & \ldots & l=m \\
\hline
x_0 & f_0 & & & & \\
 & & f_{0,1} & & & \\
x_1 & f_1 & & f_{0,1,2} & & \\
 & & f_{1,2} & & \ddots & \\
x_2 & f_2 & & \vdots & & f_{0,1,2,\ldots,m} \\
 & & \vdots & & \iddots & \\
\vdots & \vdots & & f_{m-2,m-1,m} & & \\
 & & f_{m-1,m} & & & \\
x_m & f_m & & & &
\end{array}$$

Clearly, the entries in column $l = 1$ are of the form

$$f_{0,1} = \frac{f_1 - f_0}{x_1 - x_0}\,, \quad f_{1,2} = \frac{f_2 - f_1}{x_2 - x_1}\,, \quad \ldots,$$

those in column $l = 2$

$$f_{0,1,2} = \frac{f_{1,2} - f_{0,1}}{x_2 - x_0}\,, \quad f_{1,2,3} = \frac{f_{2,3} - f_{1,2}}{x_3 - x_1}\,, \quad \ldots,$$

Instead of building the divided-difference scheme column by column, the most convenient way of computation is to start with the upper left corner and add successive ascending diagonal rows.

It is easy to see (c.f. [15]) that the polynomial

$$\begin{aligned} P(x) \quad := \quad & f_0 + f_{0,1}(x - x_0) + \ldots + \\ & f_{0,1,\ldots,m}(x - x_0)(x - x_1)\ldots(x - x_{m-1}) \end{aligned}$$

with the entries of the uppermost descending diagonal row in the divided-difference scheme as its coefficients is the interpolating polynomial of maximum degree m that interpolates the given support abscissas and support ordinates:

$$P(x_i) = f_i \quad i = 0, 1, \ldots, m\,.$$

4. The New Protocol and its Complexity

4.1 The new protocol

The key for reducing the complexity in the multiplication protocol of Gennaro, Rabin and Rabin [9] is the observation that in Step 1 of the protocol in Figure 1 each of the players P_i $(1 \le i \le 2t + 1)$ chooses a random polynomial of maximum degree t

$$h_i(x) = a_t x^t + a_{t-1}x^{t-1} + \ldots + a_1 x + a_0$$

with $a_0 = f_\alpha(i) f_\beta(i)$ and then has to evaluate this polynomial at n different points. The present paper suggests that instead of choosing the coefficients a_j $(1 \le j \le t)$, each of the players P_i $(1 \le i \le 2t + 1)$ randomly picks t support ordinates f_j for the t abscissas $x_j = j$ $(1 \le j \le t)$. Together with the condition

$$h_i(0) = f_\alpha(i) f_\beta(i)$$

this implicitly defines the unique interpolation poynomial $h_i(x)$ of maximum degree t. Then player P_i has to evaluate this polynomial for $x_j = j$ $(t+1 \le j \le n)$. Using Newton's scheme of divided differences these computations can be performed very efficiently. The details are given in Figure 3 with $f_j = h_i(j)$ $(0 \le j \le n)$. For readability reasons the index i is omitted.

A few remarks are in place:

1. As in [9] the support abscissas for the interpolating polynomial $h_i(x)$ are chosen as $x_j = j$ for $0 \le j \le n$.
2. Instead of calculating the diveded differences as defined in Equation (1), the numbers
   $$f_{i_0,i_1,\ldots,i_l} \cdot (x_{i_l} - x_{i_0}) = f_{i_1,i_1,\ldots,i_l} - f_{i_0,i_1,\ldots,i_{l-1}}\,.$$
   are computed. This modification is the reason for the factors $l!$ in column l of the scheme of Figure 3 and avoids superfluous arithmetic operations.
3. The zeros in the columns $l = t+1, t+2, \ldots, n$ of Figure 3 are not computed. Instead, they are prescribed and force the interpolating polynomial to be of maximum degree t. As a consequence, all the entries in column $l = t$ are identical.
4. The first $t + 1$ ascending diagonal rows are computed from left to right starting from the prescribed support ordinate $f_0 = h_i(0) = f_\alpha(i) f_\beta(i)$ and the randomly chosen support ordinates $f_1 = h_i(1), \ldots, f_t = h_i(t)$. The following diagonal rows are computed from right to left starting from the entries in column $l = t$ (which are identical to the already computed value at the top of this column) and ending in the computed support ordinates
   $$f_{t+1} = h_i(t + 1), \ldots, f_n = h_i(n)\,.$$

These ideas are the basis of the new protocol given in Figure 4, where all operations take place in $\mathbb{Z}_q$ with a public prime number q (see Section 1).

A few comments are in place:

1. Step 1(a) of the protocol in Figure 4 calculates the upper left corner in Newton's diveded difference scheme of Figure 3.
2. Step 1(b) of this protocol calculates the following t ascending diagonal rows from left to right. Here, the index k is running downwards for storage efficiency reasons.
3. Step 1(c) of this protocol calculates the following $n-t$ ascending diagonal rows from right to left.

Apart from technical details in the calculations, the protocol of Gennaro, Rabin and Rabin [9] (cf. Figure 1) and the new protocol of Figure 4 differ in only one respect: In the protocol of Gennaro, Rabin and Rabin each player P_i randomly chooses a polynomial $h_i(x)$ of maximum degree t by choosing its coefficients of $x^1, x^2, \ldots, x^t$. In the new protocol the same player P_i randomly chooses t support ordinates for this polynomial:

$$f_1 = h_i(1), \ldots, f_t = h_i(t)\,.$$

$$\begin{array}{c|cccccccc}
 & l=0 & l=1 & l=2 & \dots & l=t & l=t+1 & \dots & l=n \\ \hline
0 & f_0 & & & & & & & \\
 & & f_{0,1} & & & & & & \\
1 & f_1 & & 2!\cdot f_{0,1,2} & & & & & \\
 & & f_{1,2} & & \ddots & & & & \\
2 & f_2 & & \vdots & & t!\cdot f_{0,1,\dots,t} & & & \\
 & & \vdots & & \iddots & & 0 & & \\
\vdots & \vdots & & 2!\cdot f_{t-2,t-1,t} & & t!\cdot f_{1,2,\dots,t+1} & & \ddots & \\
 & & f_{t-1,t} & & \iddots & & \vdots & & 0 \\
t & f_t & & 2!\cdot f_{t-1,t,t+1} & & \vdots & & \iddots & \\
 & & f_{t,t+1} & & \vdots & & 0 & & \\
t+1 & f_{t+1} & & \vdots & & t!\cdot f_{n-t,n-t+1,\dots,n} & & & \\
 & & \vdots & & \iddots & & & & \\
\vdots & \vdots & & 2!\cdot f_{n-2,n-1,n} & & & & & \\
 & & f_{n-1,n} & & & & & & \\
n & f_n & & & & & & &
\end{array}$$

This difference does not affect the randomness of the chosen polynomial $h_i(x)$. Therefore, the proof of Theorem 3 in [9] applies to the new protocol as well and the following theorem follows:

Theorem 1 *The protocol of Figure 4 is a secure multiplication protocol in the presence of a passive adversary computationally unbounded.*

4.2 Complexity of the new protocol

Step 1(b) of the new protocol (Figure 4) needs $t(t+1)/2$ additions of two k-bit numbers, where k is the bit size of the prime q. Step 1(c) of the same protocol requires $(n-t)t$ additions, where n is the number of players and $t+1$ is the threshold. Clearly, the complexity for the addition of two k-bit numbers is $O(k)$. Since $t < 2t+1 \leq n$, a complexity of $O(n^2k)$ bit-operations per player for step 1 of the new protocol follows. Step 2 requires $O(nk^2)$ bit-operations (see Section 2). So the following theorem is proven:

Theorem 2 *The multiplication protocol of Figure 4 requires $O(n^2k + nk^2)$ bit-operations per player.*

This result has to be compared with the complexity of $O(n^2k \log n + nk^2)$ for the multiplication protocol of Gennaro, Rabin and Rabin [9].

5. Impact on Distributed Signatures

Algesheimer, Camenisch and Shoup [1] and Catalano [4] have pointed out the attractiveness of modularity in the construction of protocols for distributed computations. Simple protocols can be combined to address more complicated tasks.

Let n players want to jointly share a random secret in $\mathbb{Z}_q$ with a public prime number q. For this purpose, each player chooses a random value $r_i \in \mathbb{Z}_q$ and shares this value according to the polynomial sharing sheme [14] as described in Section 1. Then each player sends the obtained shares to the remaining players involved in the protocol. At this point each player sums up (modulo q) all the received values and sets the obtained value as his share of the jointly chosen random value. Please note, that no trusted dealer is involved in this process.

Let n players want to distributively generate a shared RSA modulus N being the product of two primes or of two safe primes without the need for a trusted dealer. This requires the execution of rather complicated protocols: Reduction of a shared integer modulo a shared p, distributed versions of the square and multiply algorithm and of the Miller-Rabin primality test [11, 12] (cf. Algesheimer, Camenisch and Shoup [1] and Catalano [4]). All these protocols call for the distributive multiplication module to a high extent. Consequently, they significantly benefit from

Input of player P_i: The values $f_\alpha(i)$ and $f_\beta(i)$.

1. Player P_i ($1 \le i \le 2t+1$) computes $f_\alpha(i) f_\beta(i)$ and shares this value by randomly choosing t support ordinates $f_1, f_2, \ldots, f_t$ and executing the following steps:

```
(a)
        $g_0 := f_\alpha(i) f_\beta(i)\,.$
(b) For $j = 1, 2, \ldots, t$ :
        $g_j := f_j\,,$
        for $k = j-1, j-2, \ldots, 0$ :
            $g_k := g_{k+1} - g_k\,.$
(c) For $j = t+1, t+2, \ldots, n$ :
        for $k = 0, 1, \ldots, t-1$ :
            $g_{k+1} := g_{k+1} + g_k\,.$
        $f_j := g_t\,.$
```

He gives player P_j ($1 \le j \le n$) the value

$$f_j^{(i)} := f_j\,.$$

2. This step is identical to Step 2 in the protocol of Gennaro, Rabin and Rabin (Figure 1) with

$$h_i(j) = f_j^{(i)}\,.$$

the reduction of complexity as described in Section 4 of the present paper, in particular when the number n of players is large.

The subsequent distributive generation of shares of the private exponent is much less computationally involved than distributively generating the modulus N. In particular, Boneh and Franklin [3] and Catalano, Gennaro and Halevi [5] present efficient protocols to accomplish this. One of the main applications of these results is the construction of threshold variants of signature schemes. In such a scheme n parties hold a t-out-of-n sharing of the secret key. Only when at least $t+1$ of them cooperate they can sign a given message. The reader is referred to [5], where two such signature schemes are constructed. The first is an appropriate variant of the signature scheme of Gennaro, Halevi and Rabin [8]; the second relies on the signature scheme of Cramer and Shoup [7].

References

[1] J. Algesheimer, J. Camenisch, and V. Shoup. Efficient computation modulo a shared secret with application to the generation of shared safe-prime products. In *Advances in Cryptology – CRYPTO 2002*, Lecture Notes in Computer Science 2442:417–432, Springer, Berlin, 2002.
Full version: `http://eprint.iacr.org/2002/029`.

[2] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of 20th Annual Symposium on Theory of Computing (STOC'88)*, 1–10, ACM Press, 1988.

[3] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. In *Advances in Cryptology – CRYPTO 1997*, Lecture Notes in Computer Science 1294:425–439, Springer, Berlin, 1997.

[4] D. Catalano. Efficient distributed computation modulo a shared secret. In D. Catalano, R. Cramer, I. Damgård, G. Di Crescenco, D. Pointcheval, and T. Takagi (eds.) *Contemporary Cryptology*, Advanced Courses in Mathematics, CRM Barcelona, 1–39, Birkhäuser, Basel, 2005.

[5] D. Catalano, R. Gennaro, and S. Halevi. Computing inverses over a shared secret modulus. In *Advances on Cryptology – EUROCRYPT 2000*, Lecture Notes in Computer Science 1807:190–206, Springer, Berlin, 2000.

[6] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proceedings of 20th Annual Symposium on Theory of Computing (STOC'88)*, 11–19, ACM Press, 1988.

[7] R. Cramer and V. Shoup. Signature schemes based on the Strong RSA Assumption. *ACM Transactions on Information and System Security (ACM TISSEC)*, 3(3):161-185, 2000.

[8] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Advances in Cryptology – EUROCRYPT 1999*, Lecture Notes in Computer Science 1592:123–139, Springer, Berlin, 1999.

[9] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the 17th ACM Symposium on Principles of Distributed Computing (PODC'98)*, 1998.

[10] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of 19th Annual Symposium on Theory of Computing (STOC'87)*, 218–229, ACM Press, 1987.

[11] G. L. Miller. Riemann's hypothesis and tests for primality. *Journal of Computers and System Sciences*, 13:30–317, 1976.

[12] M. O. Rabin. Probabilistic algorithms for testing primality. *Journal of Number Theory*, 12:128–138, 1980.

[13] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[14] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[15] J. Stoer and R. Bulirsch. Introduction to Numerical Analysis, Springer, Berlin, 2002.

[16] A. C. Yao. How to generate and exchange secrets. In *Proceedings of 27th IEEE Symposium on Foundations of Computer Science (FOCS'86)*, 162–167, IEEE Computer Society, 1986.