A Note on Secure Multiparty Multiplication

Prof. Dr. Peter Lory, Jürgen Wenzl

University of Regensburg, D-93040 Regensburg, Germany,

`Peter.Lory@wiwi.uni-regensburg.de`,

,

Abstract. The protocol of Gennaro, Rabin and Rabin (1998) is a well known and efficient protocol for the secure multiparty multiplication of two polynomially shared values over $\mathbb{Z}_q$ with a public prime number q. It requires $O(n^2 k \log n + nk^2)$ bit-operations per player, where k is the bit size of the prime q and n is the number of players. In previous papers (2007, 2009), the first author has presented modifications of this protocol, that reduces its complexity to $O(n^2 k)$. The present report gives an additional modification and compares the running times for these variants by numerical experiments.

1 Introduction

Classical theoretical results [4, 7, 15, 24] show that any multiparty computation can be performed securely if the number of corrupted participants does not exceed certain bounds. For a survey of these results the reader is referred to the article of Cramer and Damgård [8]. However - as Damgård [10] points out - this line of research was clearly oriented towards basic research and there was not much interest in the efficiency of the protocols, beyond the fact that they were polynomial time. Thus, it is a high priority to accelerate the classical protocols. One of the most prominent examples for these efforts is the paper of Gennaro, Rabin and Rabin [13]. Among other results, it presents a more efficient variant of the Ben-Or, Goldwasser and Wigderson [4] multiplication protocol. It gives a protocol for the fast multiparty multiplication of two polynomially shared values over $\mathbb{Z}_q$ with a public prime number q. Section 4 further accelerates this protocol.

Polynomial sharing refers to the threshold scheme originally proposed by Shamir [22], which assumes 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 degree at most 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 $\alpha\beta = H(0)$. Multiplication protocols of this type are important cryptographic primitives. In particular, they play a decisive role in comparing shared numbers (see [11]) and in the shared generation of an RSA [21] modulus by a number of participants such that none of them knows the factorization (see [1, 5]).

The multiplication protocol of Gennaro, Rabin and Rabin [13] requires one round of communication and $O(n^2 k \log n + nk^2)$ bit-operations per player, where k is the bit size of the prime q and n is the number of players. In [17] a modification of this protocol is given, which reduces this complexity to $O(n^2 k + nk^2)$. However, in many practical situations (e. g. the above mentioned shared generation of an RSA modulus) k (typically $k = 1024$) will exceed n and the $O(nk^2)$-term will still dominate. For these cases, in [18] a protocol is given, which requires only $O(n^2 k)$ bit-operations per player. It needs one round of communication.

2 The Network Model and Definitions

It is assumed that the n parties with $n \geq 2t+1$ are connected by perfectly secure point-to-point channels in a synchronous network. Failures in the network are modeled by an adversary $\mathcal{A}$, who can corrupt at most t of the players under the so-called "honest-but-curious" model. This means that the adversary is passive and can read the memories of the corrupted players but not modify their behavior.

Let a be a real number. The symbol $\lceil a \rceil$ ($\lfloor a \rfloor$) denotes the smallest (largest) integer b with $b \geq a$ ($b \leq a$) . The multiplication protocol is called *correct*, if the output values constitute a $(t+1)$-out-of-n secret sharing of $\alpha\beta \mod q$. It is *private*, if the adversary can deduce absolutely nothing about the real values of α, β and $\alpha\beta$. The protocol is called *unconditionally secure*, if it is correct and private. Unconditional security implies that (under the assumed network model) the protocol cannot be broken even with infinite computational resources.

For the investigation of the time complexities two basic assumption are made:

a) The addition or subtraction of two k-bit-integers requires $\rho_{add} k$ bit-operations.
b) The multiplication of a k-bit-integer and an l-bit-integer requires $\rho_{mult} kl$ bit-operations. This is a reasonable estimate for realistic values (e.g. $k = l = 1024$).

The concrete values for ρ_{add} and ρ_{mult} are machine dependent (see e.g. Knuth [16]).

3 The Protocol of Gennaro, Rabin and Rabin

The protocol in [13] 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 shares of 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

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

Due to Lagrange's interpolation formula

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

with the known non-zero constants

$$\lambda_i = \prod_{\substack{1 \le k \le 2t+1 \\ k \ne i}} \frac{k}{k-i} \bmod q \,. \tag{2}$$

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

$$H(x) \overset{def}{=} \sum_{i=1}^{2t+1} \lambda_i h_i(x) \,, \tag{3}$$

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, where all operations are in $\mathbb{Z}_q$.

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

Fig. 1. The multiplication protocol of Gennaro, Rabin and Rabin

Step 1 of the protocol of Figure 1 requires n evaluations of the polynomial $h_i(x)$ of degree t. If Horner's scheme is used for this purpose, one evaluation requires t multiplications of a k-bit integer and an integer with at most $\lceil \log_2 n \rceil$ bits and t additions of two k-bit integers. According to the assumptions in Subsection 2 a total of

$$\rho_{mult} ntk \lceil \log_2 n \rceil + \rho_{add} ntk \tag{4}$$

i.e. $O(n^2 k \log n)$ bit-operations per player in Step 1 follows. In **Step 2** of the protocol each player has to compute $2t+1$ multiplications and $2t$ additions of two k-bit numbers. Consequently,

$$\rho_{mult}(2t+1)k^2 + \rho_{add}2tk \tag{5}$$

i.e. $O(nk^2)$ bit-operations per player are required. This is consistent with propositions in Algesheimer, Camenisch and Shoup [1] and Catalano [5].

4 Accelerations of the protocol of Gennaro, Rabin and Rabin

An acceleration of **Step 1** of the protocol of Figure 1 is given in [17]. It reduces the complexity from $O(n^2 k \log n)$ to $O(n^2 k)$. A comparison of the running times on the basis of numerical experiments is given in [23].

An acceleration of **Step 2** of the protocol of Figure 1 is given in [18]. It reduces the number of bit-operations per player from Equation (5), which is $O(nk^2)$, to

$$[t(2t+1) + 2t]k\,,$$

which is $O(n^2 k)$. This reduction is profitable in situations where n is small, which is often the case.

Another modification of **Step 2** is based on the following observation: Let $d-1$ be the degree of a polynomial. Then the (unreduced) coefficients of Lagrange's interpolation formula with support abscissas $i = 1, 2, \ldots, d$ are given by

$$\lambda_i^{(d)} = \prod_{\substack{1\le k\le d\\ k\ne i}} \frac{k}{k-i}\,. \tag{6}$$

A straightforward calculation yields the values of these coefficients for $1 \le d \le 6$. They are given in Table 1.

	$\lambda_1^{(d)}$	$\lambda_2^{(d)}$	$\lambda_3^{(d)}$	$\lambda_4^{(d)}$	$\lambda_5^{(d)}$	$\lambda_6^{(d)}$
$d=1$	1					
$d=2$	2	-1				
$d=3$	3	-3	1			
$d=4$	4	-6	4	-1		
$d=5$	5	-10	10	-5	1	
$d=6$	6	-15	20	-15	6	-1

Table 1. Coefficients of Lagrange's interpolation formula (see Equation (6))

For general d:

$$\begin{aligned}\lambda_i^{(d+1)} &= \frac{1\cdot 2\cdot\ldots\cdot(i-1)\cdot(i+1)\cdot\ldots\cdot d\cdot(d+1)}{(-(i-1))\cdot(-(i-2))\cdot\ldots\cdot(-1)\cdot 1\cdot 2\cdot\ldots\cdot(d-i)\cdot(d+1-i)}\\ &= (-1)^{i-1}\frac{(d-i+2)\cdot(d-i+3)\cdot\ldots\cdot d\cdot(d+1)}{2\cdot 3\cdot\ldots\cdot i}\,.\end{aligned}$$

Consequently

$$\begin{aligned}
|\lambda_i^{(d+1)}| &= \frac{(d-i+2)\cdot(d-i+3)\cdot\ldots\cdot d\cdot(d+1)}{2\cdot 3\cdot\ldots\cdot i}\,,\\
|\lambda_i^{(d)}| &= \frac{(d-i+1)\cdot(d-i+2)\cdot\ldots\cdot(d-1)\cdot d}{2\cdot 3\cdot\ldots\cdot i}\,,\\
|\lambda_{i-1}^{(d)}| &= \frac{(d-i+2)\cdot(d-i+3)\cdot\ldots\cdot(d-1)\cdot d}{2\cdot 3\cdot\ldots\cdot(i-1)}\,,
\end{aligned}$$

and

$$\begin{aligned}
|\lambda_{i-1}^{(d)}| + |\lambda_i^{(d)}| &= \frac{i\cdot(d-i+2)\cdot(d-i+3)\cdot\ldots\cdot(d-1)\cdot d}{2\cdot 3\cdot\ldots\cdot i} +\\
&\quad \frac{(d-i+1)\cdot(d-i+2)\cdot\ldots\cdot(d-1)\cdot d}{2\cdot 3\cdot\ldots\cdot i}\\
&= \frac{(d-i+2)\cdot(d-i+3)\cdot\ldots\cdot(d-1)\cdot d}{2\cdot 3\cdot\ldots\cdot i}\cdot(i+d-i+1)\\
&= \frac{(d-i+2)\cdot(d-i+3)\cdot\ldots\cdot(d-1)\cdot d\cdot(d+1)}{2\cdot 3\cdot\ldots\cdot i}\\
&= |\lambda_i^{(d+1)}|\,.
\end{aligned}$$

Thus, the recursion formula

$$|\lambda_i^{(d+1)}| = |\lambda_{i-1}^{(d)}| + |\lambda_i^{(d)}| \tag{7}$$

follows. Because of this recursion formula and the initial values of Table 1 the following theorem is proven:

Theorem 1. *The coefficients of Lagrange's interpolation formula with support abscissas $i = 1, 2, 3, \ldots, d$ as given by Equation (6) are integers.*

Please note that for non-equidistant support abscissas the coefficients of Lagrange's interpolation formula are usually fractions. Theorem 1 has the consequence that the reduced coefficients as given by Equation (2) can be calculated very easily, because no computation of an inverse is necessary. In order to keep the absolute values of the coefficients low, the reduction should not be done into $\mathbb{Z}_q = \{x \in \mathbb{Z} | 0 \le x < q\}$. Rather, the coefficients should be from $\mathcal{Z}_q := \{x \in \mathbb{Z} | -q/2 < x \le q/2\}$ (cf. [1]). For small values of $d = 2t+1$ this guarantees small absolute values for the coefficients and saves computing time. Table 2 compares the running times of three versions of Step 2 of the multiplication potocol: The first version is given in Figure 1 with coefficients λ_i in the interval $\mathbb{Z}_q$; the second version is designed for small values of n and is presented in [18]; the third version exploits the observations of the present subsection and uses coefficients λ_i from $\mathcal{Z}_q$. All the computations use [14] and are on an *Intel(R) Core(TM)2 Duo CPU T9400 @ 2.53 GHz* for $n = 2t + 1$ and use the GNU Multiple Precision Arithmetic Library.

Clearly, the protocols require the same amount of communication, namely one round. Their communication complexity is $O(dn)$.

$k = 1024$	Reduction to $\mathbb{Z}_q$	[18]	Reduction to $\mathcal{Z}_q$
$n = 2^2 + 1 = 5$	0.020	0.007	0.005
$n = 2^3 + 1 = 9$	0.069	0.032	0.014
$n = 2^5 + 1 = 33$	0.872	1.138	0.150
$n = 2^7 + 1 = 129$	14.850	60.630	4.060
$n = 2^8 + 1 = 257$	64.700	491.660	25.940
$n = 2^9 + 1 = 513$	290.600	4097.200	175.000
$n = 2^{10} + 1 = 1025$	1454.000	35742.000	1328.000
$n = 2^{11} + 1 = 2049$	6298.000	323978.000	6281.000

Table 2. Running times in milliseconds for three versions of Step 2 of the multiplication protocol

5 Applications

Damgård et al. [11] have presented a protocol that computes, in constant rounds and with unconditional security, sharings of the bits of a shared value $a \in \mathbb{Z}_q$ with some prime q. Their protocol works for any linear secret sharing scheme with a multiplication protocol. In particular, this applies to Shamir's secret sharing scheme [22] with the multiplication protocol of Section 3 and its accelerated modifications of Section 4. The complexity of the protocol in [11] is $O(d \log_2 k)$ invocations of the multiplication protocol for the underlying secret sharing scheme, where k is the bit size of q. Clearly, the protocol benefits from any improvement of the multiplication protocol as presented in the preceeding subsections.

The result in [11] immediately implies solutions to other long-standing open problems such as constant-rounds and unconditionally secure protocols for comparing shared numbers, raising a shared number to a shared exponent and reducing a shared number modulo a shared modulus. These techniques enable, for instance, truly practical double auctions. For more details see [2] and [10].

Distributed signature schemes are another application: Distributed versions of the Miller-Rabin primality test [19, 20] can be built from the above mentioned protocols. For details see [1, 5]. This allows the distributed generation of a shared RSA modulus N being the product of two primes or of two safe primes without the need for a trusted dealer. The subsequent distributed generation of shares of the private exponent is much less computationally involved. In particular, Boneh and Franklin [3] and Catalano, Gennaro and Halevi [6] present efficient protocols to accomplish this. One of the main applications of these results is the construction of theshold variants of signature schemes. In such a scheme n parties hold a $(t + 1)$-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 [6], where two such signature schemes are constructed. The first is an appropriate variant of the signature scheme of Gennaro, Halevi and Rabin [12]; the second relies on the signature scheme of Cramer and Shoup [9]. As all these protocols employ distributive multiplication as an essential part, they significantly benefit from the reduction of complexity.

Acknowledgement

The first author is supported by the European Regional Development Fund – Europäischer Fonds für regionale Entwicklung (EFRE).

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: M. Yung (ed.), Advances in Cryptology – CRYPTO 2002, Lecture Notes in Computer Science 2442, pp. 417–432, Springer, Berlin, 2002.
2. P. Bogetoft, I. Damgård, T. Jakobsen, K. Nielsen, J. Pagter, and T. Toft. A practical implementation of secure auctions based on multiparty integer computation. In: 10th International Conference on Financial Cryptography and Data Security – FC 2006, Lecture Notes in Computer Science 4107, 142–147, IFCA/Springer, Berlin, 2006.
3. D. Boneh and M. Franklin, Efficient generation of shared RSA keys, in: Advances in Cryptology – CRYPTO 1997,
4. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the 20th Annual Symposium on Theory of Computing (STOC'88), 1–10, ACM Press, 1988.
5. 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, pp. 1–39, Birkhäuser, Basel, 2005.
6. D. Catalano, R. Gennaro, and S. Halevi. Computing inverses over a shared secret modulus. In: Advances in Cryptology – EUROCRYPT 2000, Lecture Notes in Computer Science 1807, 190–206, Springer, Berlin, 2000.
7. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In: Proceedings of the 20th Annual Symposium on Theory of Computing (STOC'88), 11–19, ACM Press, 1988.
8. R. Cramer and I. Damgård. Multiparty computation, an introduction. In: D. Catalano, R. Cramer, I. Damgård, G. Di Crescenco, D. Pointcheval, T. Takagi (eds.), *Contemporary Cryptology*, Advanced Courses in Mathematics, CRM Barcelona, pp. 1–39, Birkhäuser, Basel, 2005.
9. 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.
10. I. Damgård. Theory and practice of multiparty computation. In: Proceedings of the 6th Conference on Security and Cryptography for Networks (SCN'2006), Lecture Notes in Computer Science 4116, 360–364, Springer, Berlin, 2006.
11. I. Damgård, M. Fitzi, E. Kiltz, J. Nielsen, and T. Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentation. In: Proceedings of the 3rd Theory of Cryptography Conference (TCC'2006), Lecture Notes in Computer Science, 3876, 285–304, Springer, Berlin, 2006.
12. 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.
13. R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In: B. Coan and Y. Afek (eds.), Proceedings of the 17th ACM Symposium on Principles of Distributed Computing (PODC'98), pp. 101–111, ACM Press, 1998.

14. The GNU Multiple Precision Arithmetic Library, Edition 4.3.2, 2009, `http://gmplib.org`.
15. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In: Proceedings of the 19th Annual Symposium on Theory of Computing (STOC'87), 218–229, ACM Press, 1987.
16. D. E. Knuth. *The Art of Computer Programming*. Volume 2: Seminumerical Algorithms, Addison-Wesley, Reading, 1971.
17. P. Lory. Reducing the complexity in the distributed multiplication protocol of two polynomially shared values. In: C. Rong and X. Chu (eds.), Proceedings of the 3rd IEEE International Symposium on Security in Networks and Distributed Systems (SSNDS'2007), volume 1 of AINA'2007, pp. 404–408, IEEE Computer Society, 2007.
18. P. Lory. Secure distributed multiplication of two polynomially shared values: Enhancing the efficiency of the protocol. In: R. Falk, W. Goudalo, E. Chen, R. Savola, and M. Popescu (eds.), Proceedings of the 3rd International Conference on Emerging Security Information Systems and Technologies (SECURWARE 2009), pp. 287–291, IEEE Computer Society, 2009.
19. G. L. Miller, Riemann's hypothesis and tests for primality, Journal of Computers and System Sciences, 13:30–317, 1976.
20. M. O. Rabin, Probabilistic algorithms for testing primality, Journal of Number Theory, 12:128–138, 1980.
21. 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.
22. A. Shamir. How to share a secret. Communications of the ACM, 22(11), pp. 612–613, 1979.
23. J. Wenzl. Laufzeitanalyse dreier Versionen eines Mehrparteien-Multiplikationsprotokolls. University of Regensburg Working Papers in Business, Economics and Management Information Systems, Nr. 440, 2010.
24. A. C. Yao. How to generate and exchange secrets. In: Proceedings of the 27th IEEE Symposium on Foundations of Computer Science (FOCS'86), 162–167, IEEE Computer Society, 1986.