Efficient Multiparty Protocols Using Generalized Parseval's Identity and the Theta Algebra

Giorgio Sonnino
Université Libre de Bruxelles (ULB)
Brussels, Belgium
giorgio.sonnino@ulb.be

Alberto Sonnino
Mysten Labs
London, U.K.
alberto@mystenlabs.com

***Abstract*—We propose a protocol able to show publicly addition *and* multiplication on secretly shared values. To this aim, we developed a protocol based on the use of masks and FMPC (Fourier Multi-Party Computation). FMPC is a novel multiparty computation protocol of arithmetic circuits based on secret-sharing, capable to compute the addition and multiplication of secrets with no communication. We achieve this task by introducing the first generalization of Parseval's identity for Fourier series applicable to an arbitrary number of inputs and a new algebra referred to as the $\Theta^{[n]}$-*algebra*. FMPC operates in a setting where users wish to compute a function over some secret inputs by submitting the computation to a set of nodes, without revealing those inputs. FMPC offloads most of the computational complexity to the end users and includes an online phase that mainly consists of each node locally evaluating specific functions. FMPC paves the way for a new kind of multiparty computation protocol; making it possible to compute the addition and multiplication of secrets stepping away from circuit garbling and the traditional algebra introduced by Donald Beaver in 1991. Our protocol is capable to compute addition and multiplication with no communication and its simplicity provides efficiency and ease of implementation.**

***Index Terms*—Cryptography, Multiparty Protocols, Multi-Party Computation**

I. Introduction

MPC (Multi-Party Computations) are cryptographic protocols where several distinct, yet connected, computing devices (or parties) jointly evaluate a public function while preserving several security properties despite adversarial behavior [DPSZ12]. This work aims to solve the following problem: *Develops a method able to show publicly the result of a general mathematical expression while keeping the inputs of the expression secret*. This problem must be solved by satisfying the following conditions:

1) Users are indistinguishable from each other;
2) Users cannot communicate with each other;
3) All operations must be performed simultaneously for all users;
4) The number of nodes that are not corrupted must not depend on the number of nodes involved in the process;
5) The operations performed by the display are visible to the public.

It is trivial to solve the problem when the mathematical expression is composed only of sums or only by multiplications of (secret) codes. Indeed, in the first case (i.e., the expression is made up of sums of the codes only) it is sufficient that each user splits the codes into two contributions and sends them to two distinct nodes that cannot communicate with each other, according to the following procedure. Denoting with a_j the secret code of the user j, with $j = 1, \cdots, n$, each user splits its code in two parts, $a_j^{(1)}$ and $a_j^{(2)}$, such that $a_j = a_j^{(1)} + a_j^{(2)} \ \ \forall j$. Successively, they send the contribution $a_j^{(1)}$ to the $Node_1$ and the contribution $a_j^{(2)}$ to the node $Node_2$, respectively. $Node_1$ and $Node_2$ perform the partial sum $S^{(1)} = \sum_{j=1}^{n} a_j^{(1)}$ and $S^{(2)} = \sum_{j=1}^{n} a_j^{(2)}$, respectively. Finally, the *Display* shows publicly the value of the sum S with $S = S^{(1)} + S^{(2)}$ (see Figure 1). Similarly, when the expression is made up of products of the codes only, it is sufficient that each user splits its code in two parts, $a_j^{(1)}$ and $a_j^{(2)}$, such that $a_j = a_j^{(1)} \cdot a_j^{(2)} \ \ \forall j$. Successively, they send the contribution $a_j^{(1)}$ to the $Node_1$ and the contribution $a_j^{(2)}$ to the node $Node_2$, respectively. $Node_1$ and $Node_2$ perform the partial product $P^{(1)} = \Pi_{j=1}^{n} a_j^{(1)}$ and $P^{(2)} = \Pi_{j=1}^{n} a_j^{(2)}$, respectively. Finally, the *Display* shows publicly the value of the product P with $P = P^{(1)} \cdot P^{(2)}$ (see Figure 2):

Another very efficient way to solve the previous problem is based on the use of numerical *masks*.

As for the sum of the codes, the users chose *additive masks* ω_j ($j = 1, \cdots, n$) and split the code in two parts: $a_j^{(1)} = a_j + \omega_j$ and $a_j^{(2)} = -\omega_j$ and they send $a_j^{(1)}$ and $a_j^{(2)}$ to the the $Node_1$ and $Node_2$, respectively. Successively, $Node_1$ and $Node_2$ perform the partial sums $S^{(1)} = \sum_{j=1}^{n} a_j^{(1)}$ and $S^{(2)} = \sum_{j=1}^{n} a_j^{(2)}$, respectively. Finally, the *Display* shows publicly the value of the sum S with $S = S^{(1)} + S^{(2)}$.

When the expression is made only by a multiplication of the codes, the users chose the *multiplicative masks* ω_j and $\widetilde{\omega}_j$ ($j = 1, \cdots, n$) such that $|a_j|\omega_j\widetilde{\omega}_j = a_j$ with $|a_j|$ denoting the absolute value (i.e., the modulus) of the code a_j[1]. Successively, the users send the values of $|a_j|\omega_j$ and $\widetilde{\omega}_j$ to $Node_1$ and $Node_2$, respectively. $Node_1$ and $Node_2$ perform the partial product $P^{(1)} = \Pi_{j=1}^{n}|a_j|\omega_j$ and $P^{(2)} = \Pi_{j=1}^{n}\widetilde{\omega}_j$, respectively. Finally, the *Display* shows publicly the numerical value of the product P with $P = P^{(1)} \cdot P^{(2)}$.

[1]Notice that by choosing the masks ω_j in such a way that $|a_j|\omega_j\widetilde{\omega}_j = a_j$, it is not possible to determine neither the value nor the sign of the code a_j

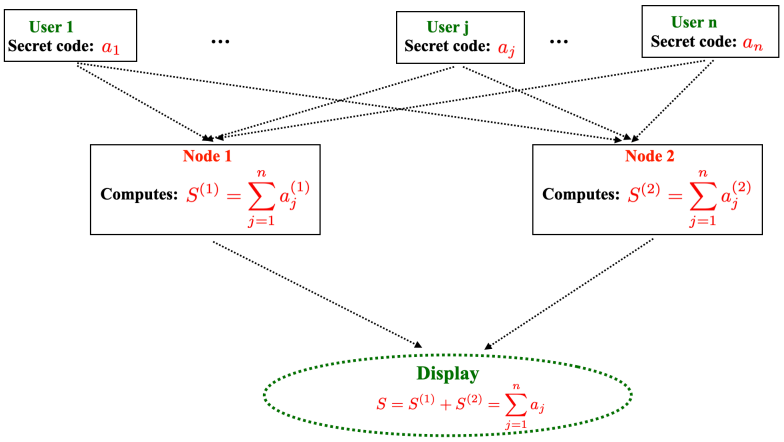

Fig. 1. **Display of the expression $S = \sum_{j=1}^{n} a_j$ by keeping secret the codes a_j.** This scheme illustrates how an expression, composed of a sum of secret codes only, can be shown publicly by keeping secret the codes of single users. The users split their code in two pieces $a_j^{(1)}$ and $a_j^{(2)}$. The values of $a_j^{(1)}$ and $a_j^{(2)}$ may be chosen such that $a_j = a_j^{(1)} a_j^{(2)}$ or, by using additive masks ω_j, $a_j^{(1)} \equiv a_j + \omega_j$ and $a_j^{(2)} \equiv -\omega_j$. At least, one node is not corrupted.

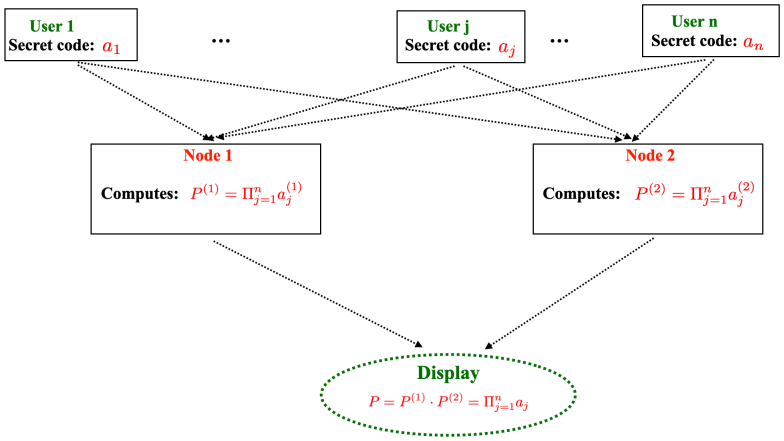

Fig. 2. **Display of the expression $P = \Pi_{j=1}^{n} a_j$ by keeping secret the codes a_j.** This scheme illustrates how an expression composed only by a product of secret codes can be shown publicly by keeping secret the codes of single users. The users split their code in two pieces $a_j^{(1)}$ and $a_j^{(2)}$. The values of $a_j^{(1)}$ and $a_j^{(2)}$ may be chosen such that $a_j = a_j^{(1)} a_j^{(2)}$ or, by using multiplicative masks ω_j, $a_j^{(1))} \equiv |a_j|\omega_j$ and $a_j^{(2))} \equiv \widetilde{\omega}_j$ with $\widetilde{\omega}_j$ defined such that $a_j = |a_j|\omega_j\widetilde{\omega}_j$.At least one node is not corrupted.

Another important problem that we have to solve is the determination of a procedure able to use, instead of two nodes, an arbitrary number of nodes N, *while keeping constant the number of nodes that must not be corrupted*. It is customary to enumerate the nodes with $N = 3f + 1$ with f denoting a natural number (i.e., $f = 1, 2, \cdots$). It is easy to set up this procedure when the mathematical expression is made only by sums of the codes or products of the codes. Indeed, it is sufficient for each user to divide their code into $2f+1$ pieces:

$$a_j = \sum_{i=1}^{3f+1} a_j^{(i)} \quad \text{in case of sums}$$

$$a_j = \Pi_{i=1}^{3f+1} a_j^{(i)} \quad \text{in case of products}$$

and then send each piece to a different node. The numerical value of the expression can be shown publicly without revealing the secrets provided that at least one node is not corrupted. Of course, the same procedure applies when using *masks*.

To sum up when the expression contains either sums or products of the (secret) codes, the problem of showing publicly the numerical value of this expression is trivially solved. The issue arises when the expression is composed of a combination of sums and products of codes. This work aims to present a protocol able to show publicly addition *and* multiplication on secretly shared values. This problem has already been solved by Donald Beaver in 1991 by using traditional algebra - essentially based on the number theory - and a wide series of circuits garbling [Bea91]. Unfortunately, Beaver's protocol requires many rounds of communication and this slows down the machine time with consequent expensive energy. To overcome these obstacles we resort to a more sophisticated mathematical approach, based on the use of masks

and FMPC (Fourier-based Multi-Party Computation) [Son19]. Our protocol operates by using the masks' method and paves the way for a new kind of multiparty computation protocol capable of computing sums and multiplications of secrets as an alternative to circuit garbling. More precisely, we hide the users' input codes within the cosine components of the Fourier series of the *main function* (denoted by $f(x)$), combined with *additive masks* that can be chosen arbitrarily by the users. The result of the function is visible publicly but, of course, the user's mask is kept secret. Users send the secret codes hidden by the masks to four nodes. Computations are performed by using a new algebra, referred to as the $\Theta^{[n]}$-*algebra* (or *Theta-algebra*). Finally, the results of the calculation are transmitted to the display which, thanks to the *generalized Parseval's identity for Fourier series* applicable to an arbitrary number of inputs, can show publicly the numerical result of the mathematical expression while keeping secret the users' codes. The participants' privacy is guaranteed if at least three nodes are not corrupted. Successively, we solved the problem using $3f+1$ nodes, grouped in four *categories*. In this case users, besides the additive mask, arbitrarily choose another *multiplicative mask*. Even for multiple nodes, the participants' privacy is guaranteed if at least three nodes belonging to three different categories are not corrupted and at least one node, belonging to the *second level* of computation, is not corrupted. Compared to the original Beaver's method, our protocol is capable to compute addition and multiplication with *no online communication* and its simplicity provides efficiency and ease of implementation.

The manuscript is organized as follows. Section II presents a protocol capable to show publicly addition *and* multiplication for two players, This task is accomplished by using additive masks and FMPC. Section III is devoted to the application of the theorems valid for the Chebyshev polynomials. We shall see that these theorems allow showing a general mathematical expression while keeping secret the users' codes. The mathematical framework to be used for treating the case of n players is described in Section IV. In Section V we treat the problem for n players by using the generalized Parseval's identity and the $\Theta^{[n]}$-*algebra*. Finally, in Section VI we solve the problem for $3f+1$ nodes, grouped in four *categories*. In this case, users need to choose, arbitrarily, multiplicative as well as additive masks. The generalized Parseval's identity and the use of the *Theta-algebra* ensure the correct result shown by the display. Section VII presents designs that incorporate our nodes within the infrastructure of several semi-permissioned Blockchains. Limitations of our work and concluding remarks can be found in Section IX and Section X, respectively. Useful relations for getting the values of the infinite sums, the proof of the generalized Parseval's identity, and the rules for the $\Theta^{[n]}$-*algebra* are reported in Appendices A, A, and A, respectively.

II. Display of the expression $x_1a + x_2b + yab$ by keeping secret the codes a and b.

A. Background

We recall the Fourier series of the convolution between two functions $\phi(x)$ and $\psi(x)$ periodic on $(-l,\ l)$. Assuming that $\phi(x)$ and $\psi(x) \in \mathbb{L}^2[-l,l]$ ($\phi(x)$ and $\psi(x)$ are square-integrable in the interval $[-l,\ l]$), their respective Fourier series representations read:

$$\phi(x) = \frac{\alpha^{(0)}}{2} + \sum_{m=1}^{+\infty} \alpha_m \cos\Big(\frac{n\pi x}{l}\Big) + \sum_{m=1}^{+\infty} \beta_m \sin\Big(\frac{n\pi x}{l}\Big)$$
$$\psi(x) = \frac{a^{(0)}}{2} + \sum_{m=1}^{+\infty} a_m \cos\Big(\frac{n\pi x}{l}\Big) + \sum_{m=1}^{+\infty} b_m \sin\Big(\frac{n\pi x}{l}\Big)$$

where the Fourier coefficients $(\alpha^{(0)},\ \alpha_m,\ \beta_m)$ and $(a^{(0)},\ a_m,\ b_m)$ (for $m = 1, 2, \cdots$) are given below:

$$\alpha^{(0)} = \frac{1}{l}\int_{-l}^{l} \phi(x)dx\ ; \quad \alpha_m = \frac{1}{l}\int_{-l}^{l} \phi(x)\cos\Big(\frac{m\pi x}{l}\Big)dx$$
$$\beta_m = \frac{1}{l}\int_{-l}^{l} \phi(x)\sin\Big(\frac{m\pi x}{l}\Big)dx$$
$$a^{(0)} = \frac{1}{l}\int_{-l}^{l} \psi(x)dx\ ; \quad a_m = \frac{1}{l}\int_{-l}^{l} \psi(x)\cos\Big(\frac{m\pi x}{l}\Big)dx$$
$$b_m = \frac{1}{l}\int_{-l}^{l} \psi(x)\sin\Big(\frac{m\pi x}{l}\Big)dx$$

Parseval's identity holds for $f(x)$ and $g(x)$ [GR14]:

$$\Big(\frac{a_0\alpha_0}{2} + \sum_{m=1}^{\infty} a_m\alpha_m\Big) + \Big(\sum_{m=1}^{\infty} b_m\beta_m\Big) = \frac{1}{l}\int_{-l}^{l} \phi(x)\psi(x)dx \tag{1}$$

We introduce the *normalised functions* $f(x) = \eta^{-1/2}\phi(x)$ and $g(x) = \eta^{-1/2}\psi(x)$ with

$$\eta = \frac{1}{l}\int_{-l}^{l} \phi(x)\psi(x)dx \tag{2}$$

For functions $f(x)$ and $g(x)$, the Parseval identity reads:

$$\Big(\frac{\widetilde{a}_0\widetilde{\alpha}_0}{2} + \sum_{m=1}^{\infty} \widetilde{a}_m\widetilde{\alpha}_m\Big) + \Big(\sum_{m=1}^{\infty} \widetilde{b}_m\widetilde{\beta}_n\Big) = 1 \tag{3}$$

with $\{\widetilde{\alpha}_0, \widetilde{\alpha}_n, \widetilde{\beta}_n\}$ and $\{\widetilde{a}_0, \widetilde{a}_n, \widetilde{b}_n\}$ denoting the Fourier coefficients of $f(x)$ and $g(x)$, respectively. Parseval's identity only applies to two functions. Section IV-A presents our generalization of Parseval's identity that applies to an arbitrary number of functions.

Now, we start by solving the case of two players called *Alice* and *Bob*. To this aim, we choose $f(x) = g(x)$. Furthermore, for simplicity, we consider an even main function $f(x)$ i.e., $f(x) = f(-x)$ (so, $\widetilde{\beta}_m = 0$). In this case, Parseval's identity reduces to:

$$\frac{\widetilde{\alpha}_0^2}{2} + \sum_{m=1}^{\infty} \widetilde{\alpha}_m^2 = 1 \quad \text{and} \quad \eta = \frac{1}{l}\int_{-l}^{l} \phi(x)^2dx \tag{4}$$

In the sequel, in order not to burden the notations the tilde over the Fourier coefficients will be omitted being understood that the main functions are normalized. In the following Subsections, we establish the tasks of Alice and Bob.

B. Tasks of Alice

1) Alice splits her secret code in four parts: $x_1 a = a_1 + a_2 + a_3 + a_4$; .
2) Alice choses two musks, by choosing *arbitrary pairs of parameters*[2] : $\omega_{1,m} = a_{1,m} + ib_{1,m}$ and $\omega_1^{(0)} = a_1^{(0)} + ib_1^{(0)}$;
3) Let us call $(\alpha^{(0)}, \alpha_m)$ the cosine Fourier components of the *main function* $f(x)$. We define $(\alpha_1^{(0)}, \alpha_{1,m}) \equiv (|y|^{1/2} a\alpha^{(0)}, |y|^{1/2} a\alpha_m)$;
4) Alice constructs the four hyper-vectors $A_1^{(1)}, A_1^{(2)}, B_1^{(1)}, B_1^{(2)}$, defined as

$$\begin{aligned} A_1^{(1)} &\equiv \left\{ a_1, \alpha_1^{(0)} + \omega_1^{(0)}, \alpha_{1,m} + \omega_{1,m} \right\} \\ B_1^{(1)} &\equiv \left\{ a_3, \alpha_1^{(0)} + i\omega_1^{(0)}, \alpha_{1,m} + i\omega_{1,m} \right\} \\ A_1^{(2)} &\equiv \left\{ a_2, \alpha_1^{(0)} - \omega_1^{(0)}, \alpha_{1,m} - \omega_{1,m} \right\} \\ B_1^{(2)} &\equiv \left\{ a_4, \alpha_1^{(0)} - i\omega_1^{(0)}, \alpha_{1,m} - i\omega_{1,m} \right\} \end{aligned}$$

5) Alice sends the *hyper-vectors* $A_1^{(1)}$ and $A_1^{(2)}$ to the *node 1* and the*node 2*, respectively, and the hyper-vectors $B_1^{(1)}$ and $B_1^{(2)}$ to the *node 3* and *nde 4*, respectively.

C. Tasks of Bob

1) Bob splits his secret code in four parts: $x_2 b = b_1 + b_2 + b_3 + b_4$; .
2) Bob choses two musks, by choosing *arbitrary pairs of parameters*[3]: $\omega_{2,m} = a_{2,m} + ib_{2,m}$ and $\omega_2^{(0)} = a_2^{(0)} + ib_2^{(0)}$; .
3) With the cosine Fourier components of the main function $f(x)$, i.e. $(\alpha^{(0)}, \alpha_m)$, we define $(\alpha_2^{(0)}, \alpha_{2,m}) \equiv (|y|^{1/2} b\alpha^{(0)}, |y|^{1/2} b\alpha_m)$;
4) Bob constructs the four *hyper-vectors* $A_2^{(1)}, A_2^{(2)}, B_2^{(1)}, B_2^{(2)}$, defined as

$$\begin{aligned} A_2^{(1)} &\equiv \left\{ b_1, \alpha_2^{(0)} + \omega_2^{(0)}, \alpha_{2,m} + \omega_{2,m} \right\} \\ B_2^{(1)} &\equiv \left\{ b_3, \alpha_2^{(0)} + i\omega_2^{(0)}, \alpha_{2,m} + i\omega_{2,m} \right\} \\ A_2^{(2)} &\equiv \left\{ b_2, \alpha_2^{(0)} - \omega_2^{(0)}, \alpha_{2,m} - \omega_{2,m} \right\} \\ B_2^{(2)} &\equiv \left\{ b_4, \alpha_2^{(0)} - i\omega_2^{(0)}, \alpha_{2,m} - i\omega_{2,m} \right\} \end{aligned}$$

5) Bob sends the hyper-vectors $A_2^{(1)}$ and $A_2^{(2)}$ to the *node 1* and the*node 2*, respectively, and the hyper-vectors $B_2^{(1)}$ and $B_2^{(2)}$ to the *node 3* and *node 4*, respectively.

[2]It is convenient to choose $\omega_1^{(0)} = \alpha^{(0)}(a_1^{(0)} + ib_1^{(0)})$ and $\omega_{1,m} = (a_1 + ib_1)\alpha_m$.

[3]It is convenient to choose $\omega_2^{(0)} = \alpha^{(0)}(a_2^{(0)} + ib_2^{(0)})$ and $\omega_{2,m} = (a_2 + ib_2)\alpha_m$.

D. Tasks of the Nodes

The four nodes perform the following tasks. Note that the sign is $+$ if $y > 0$ and $-$ if $y < 0$:
Node 1 computes:

$$\begin{aligned} N_1 = a_1 + b_1 \pm \Big(&\frac{1}{8}(\alpha_1^{(0)} + \omega_1^{(0)})(\alpha_2^{(0)} + \omega_2^{(0)}) \\ &+ \frac{1}{4}\sum_{m=1}^{\infty}(\alpha_{1,m} + \omega_{1,m})(\alpha_{2,m} + \omega_{2,m}) \Big) \end{aligned} \tag{5}$$

Node 2 computes

$$\begin{aligned} N_2 = a_2 + b_2 \pm \Big(&\frac{1}{8}((\alpha_1^{(0)} - \omega_1^{(0)})(\alpha_2^{(0)} - \omega_2^{(0)})) \\ &+ \frac{1}{4}\sum_{m=1}^{\infty}((\alpha_{1,m} - \omega_{1,m})(\alpha_{2,m} - \omega_{2,m})) \Big) \end{aligned} \tag{6}$$

Node 3 computes:

$$\begin{aligned} N_3 = a_3 + b_3 \pm \Big(&\frac{1}{8}((\alpha_1^{(0)} + i\omega_1^{(0)})(\alpha_2^{(0)} + i\omega_2^{(0)})) \\ &+ \frac{1}{4}\sum_{m=1}^{\infty}((\alpha_{1,m} + i\omega_{1,m})(\alpha_{2,m} + i\omega_{2,m})) \Big) \end{aligned} \tag{7}$$

Node 4 computes:

$$\begin{aligned} N_4 = a_4 + b_4 \pm \Big(&\frac{1}{8}((\alpha_1^{(0)} - i\omega_1^{(0)})(\alpha_2^{(0)} - i\omega_2^{(0)})) \\ &+ \frac{1}{4}\sum_{m=1}^{\infty}((\alpha_{1,m} - i\omega_{1,m})(\alpha_{2,m} - i\omega_{2,m})) \Big) \end{aligned} \tag{8}$$

E. Task of the Display

The display shows:

$$\begin{aligned} N_1 + N_2 + N_3 + N_4 &= a_1 + a_2 + a_3 + a_4 + b_1 + b_2 + b_3 \\ &\quad + b_4 \pm \frac{1}{2}\alpha_1^{(0)}\alpha_2^{(0)} \pm \sum_{m=1}^{\infty}\alpha_{1,m}\alpha_{2,m} \\ &= x_1 a + x_2 b + yab \end{aligned} \tag{9}$$

due to *Parseval's identity* (4). Here, at least three nodes must not be corrupted. Fig. (3) illustrates the procedure.

F. Example: $Expr = 3a + 5b - 9ab$

Let us apply the procedure to a concrete example where $(x_1, x_2, y) = (3, 5, 9)$. This triad of numbers is shown publicly. We suppose that the secret codes of Alice and Bob are $(a, b) = (2.2, 4.1)$, so the value of the expression shown by the display is $Expr = -54.08$. First, we have to choose the main function $f(x)$. We choose, for example, $f(x) = \left(1 + \frac{\sin(2\pi\tau)}{2\pi\tau}\right)^{-1/2} \cos(\pi\tau x/L)$. So, the Fourier coefficients read

$$\begin{aligned} \alpha^{(0)} &= 2\left(1 + \frac{\sin(2\pi\tau)}{2\pi\tau}\right)^{-1/2} \frac{\sin(\pi\tau)}{\pi\tau} \\ \alpha^{(m)} &= \alpha^{(0)}\tau^2 \frac{(-1)^m}{\tau^2 - m^2} \end{aligned} \tag{10}$$

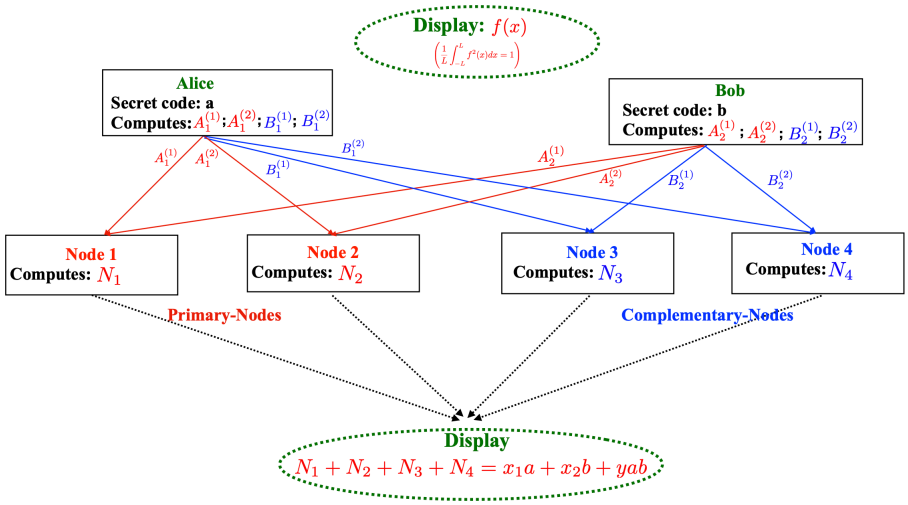

Fig. 3. **Display of the expression $x_1 a + x_2 b + yab$ by keeping secret the code a and b.** This procedure allows showing publicly the expression by keeping secret the codes of Alice and Bob. In this case, at least three nodes must not be corrupted

If the system chooses, for example, $\tau = 1/6$, we get

$$\alpha^{(0)} = 6\sqrt{\frac{2}{3\sqrt{3}\pi + 2\pi^2}} \qquad (11)$$
$$\alpha_m = 6\sqrt{\frac{2}{3\sqrt{3}\pi + 2\pi^2}}\frac{(-1)^m}{1-36m^2}$$

It is useful to take into account the following identity [GR14][4]

$$\sum_{m=1}^{\infty}(\alpha_m)^2 = \frac{-36 + 3\sqrt{3}\pi + 2\pi^2}{\pi(3\sqrt{3} + 2\pi)}$$

1) Tasks of Alice: Alice possesses the code $a = 2.2$. She splits her ode in four pieces $a_1 = 3.3$; $a_2 = 1.65$ $a_3 = 1.32$; $a_4 = 0.33$ and chooses the following masks

$$\omega_1^{(0)} = \alpha^{(0)}(7 + 9i) \quad ; \quad \omega_1 = (2 + 11i)\alpha_m$$

2) Tasks of Bob: Bob possesses the code $a = 4.1$. He splits his code in four pieces $b_1 = 3.41667$; $b_2 = 2.05$ $b_3 = 5.125$; $b_4 = 9.90833$ and chooses the following masks

$$\omega_2^{(0)} = \alpha^{(0)}(5 + 3i) \quad ; \quad \omega_2 = (4 + 8i)\alpha_m$$

3) Tasks of the Nodes: The nodes perform the following tasks:

Node 1:

$$\begin{aligned} N_1 = & a_1 + b_1 - \frac{1}{8}(\alpha_1^{(0)} + \omega_1^{(0)})(\alpha_2^{(0)} + \omega_2^{(0)}) \\ & + \frac{1}{4}\sum_{m=1}^{\infty}(\alpha_{1,m} + \omega_{1,m})(\alpha_{2,m} + \omega_{2,m}) \\ = & 3.3 + 3.41667 - 1/8(\alpha^{(0)})^2(13.6 + 9i)(17.3 + 3i) \\ & - 1/4(8.6 + 11i)(16.3 + 8i)\sum_{m=1}^{\infty}(\alpha_m)^2 \end{aligned} \qquad (12)$$

[4]Useful relations for getting the values of the infinite sums can be found in Appendix A.

Node 2:

$$\begin{aligned} N_2 = & a_2 + b_2 - \left(\frac{1}{8}((\alpha_1^{(0)} - \omega_1^{(0)})(\alpha_2^{(0)} - \omega_2^{(0)})) \right. \\ & \left. + \frac{1}{4}\sum_{m=1}^{\infty}((\alpha_{1,m} - \omega_{1,m})(\alpha_{2,m} - \omega_{2,m}))\right) \\ = & 11.1887 + 16.153i \end{aligned} \qquad (13)$$

Node 3:

$$\begin{aligned} N_3 = & a_3 + b_3 - \left(\frac{1}{8}((\alpha_1^{(0)} + i\omega_1^{(0)})(\alpha_2^{(0)} + i\omega_2^{(0)})) \right. \\ & \left. + \frac{1}{4}\sum_{m=1}^{\infty}((\alpha_{1,m} + i\omega_{1,m})(\alpha_{2,m} + i\omega_{2,m}))\right) \\ = & 20.7616 - 13.2477i \end{aligned} \qquad (14)$$

Node 4:

$$\begin{aligned} N_4 = & a_4 + b_4 - \left(\frac{1}{8}((\alpha_1^{(0)} - i\omega_1^{(0)})(\alpha_2^{(0)} - i\omega_2^{(0)})) \right. \\ & \left. + \frac{1}{4}\sum_{m=1}^{\infty}((\alpha_{1,m} - i\omega_{1,m})(\alpha_{2,m} - i\omega_{2,m}))\right) \\ = & -40.7457 + 46.2424i \end{aligned} \qquad (15)$$

4) Tasks of the Display: The display shows publicly:

$$Expr = N_1 + N_2 + N_3 + N_4 = -54.08 + 7.10543 \cdot 10^{-15}i$$

due to *Parseval's identity* (4).

III. Showing a general mathematical expression by keeping secret the codes a and b

It is easy to convince ourselves that the method of the masks may successfully be applied to get a wide variety of mathematical expressions, by keeping secret the code of the players and without having to resort to any mathematical approximation. For example, it is possible to get the value of

$$\text{Expr.} = \arctan\log\sin(a^2 + 3ab - 4/b)$$

without having to approximate it with polynomial interpolations (or another kind of interpolations). The strategy is *to mask* the terms and ask the Display to show the value of the mathematical expression. In the above case, 3 terms must be masked through the use of 3 masks, one provided by player a and two provided by player b, for getting the expressions:

$$arg. = a^2 + 3ab - 4/bi$$

and request that the display publicly shows the value of the final expression:

$$\text{Expr.} = \arctan\log\sin(arg.)$$

Of course, the above considerations apply to any mathematical expression, of the type:

$$\text{Expr.} = f(arg.)$$

where f is a publicly visible function. However, in several practical applications, it is not possible to dispose of the exact mathematical expression of a variable, but only a discrete set of experimental sampling of it. In these cases, the method of the mask functions can still be successfully used. The *musk-technique* is able to evaluate the following expression, by keeping secret the codes a and b:

$$\text{Expr.} = w_1(a) + w_2(b) + \sum_{r,p=0}^{m,m'} c_{rp} g_r(a) h_p(b) \qquad (16)$$

where w_1, w_2, g_r and h_p are functions and m and m' (finite) integers, respectively. c_{lp} are constant coefficients. In this case, both Alice and Bob have to split functions w_1, and w_2 into two parts and they have to construct m and m' mask functions, respectively. As known, any regular function defined between -1 and $+1$ can be approximated, with high precision, considering only a few terms of the *Chebyshev polynomials*. This observation allows to approximate any cryptographic expression with the first (no more than, let's say the first nine) polynomials of *Chebyshev* and, then, to apply the musk-functions technique to the expression obtained with the *Chebyshev polynomials*. Note that the accurate convergence of the *Chebyshev polynomials* is guaranteed in the interval of the type $-l$ and $+l$ (if the polynomials are orthonormalized). l is an arbitrary parameter that does not participate in the encryption procedure. So, it may be chooses $l = 1$. Thanks to *Chebyshev polynomials*, the general expression may be brought into the form (16). This will be more precise in the forthcoming sections.

A. Properties and Theorems on the Chebyshev Polynomials

It is useful to recall the two main *properties* and *theorems* on the Chebyshev Polynomials[5].

A) *Properties of the Chebyshev polynomials*

i) The Chebyshev polynomials $T_m(x)$ form a complete orthogonal system.

ii) The Chebyshev series converges to $\Psi(x)$ if the function is piecewise smooth and continuous. The smoothness requirement can be relaxed in most cases – as long as there are a finite number of discontinuities in $\Psi(x)$ and its derivatives.

iii) At a discontinuity, the series will converge to the average of the right and left limits.

B) *Theorems on the Chebyshev polynomials*

Theorem 1: Accuracy

If we want polynomial interpolating a function f at $m+1$ points x_s in the interval $[-1, 1]$ to be as accurate as possible, then we should choose the data x_s so that they are the zeros of the Chebyshev polynomial $T_{m+1}(x)$. More specifically, *If the nodes x_s are chosen as the roots of the Chebyshev polynomial $T_{m+1}(x)$*

$$x_s = \cos\left(\frac{2s+1}{2m+2}\pi\right) \quad ; \quad (s = 0, 1 \ldots, m)$$

then the error term for polynomial interpolation using the nodes x_s is

$$E(x) = |f(x) - P(x)| \leq \frac{1}{2^m(m+1)!}\max_{(-1\leq t\leq 1)}|f^{m+1}(t)|$$

Moreover, this is the best upper bound we can achieve by varying the choice of x_s.

Theorem 2: Convergence

The Chebyshev Numerical Method is best with the Rate of Convergence.

The different numerical methods have different rates of convergence. And the rate of convergence is a very important issue in the solution of polynomial and transcendental equations because the rate of convergence of any numerical method determines the speed of the approximation to the solution of the problem. The following table shows the comparison of the rate of convergence. Among Secant, *Regula-Falsi* and *Newton-Raphson* which are based on 1st-degree equations Newton-Raphson has a good rate of convergence i.e. 2. Among *Muller* and *Chebyshev methods*, which are based on 2^{nd} degree equations, Chebyshev is best with the rate of convergence of 3. Table I compares the rate of convergence of various methods.

B. Interpolation of a General Expression $Expr.(a, b)$ with the Chebyshev Polynomials

Let $Expr.(a, b) = w_1(a) + w_2(b) + \Phi(a, b)$ be a general function of two variables a and b. We may interpolate this expression by proceeding in the following manner:

[5]A deep and exhaustive analysis on the Chebyshev polynomials can be found, for instance, in [MH02].

Method	Based on Equation	Rate of Convergence
Secant	1^{st} degree	1.618
Regula-Falsi	1^{st} degree	1
Newton-Raphson	1^{st} degree	2
Muller	2^{st} degree	1.84
Chebyshev	2^{st} degree	3

TABLE I
RATE OF CONVERGENCE OF VARIOUS METHODS

i) We develop the expression $\Phi(a,b)$ in terms of Chebyshev polynomials $T_r(x)$ with respect to the variable a:

$$\Phi(a,b) \simeq \sum_{r=0}^{m} T_r(a) c_r(b)$$

ii) Successively, we develop the expressions $c_r(b)$ in terms of Chebyshev polynomials $T_p(x)$ with respect to the variable b:

$$c_r(b) \simeq \sum_{p=0}^{m'} c_{rp} T_p(b)$$

Finally, we get

$$\text{Expr.} \simeq w_1(a) + w_2(b) + \sum_{r,p=0}^{m,m'} c_{rp} T_r(a) T_p(b) \qquad (17)$$

which is of the same form as Eq. (16). To sum up, the main conclusions of our analysis are:
a) Theorems **1)** and **2)** ensure the *best accuracy* - by quantifying the error - and the *best rate of convergence*;
b) The previous development can trivially be extended when *the expression depends on n variables*.

IV. MATHEMATICAL FRAMEWORK FOR TREATING THE CASE OF MULTIPLE USERS

In the previous Sections, we solved the problem for the case of two users. Unfortunately, the adopted procedure does not trivially extend to the case of n users (with $n > 2$). This is due to the following two drawbacks:
a) Parseval's identity traditionally applies only to two functions. In this case, the mathematical expression of this identity is invariant under the permutation of the two functions; in other words, the two functions are *indistinguishable* each with the other. However, the problem arises when the users are more than two: if we apply this identity in a simple sequential manner, the functions will no longer be indistinguishable from each other. What we need is to derive an expression equivalent to Eq. (1), which applies for $n \geq 2$ functions. In this way, the property of indistinguishability among functions will be preserved. Concretely, we need to derive the expression for the normalisation coefficient η that generalises Eq. (2) for $n \geq 2$ functions, so that Parseval's identity for normalised functions can be cast into the form (3). This task will be accomplished in the next Section.
b) The second obstacle is due to the fact that the algebra of complex numbers, which we used to treat the case $n = 2$, turns out to be inadequate when there are more than two users. Indeed, for $n > 2$ the additive masks no longer cancel each other leading to an erroneous final result. We may easily convince ourselves on this by noticing that for $n = 2$ the masks cancel each with other because $i^2 = -1$, while for $n > 2$ ($n = 4$, for example) we have $i^n = +1$ for $n = 4\iota$, with ι denoting a natural number. In this latter case, the masks instead of canceling each other add up! This issue will be overcome by introducing an appropriate algebra, referred to as *Theta algebra*, which has the property to allow the cancellation of the masks for $n \geq 2$, thus providing the correct final result (see the forthcoming Subsection).

A. Generalisation of Parseval's identity for n functions

In this section, we present the generalization of Parseval's identity for the Fourier series applicable to n inputs. Here, we will limit ourselves only to enunciating the theorem, delegating the proof of the theorem in Appendix A. With the n main functions $\phi^{(j)}(x)$, with $(j = 1, \cdots, n)$, we construct out n *even functions* and n *odd functions*:

$$\begin{aligned} \phi_c^{(j)}(x) &\equiv \frac{1}{2}\Big(\phi^{(j)}(x) + \phi^{(j)}(-x)\Big) \\ \phi_s^{(j)}(x) &\equiv \frac{1}{2}\Big(\phi^{(j)}(x) - \phi^{(j)}(-x)\Big) \end{aligned} \qquad (18)$$

with $(j = 1, 2, \cdots)$. We also introduce the *convolution operation* $(\star)$ between two functions $\phi(x)$ and $\psi(x)$:

$$(\phi \star \psi)(x) = \frac{1}{l}\int_{-l}^{l} \phi(t)\psi(x-t)dt$$

and the *integral transform* $(\wedge)$ of a function $\phi(x)$ by a *kernel function* of two variable $\mathbb{K}(x,t)$, defined as:

$$\begin{aligned} (\phi \wedge \mathbb{K})(x) &= \frac{1}{l}\int_{-l}^{l} \phi(t)\mathbb{K}(x,t)dt \quad \text{with} \\ \mathbb{K}(x,t) &= \sum_{m=1}^{\infty} \cos\Big(\frac{m\pi}{l}x\Big)\sin\Big(\frac{m\pi}{l}t\Big) \\ &= \frac{\sin(\pi t/l)}{2(\cos(\pi x/l) - \cos(\pi t/l))} \end{aligned}$$

Note that the Kernel $\mathbb{K}(x,t)$ should be intended in a *distribution* sense [Die18]. Let us now consider n inputs (main functions) with the Fourier representations

$$\begin{aligned} \phi^{(j)}(x) = \frac{\alpha_0^{(j)}}{2} &+ \sum_{m=1}^{+\infty} \alpha_m^{(j)} \cos\Big(\frac{m\pi x}{l}\Big) \\ &+ \sum_{m=1}^{+\infty} \beta_m^{(j)} \sin\Big(\frac{m\pi x}{l}\Big) \quad \text{with} \quad j \\ &= 1, 2, \cdots, n \end{aligned} \qquad (19)$$

We introduce the following three constants

$$\mathbb{C} = \frac{1}{l}\int_{-l}^{l} \phi_c^{(n)}(x) Input 1_C(x) dx \qquad (20)$$

$$\mathbb{S}_e = \frac{1}{l}\int_{-l}^{l} (\phi_s^{(n-1)} \star \phi_s^{(n)})(x) Input 1_{S_e}(x) dx$$

$$\mathbb{S}_o = -\frac{1}{l}\int_{-l}^{l} (\phi_s^{(n)} \wedge \mathbb{K})(x) Input 1_{S_o}(x) dx$$

where functions $Input1(x)$ are defined as

$$Input1_C(x) \equiv ((\cdots(((\phi_c^{(1)} \star \phi_c^{(2)}) \star \phi_c^{(3)}) \star \phi_c^{(4)}) \star \cdots) \star \phi_c^{(n-1)})(x) \quad (21)$$

$$Input1_{S_e}(x) \equiv ((\phi_s^{(1)} \star \phi_s^{(2)}) \star (\phi_s^{(3)} \star \phi_s^{(4)}) \star \cdots \star (\phi_s^{(n-3)} \star \phi_s^{(n-2)}))(x) \quad (22)$$

$$Input1_{S_o}(x) \equiv ((\phi_s^{(1)} \star \phi_s^{(2)}) \star (\phi_s^{(3)} \star \phi_s^{(4)}) \star \cdots \star (\phi_s^{(n-2)} \star \phi_s^{(n-1)}))(x) \quad (23)$$

Eq. (20) may conveniently be cast into the form (see Appendix (A)):

$$\mathbb{C} = \frac{1}{2}\Pi_{j=1}^n \alpha_0^{(j)} + \sum_{m=1}^{\infty}(\Pi_{j=1}^n \alpha_m^{(j)})$$
$$\mathbb{S}_\kappa = \sum_{m=1}^{\infty}(\Pi_{j=1}^n \beta_m^{(j)}) \quad \text{with}$$
$$\mathbb{S}_\kappa = \begin{cases} \mathbb{S}_e & \text{if n=even number} \\ \mathbb{S}_o & \text{if n=odd number} \end{cases}$$

Hence, the generalised Parseval's identity for n inputs $\phi^{(j)}(x)$ (with $j = 1, 2, \cdots, n$) reads:

$$\mathbb{C} + \mathbb{S}_\kappa \quad (24)$$
$$= \frac{1}{2}\Pi_{j=1}^n \alpha_0^{(j)} + \sum_{m=1}^{\infty}(\Pi_{j=1}^n \alpha_m^{(j)}) + \sum_{m=1}^{\infty}(\Pi_{j=1}^n \beta_m^{(j)})$$
$$\text{with } \mathbb{S}_\kappa = \begin{cases} \mathbb{S}_e & \text{if n=even number} \\ \mathbb{S}_o & \text{if n=odd number} \end{cases}$$

Hence, the *normalisation constant* η is given by

$$\eta = (\mathbb{C} + \mathbb{S}_\kappa)^{-1/n} \text{ with } \mathbb{S}_\kappa = \begin{cases} \mathbb{S}_e & \text{if n=even number} \\ \mathbb{S}_o & \text{if n=odd number} \end{cases} \quad (25)$$

and the *normalised main functions* $f^{(j)}(x)$ read

$$f^{(j)}(x) = \eta\phi^{(j)}(x) \quad \text{with} \quad j = 1, 2, \cdots, n \quad (26)$$

If we use the main functions $f^{(j)}(x)$ the generalised Parseval's identity reads

$$\frac{1}{2}\Pi_{j=1}^n \widetilde{\alpha}_0^{(j)} + \sum_{m=1}^{\infty}(\Pi_{j=1}^n \widetilde{\alpha}_m^{(j)}) + \sum_{m=1}^{\infty}(\Pi_{j=1}^n \widetilde{\beta}_m^{(j)}) = 1 \quad (27)$$

with $\widetilde{\alpha}_0^{(j)}$, $\widetilde{\alpha}_m^{(j)}$, and $\widetilde{\beta}_m^{(j)}$ denoting the Fourier coefficients of $f^{(j)}(x)$. As we can see, users in the generalized Parseval's identity are indistinguishable from each other. The schemes of calculation of the coefficients $\mathbb{C}$, $\mathbb{S}_e$, and $\mathbb{S}_o$ are illustrated in Figure 4, 5 and 6, respectively.

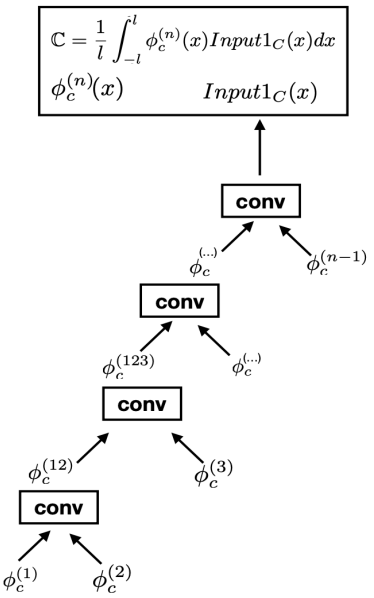

Fig. 4. Illustration of the algorithm for calculating constant $\mathbb{C}$. $n-2$ convolution operations are performed sequentially starting from $\phi_c^{(1)}(x)$ to $\phi_c^{(n-1)}(x)$ i.e. until we get only one function. This latter function has to be integrated with $\phi_c^{(n)}(x)$. Note that, due to the commutative property, the order of the input functions is not relevant in the convolution operations.

B. The $\Theta^{[n]}$-Algebra

We have already noted that the algebra of complex numbers is not adequate to solve the problem when the number of users is greater than two. Indeed, it is easily checked that in case of multiple users we are unable to make the masks disappear, and this is because, according to the algebra of complex numbers, we have

$$+1 = i^4 = i^8 = i^{12} = ... \quad \text{and} \quad -i = i^3 = i^7 = i^{11} = ...$$

So, by raising the imaginary number i to powers of integers, we follow the path illustrated in Figure 7. Our masks must then be constructed by using an algebra where the imaginary number i is replaced by another number, let's call it $\zeta^{(1)}$, satisfying the identities

$$-1 = \zeta^{(2)} = \zeta^{(4)} = \zeta^{(6)} = ...$$
$$\text{and} \quad +i = \zeta^{(1)} = \zeta^{(3)} = \zeta^{(5)} = \zeta^{(7)} = ...$$

In other terms, we would need to follow the path shown in Figure 28.
This algebra is referred to as the $\Theta^{[n]}$-algebra and $\zeta^{(n)} \equiv \text{EvalT}[\Theta^{[n]}]$ with $\text{EvalT}[\Theta^{[n]}]$ denoting the *numerical evaluation of* $\Theta^{[n]}$. All of this will be specified in the following Subsection.

C. Rules of the $\Theta^{[n]}$-algebra

In the previous section we introduced $\Theta^{[n]}$ and the symbol $\text{EvalT}[\Theta^{[n]}]$. More precisely,
a) The *Theta-algebra* is an algebraic structure that deals with elements denoted by $\Theta^{[n]}$ and is equipped by an operation denoted by $*$;

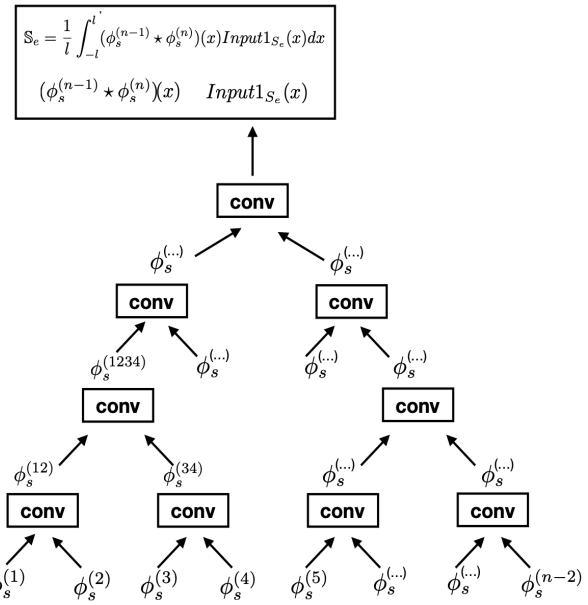

Fig. 5. Illustration of the algorithm for calculating constant $\mathbb{S}_e$. If n is an even number, we pair up two by two the $n-2$ input functions and we perform $n-3$ convolution operations between each pair. We iterate this process until there remains only one final function. This latter function is finally integrated with the convolution operation between the function $\phi_s^{(n-1)}$ and $\phi_s^{(n)}$ [i.e., $(\phi_s^{(n-1)} \star \phi_s^{(n)})(x)$]. Due to the commutative property, the order of the input functions is not relevant in the convolution operations.

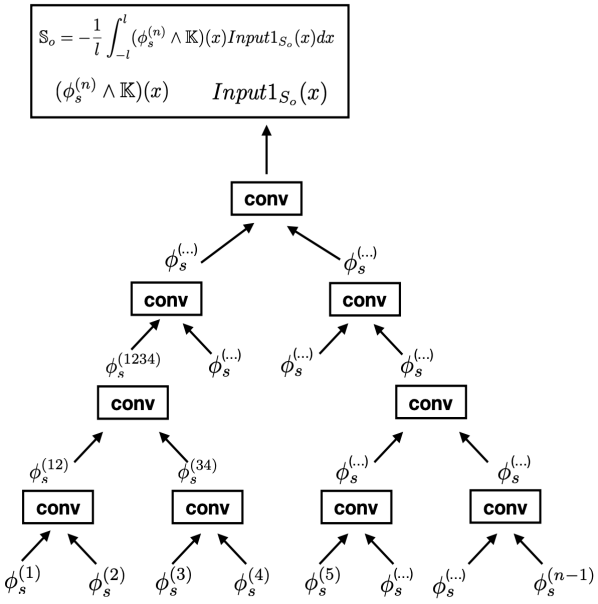

Fig. 6. Illustration of the algorithm for calculating constant $\mathbb{S}_o$. If n is an odd number, we pair up two by two the $n-1$ input functions and we perform $n-3$ convolution operations between each pair. We iterate this process until there remains only one final function is finally integrated. This latter function has to be integrated with function $(\phi_s^{(n)} \wedge \mathbb{K}(x))(x)$. Due to the commutative property, the order of the input functions is not relevant in the convolution operations.

b) $\Theta^{[n]}$ is an abstract symbol satisfying a finite set of axioms reported in Appendix A;
c) $\mathrm{EvalT}[\Theta^{[n]}]$ is a command that assigns a complex number to mathematical expressions involving $\Theta^{[n]}$ according to the definition shown in Appendix A.
For easy reference, and in order not to burden the reading of the work, here we report only the main algorithms for $\Theta^{[n]}$ and $\mathrm{EvalT}[\Theta^{[n]}]$ In fact, these are the main rules we need to perform our calculations. In Appendix A we can find a more complete set of rules to be satisfied by $\Theta^{[n]}$ and $\mathrm{EvalT}[\Theta^{[n]}]$ with some examples of calculation.

$$\begin{aligned}&\Theta^{[1]} * \Theta^{[1]} * \Theta^{[1]} * \cdots * \Theta^{[1]} \equiv \Theta^{[n]} \quad (n \text{ times})\\ &\Theta^{[m]} * \Theta^{[n]} = \Theta^{[n]} * \Theta^{[m]} = \Theta^{[m+n]}\end{aligned} \tag{28}$$

$$\begin{aligned}&\mathrm{EvalT}[\Theta^{[n]}]\\ &\equiv \cos\left(\frac{\pi}{4}(3+(-1)^n)\right)\\ &+ i\sin\left(\frac{\pi}{4}(3+(-1)^n)\right)\end{aligned} \tag{29}$$

$$\begin{aligned}&(\mathrm{EvalT}[\Theta^{[n]}])^m\\ &\equiv \cos\left(\frac{m\pi}{4}(3+(-1)^n)\right)\\ &+ i\sin\left(\frac{m\pi}{4}(3+(-1)^n)\right)\end{aligned} \tag{30}$$

Hence, as desired

$$\begin{aligned}&\mathrm{EvalT}[\Theta^{[n]}] = -1 \quad \text{if } n \text{ is an even number}\\ &\mathrm{EvalT}[\Theta^{[n]}] = +i \quad \text{if } n \text{ is an odd number}\end{aligned}$$

V. Display of the expression $\sum_{j=1}^{n} x_j a_j + y\Pi_{j=1}^{n} a_j$ by keeping secret the codes a_j

We are now in a position to solve our problem for n users. First of all, without compromising the participants' privacy, we choose the same function for all users (i.e., $\phi^{(j)}(x) = \phi(x)$ $\forall j$) where $\phi(x)$ is an even function i.e., $\phi(x) = \phi(-x)$. In this case, we have $\mathbb{S}_e = \mathbb{S}_o = 0$ and the generalized Parseval's identity reads[6]:

$$\frac{\alpha_0^n}{2} + \sum_{m=1}^{\infty} \alpha_m^n = 1 \quad \text{and} \quad \eta = \mathbb{C}^{-1/n} \tag{31}$$

with α_0 and α_m denoting the Fourier coefficients of $f(x)$. So, the main function reads $f(x) = \mathbb{C}^{-1/n}\phi(x)$. We define $(\alpha_j^{(0)}, \alpha_{j,m}) \equiv (|y|^{1/n} a_j \alpha_0, |y|^{1/n} a_j \alpha_m)$ with $j = 1, \cdots, n$.

A. Tasks of the Users

1) Users split their secret codes in four parts: $x_j a_j = a_j^{(1)} + a_j^{(2)} + a_j^{(3)} + a_j^{(4)}$;
2) Users chose the masks $\omega_{j,m} = a_{j,m} + \Theta^{[1]} b_{j,m}$; $\widehat{\omega}_{j,m} = \Theta^{[1]} a_{j,m} - b_{j,m}$; $\omega_j^{(0)} = a_j^{(0)} + \Theta^{[1]} b_j^{(0)}$ and $\widehat{\omega}_j^{(0)} = \Theta^{[1]} a_j^{(0)} - b_j^{(0)}$ with $j = 1, \cdots, n$. $\omega_j^{(0)}$, ω_j, $\widehat{\omega}_j^{(0)}$ and $\widehat{\omega}_j$ are *numbers*

[6]In order not to burden the notations, the tilde over the Fourier coefficients has been omitted.

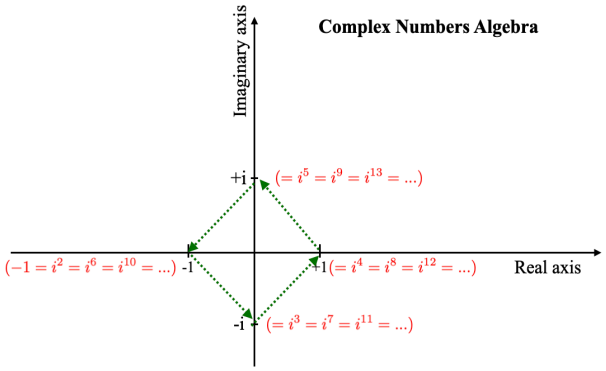

Fig. 7. The path is followed in the complex plane by raising to power the imaginary number i.

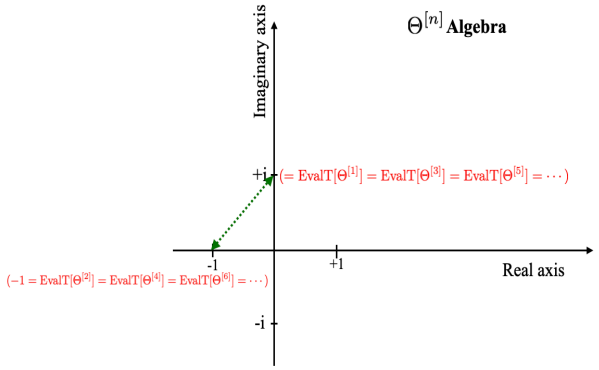

Fig. 8. Path in the complex plane obtained by raising $\zeta^{(1)}$ to a power.

arbitrarily chosen by the users. Note that is convenient to choose $\omega_j^{(0)} = \alpha^{(0)}(a_j^{(0)} + \Theta^{[1]} b_j^{(0)})$, $\widehat{\omega}_j^{(0)} = \alpha^{(0)}(\Theta^{[1]} a_j^{(0)} - b_j^{(0)})$, $\omega_{j,m} = (a_j + \Theta^{[1]} b_j)\alpha_m$, and $\widehat{\omega}_{j,m} = (\Theta^{[1]} a_j - b_j)\alpha_m$.

3) Users construct the four *hyper-vectors* $A_j^{(1)}, A_j^{(2)}, A_j^{(3)}, A_j^{(4)}$ defined as

$$
\begin{aligned}
A_j^{(1)} &\equiv \left\{ a_j^{(1)}, \alpha_j^{(0)} + \omega_j^{(0)}, \alpha_{j,m} + \omega_{j,m} \right\} \\
A_j^{(3)} &\equiv \left\{ a_j^{(3)}, \alpha_j^{(0)} + \widehat{\omega}_j^{(0)}, \alpha_{j,m} + \widehat{\omega}_{j,m} \right\} \\
A_j^{(2)} &\equiv \left\{ a_j^{(2)}, \alpha_j^{(0)} - \omega_j^{(0)}, \alpha_{j,m} - \omega_{j,m} \right\} \\
A_j^{(4)} &\equiv \left\{ a_j^{(4)}, \alpha_j^{(0)} - \widehat{\omega}_j^{(0)}, \alpha_{j,m} - \widehat{\omega}_{j,m} \right\}
\end{aligned}
$$

4) Users send the hyper-vectors $A_j^{(1)}$ and $A_j^{(2)}$ to the *Node 1* and the *Node 2*, respectively, and the hyper-vectors A_j^3 and A_j^4 to the *Node 3* and *Node 4*, respectively.

B. Tasks of the Nodes

The four nodes perform the following tasks. Note that the sign is $+$ if $y > 0$ and $-$ if $y < 0$:

Node 1 computes:

$$
\begin{aligned}
& N_1 \\
& \quad \equiv \sum_{j=1}^{n} a_j^{(1)} \\
& \quad \pm \left(\frac{1}{8} \widehat{\Pi}_{j=1}^{n} (\alpha_j^{(0)} + \omega_j^{(0)}) + \frac{1}{4} \sum_{m=1}^{\infty} \widehat{\Pi}_{j=1}^{n} (\alpha_{j,m} + \omega_{j,m}) \right)
\end{aligned}
\tag{32}
$$

Node 2 computes:

$$
\begin{aligned}
& N_2 \\
& \quad \equiv \sum_{j=1}^{n} a_j^{(2)} \\
& \quad \pm \left(\frac{1}{8} \widehat{\Pi}_{j=1}^{n} (\alpha_j^{(0)} - \omega_j^{(0)}) + \frac{1}{4} \sum_{m=1}^{\infty} \widehat{\Pi}_{j=1}^{n} (\alpha_{j,m} - \omega_{j,m}) \right)
\end{aligned}
\tag{33}
$$

Node 3 computes:

$$N_3 \equiv \sum_{j=1}^{n} a_j^{(3)} \pm \left(\frac{1}{8} \widehat{\Pi}_{j=1}^{n} (\alpha_j^{(0)} + \hat{\omega}_j^{(0)}) + \frac{1}{4} \sum_{m=1}^{\infty} \widehat{\Pi}_{j=1}^{n} (\alpha_{j,m} + \hat{\omega}_{j,m}) \right) \tag{34}$$

Node 4 computes:

$$N_4 \equiv \sum_{j=1}^{n} a_j^{(4)} \pm \left(\frac{1}{8} \widehat{\Pi}_{j=1}^{n} (\alpha_j^{(0)} - \hat{\omega}_j^{(0)}) + \frac{1}{4} \sum_{m=1}^{\infty} \widehat{\Pi}_{j=1}^{n} (\alpha_{j,m} - \hat{\omega}_{j,m}) \right) \tag{35}$$

C. Task of the Display

The display computes:

$$S = N_1 + N_2 + N_3 + N_4$$

The display shows:

$$\text{EvalT}[S] = \sum_{j=1}^{n} x_j a_j + y \Pi_{j=1}^{n} a_j$$

due to the *generalized Parseval's identity* (31). Note that to perform the operations, the Display takes into account the identities $\text{EvalT}[\Theta^{[2m]}] = -1$ and $\text{EvalT}[\Theta^{[2m-1]}] = +i$. The privacy of the participants is guaranteed if at least three nodes are not corrupted. Figure 9 shows the procedure.

VI. Display of the expression $\sum_{j=1}^{n} x_j a_j + y \Pi_{j=1}^{n} a_j$ by using $N = 3f + 1$ nodes

At this stage, we have solved the problem for n users, using 4 nodes. Participants' privacy is guaranteed that at least three nodes are not corrupted. However, requiring that three out of four nodes must not be corrupted is a quite restrictive constraint. This drawback can easily be overcome by using multiple nodes, say $N = 3f + 1$ with f denoting a natural number. The problem is solved if we can establish an algorithm that guarantees participants' privacy without modifying the constraint on the number of nodes that must not be corrupted. In other words, the number of nodes that are not to be corrupted must be 3 and must not depend on f. We shall solve this problem using $3f + 1$ nodes, grouped in different 4 categories, with the constraint that at least three nodes (and no more than three) belonging to three different categories must not be corrupted and at list one node, belonging to the *second level* of computation, is not corrupted. Firstly, it is convenient to account for all the quantities entered in the algorithm by a unique index, say ι, that can take only natural numbers (i.e., $\iota = 0, 1, 2, \cdots$). We group the $3f+1$ nodes in 4 different categories. Let us denote with κ the number of nodes in each category. If the total nodes is $N = 3f + 1$, we have $\kappa = 3(f - 1)/4$. Hence, in terms of ι:

$$\begin{aligned} f &= 1 + 4\iota \\ \kappa &= 3\iota \\ N &= 4(1 + 3\iota) \quad \text{with} \quad \iota = 0, 1, 2, \cdots \end{aligned}$$

A. Tasks of the Users

1) Users split the codes in 12ι parts: $x_j a_j = a_j^{(1)} + a_j^{(2)} + \cdots + a_j^{(12\iota)}$;
2) Each user (j) chooses:
2a) κ *positive supplementary masks*: $\lambda_j^{(0,s)}$, $\mu_j^{(0,s)}$, $\nu_j^{(0,s)}$, and $\sigma_j^{(0,s)}$ with $s = 1, \cdots, \kappa$;
2b) κ positive supplementary *multiplicative masks*: $\lambda_j^{(s)}$, $\mu_j^{(s)}$, $\nu_j^{(s)}$, and $\sigma_j^{(s)}$ with $s = 1, \cdots, \kappa$;
3) Users form the *hyper-vectors* $A^{(1)} - A^{(\kappa)}$, $B^{(\kappa+1)} - B^{(2\kappa)}$, $C^{(2\kappa+1)} - C^{(3\kappa)}$, and $D^{(3\kappa+1)} - B^{(4\kappa)}$;
(**4**) Users send the encrypted codes to the nodes.

B. Tasks of the Nodes in the Categories

The $N = 4(1 + 3\iota)$ nodes are grouped in 4 categories A, B, C, and D built as follows:

A = CATEGORY 1

Category A receives:

$$\begin{aligned} A^{(1)} &\equiv \Big\{ S^{(1)} = \sum_{j=1}^{n} a_j^{(1)}, \\ PA^{(0,1)} &= \widehat{\Pi}_{j=1}^{n} (\alpha_j^{(0)} \lambda_j^{(0,1)} + \omega_j^{(0)} \lambda_j^{(0,1)})^{1/\kappa}, \\ PA^{(1)} &= \widehat{\Pi}_{j=1}^{n} (\alpha_{j,m} \lambda_j^{(1)} + \omega_{j,m} \lambda_j^{(1)})^{1/\kappa} \Big\} \end{aligned}$$

$\cdots$

$$\begin{aligned} A^{(\kappa-1)} &\equiv \Big\{ S^{(\kappa-1)} = \sum_{j=1}^{n} a_j^{(\kappa-1)}, \\ PA^{(0,\kappa-1)} &= \widehat{\Pi}_{j=1}^{n} (\alpha_j^{(0)} \lambda_j^{(0,\kappa-1)} + \omega_j^{(0)} \lambda_j^{(0,\kappa-1)})^{1/\kappa} \\ PA^{(\kappa-1)} &= \widehat{\Pi}_{j=1}^{n} (\alpha_{j,m} \lambda_j^{(\kappa-1)} + \omega_{j,m} \lambda_j^{(\kappa-1)})^{1/\kappa} \Big\} \end{aligned}$$

$$\begin{aligned} A^{(\kappa)} &\equiv \Big\{ S^{(\kappa)} = \sum_{j=1}^{n} a_j^{(\kappa)}, \\ PA^{(0)} &= \widehat{\Pi}_{j=1}^{n} (\alpha_j^{(0)} \widetilde{\lambda}_j^{(0)} + \omega_j^{(0)} \widetilde{\lambda}_j^{(0)})^{1/\kappa}, \\ PA &= \widehat{\Pi}_{j=1}^{n} (\alpha_{j,m} \widetilde{\lambda}_j + \omega_{j,m} \widetilde{\lambda}_j)^{1/\kappa} \Big\} \end{aligned}$$

B = CATEGORY 2

Category B receives:

$$\begin{aligned} B^{(1)} &\equiv \Big\{ S^{(\kappa+1)} = \sum_{j=1}^{n} a_j^{(\kappa+1)}, \\ PB^{(0,1)} &= \widehat{\Pi}_{j=1}^{n} (\alpha_j^{(0)} \mu_j^{(0,1)} - \omega_j^{(0)} \mu_j^{(0,1)})^{1/\kappa}, \\ PB^{(\kappa)} &= \widehat{\Pi}_{j=1}^{n} (\alpha_{j,m} \mu_j^{(1)} - \omega_{j,m} \mu_j^{(1)})^{1/\kappa} \Big\} \end{aligned}$$

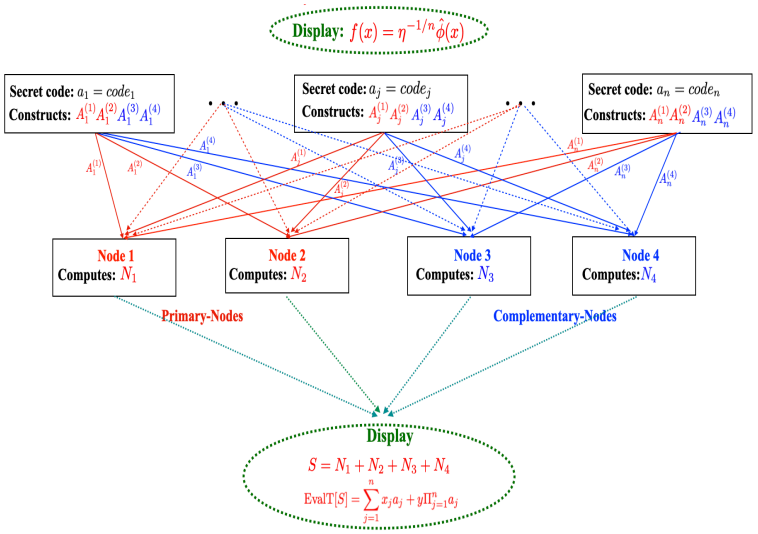

Fig. 9. Display of the expression $\sum_{j=1}^{n}x_ja_j+y\Pi_{j=1}^{n}a_j$ by keeping secret the codes a_j.

...

$$
\begin{aligned}
B^{(\kappa-1)} &\equiv \Big\{ S^{(2\kappa-1)}=\sum_{j=1}^{n}a_j^{(2\kappa-1)},\\
PB^{(0,\kappa-1)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_j^{(0)}\mu_j^{(0,\kappa-1)}-\omega_j^{(0)}\mu_j^{(0,\kappa-1)})^{1/\kappa},\\
PB^{(\kappa-1)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_{j,m}\mu_j^{(\kappa-1)}-\omega_{j,m}\mu_j^{(\kappa-1)})^{1/\kappa}\Big\}
\end{aligned}
$$

$$
\begin{aligned}
B^{(\kappa)} &\equiv \Big\{ S^{(2\kappa)}=\sum_{j=1}^{n}a_j^{(2\kappa)},\\
PB^{(0)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_j^{(0)}\widetilde{\mu}_j^{(0)}-\omega_j^{(0)}\widetilde{\mu}_j^{(0)})^{1/\kappa},\\
PB &= \widehat{\Pi}_{j=1}^{n}(\alpha_{j,m}\widetilde{\mu}_j-\omega_{j,m}\widetilde{\mu}_j)^{1/\kappa}\Big\}
\end{aligned}
$$

C = CATEGORY 3 Category C receives:

$$
\begin{aligned}
C^{(1)} &\equiv \Big\{ S^{(2\kappa+1)}=\sum_{j=1}^{n}a_j^{(2\kappa)},\\
PC^{(0,1)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_j^{(0)}\nu_j^{(0,1)}+\widehat{\omega}_j^{(0)}\nu_j^{(0,1)})^{1/\kappa},\\
PC^{(1)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_{j,m}\nu_j^{(1)}+\widehat{\omega}_{j,m}\nu_j^{(1)})^{1/\kappa}\Big\}
\end{aligned}
$$

...

$$
\begin{aligned}
C^{(\kappa-1)} &\equiv \Big\{ S^{(3\kappa-1)}=\sum_{j=1}^{n}a_j^{(3\kappa-1)},\\
PC^{(0,\kappa-1)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_j^{(0)}\nu_j^{(0,\kappa-1)}+\widehat{\omega}_j^{(0)}\nu_j^{(0,\kappa-1)})^{1/\kappa},\\
PC^{(\kappa-1)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_{j,m}\nu_j^{(\kappa-1)}+\widehat{\omega}_{j,m}\nu_j^{(\kappa-1)})^{1/\kappa}\Big\}
\end{aligned}
$$

$$
\begin{aligned}
C^{(\kappa)} &\equiv \Big\{ S^{(3\kappa)}=\sum_{j=1}^{n}a_j^{(3\kappa)},\\
PC^{(0)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_j^{(0)}\widetilde{\nu}_j^{(0)}+\widehat{\omega}_j^{(0)}\widetilde{\nu}_j^{(0)})^{1/\kappa},\\
PC &= \widehat{\Pi}_{j=1}^{n}(\alpha_{j,m}\widetilde{\nu}_j+\widehat{\omega}_{j,m}\widetilde{\nu}_j)^{1/\kappa}\Big\}
\end{aligned}
$$

D = CATEGORY 4 Category D receives:

$$
\begin{aligned}
D^{(1)} &\equiv \Big\{ S^{(3\kappa+1)}=\sum_{j=1}^{n}a_j^{(3\kappa)},\\
PD^{(0,1)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_j^{(0)}\sigma_j^{(0,1)}-\widehat{\omega}_j^{(0)}\sigma_j^{(0,1)})^{1/\kappa},\\
PD^{(1)} &= \widehat{\Pi}_{j=1}^{n}\alpha_{j,m}(\sigma_j^{(1)}-\widehat{\omega}_{j,m}\sigma_j^{(1)})^{1/\kappa}\Big\}
\end{aligned}
$$

...

$$
\begin{aligned}
D^{(\kappa-1)} &\equiv \Big\{ S^{(4\kappa-1)}=\sum_{j=1}^{n}a_j^{(4\kappa-1)},\\
PD^{(0,\kappa-1)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_j^{(0)}\sigma_j^{(0,\kappa-1)}-\widehat{\omega}_j^{(0)}\sigma_j^{(0,\kappa-1)})^{1/\kappa},\\
PD^{(\kappa-1)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_{j,m}\sigma_j^{(\kappa-1)}-\widehat{\omega}_{j,m}\sigma_j^{(\kappa-1)})^{1/\kappa}\Big\}
\end{aligned}
$$

$$
\begin{aligned}
D^{(\kappa)} &\equiv \Big\{ S^{(4\kappa)}=\sum_{j=1}^{n}a_j^{(4\kappa)},\\
PD^{(0)} &= \widehat{\Pi}_{j=1}^{n}(\alpha_j^{(0)}\widetilde{\sigma}_j^{(0)}-\widehat{\omega}_j^{(0)}\widetilde{\sigma}_j^{(0)})^{1/\kappa},\\
PD &= \widehat{\Pi}_{j=1}^{n}(\alpha_{j,m}\widetilde{\sigma}_j-\widehat{\omega}_{j,m}\widetilde{\sigma}_j)^{1/\kappa}\Big\}
\end{aligned}
$$

Masks λ_j, μ_j,ν_j, and σ_j $\lambda_j^{(0,s)}$, $\lambda_j^{(s)}$, etc. are *real numbers* different from zero and the variable with the tilde stands for the *inverse of the variable*, i.e.,

$$\lambda_j\widetilde{\lambda}_j = 1 \quad ; \quad \mu_j\widetilde{\mu}_j = 1 \quad ; \quad \nu_j\widetilde{\nu}_j = 1 \; ;$$
$$\sigma_j\widetilde{\sigma}_j = 1 \quad ; \quad \lambda_j{}^{(0,s)}\widetilde{\lambda}_j^{(0,s)} = 1 \quad \text{etc.} \quad (j = 1, \cdots, n)$$

Moreover,

$$\widetilde{\lambda}_j^{(0)} \equiv \Pi_{s=1}^{\kappa-1}\widetilde{\lambda}_j^{(0,s)} \; , \; \widetilde{\mu}_j^{(0)} \equiv \Pi_{s=1}^{\kappa-1}\widetilde{\mu}_j^{(0,s)}$$
$$\widetilde{\nu}_j^{(0)} \equiv \Pi_{s=1}^{\kappa-1}\widetilde{\nu}_j^{(0,s)} \; , \; \widetilde{\sigma}_j^{(0)} \equiv \Pi_{s=1}^{\kappa-1}\widetilde{\sigma}_j^{(0,s)}$$
$$\widetilde{\lambda}_j \equiv \Pi_{s=1}^{\kappa-1}\widetilde{\lambda}_j^{(s)} \; , \; \widetilde{\mu}_j \equiv \Pi_{s=1}^{\kappa-1}\widetilde{\mu}_j^{(s)}$$
$$\widetilde{\nu}_j \equiv \Pi_{s=1}^{\kappa-1}\widetilde{\nu}_j^{(s)} \; , \; \widetilde{\sigma}_j \equiv \Pi_{s=1}^{\kappa-1}\widetilde{\sigma}_j^{(s)} \qquad \forall j$$

C. Tasks of the Nodes N_1, N_2, N_3 and N_4

The nodes in the four categories perform the following tasks. Note that the sign is $+$ if $y > 0$ and $-$ if $y < 0$:
N_1 computes:

$$N_1 = \sum_{s=1}^{\kappa} S^{(s)} \pm \left(\frac{1}{8}PA^{(0)} \cdot \Pi_{s=1}^{\kappa-1}PA^{(0,s)} + \frac{1}{4}PA \cdot \Pi_{s=1}^{\kappa-1}PA^{(s)}\right) \tag{36}$$

N_2 computes:

$$N_2 = \sum_{s=\kappa+1}^{2\kappa} S^{(s)} \pm \left(\frac{1}{8}PB^{(0)} \cdot \Pi_{s=1}^{\kappa-1}PB^{(0,s)} + \frac{1}{4}PB \cdot \Pi_{s=1}^{\kappa-1}PB^{(s)}\right) \tag{37}$$

N_3 computes:

$$N_3 = \sum_{s=2\kappa+1}^{3\kappa} S^{(s)} \pm \left(\frac{1}{8}PC^{(0)} \cdot \Pi_{s=1}^{\kappa-1}PC^{(0,s)} + \frac{1}{4}PC \cdot \Pi_{s=1}^{\kappa-1}PC^{(s)}\right) \tag{38}$$

N_4 computes:

$$N_4 = \sum_{s=3\kappa+1}^{4\kappa} S^{(s)} \pm \left(\frac{1}{8}PD^{(0)} \cdot \Pi_{s=1}^{\kappa-1}PD^{(0,s)} + \frac{1}{4}PD \cdot \Pi_{s=1}^{\kappa-1}PD^{(s)}\right) \tag{39}$$

To perform calculations, the nodes use rule **vi)** of the $\Theta^{[n]}$-algebra, shown in Appendix A.

D. Task of the Display

The display computes: $S = N_1 + N_2 + N_3 + N_4$ and shows publicly
$\text{EvalT}[S] = \sum_{j=1}^{n} x_j a_j + y\Pi_{j=1}^{n} a_j$.
due to the *generalized Parseval's identity* (31). The privacy of the participants is guaranteed if at least three nodes belonging to three different categories are not corrupted. Figure 10 illustrates the procedure.

E. Considerations

To ensure the participants' privacy we have to impose the condition that at least three nodes, belonging to three different categories, are not corrupted. However, if we want to ensure that also the particular contribution of the addition or the multiplication to the final expression cannot be detected, we are bound to add that at least one of the four nodes $Node\ 1$, $Node\ 2$, $Node\ 3$ and $Node\ 4$ is not corrupted. Of course, in the second level of calculation, instead of 4 nodes we may add an arbitrary number of nodes without changing the restriction that at least one of these nodes must not be corrupted.

VII. Practical Considerations

The designs described in the previous sections rely on nodes on the side to compute mathematical expressions. In this section, we present designs that incorporate our nodes within the infrastructure of a number of semi-permissioned Blockchains. This enables the execution of our protocol as a side effect of the normal system operations, taking no additional dependency on extra authorities. It remains an open problem how to embed our protocol into permissionless systems [Nak08], [Woo17], based on proof of work or stake. These systems have a highly dynamic set of authorities maintaining the state of their blockchains, which cannot readily be mapped into the nodes of our system.

Integration of our system into Hyperledger Fabric [Cac16]—an example of permissioned blockchain platform—is straightforward. Fabric contracts run on private sets of computation authorities—and use the Fabric protocols for cross-contract calls [Woo17]. In this setting, our computing nodes can coincide with the Fabric smart contract authorities. Upon a contract set up, they assign nodes into the different categories and then compute mathematical expressions when authorized by the contract. To compute these expressions, the nodes maintain a number of secret values (see Section II) along with their traditional signing key needed for the normal system operations. Integrating our system into blockchains has obvious advantages over traditional smart contracts capable of only evaluating expression over public inputs—as currently present in the Hyperledger. In other words, our system can augment the capabilities of any permissioned blockchain to enable computations over secret inputs. The threshold trust assumption—namely that integrity and availability are guaranteed under the corruption of a subset of authorities [Cac16] is preserved, but need to consider the different categories of our nodes.

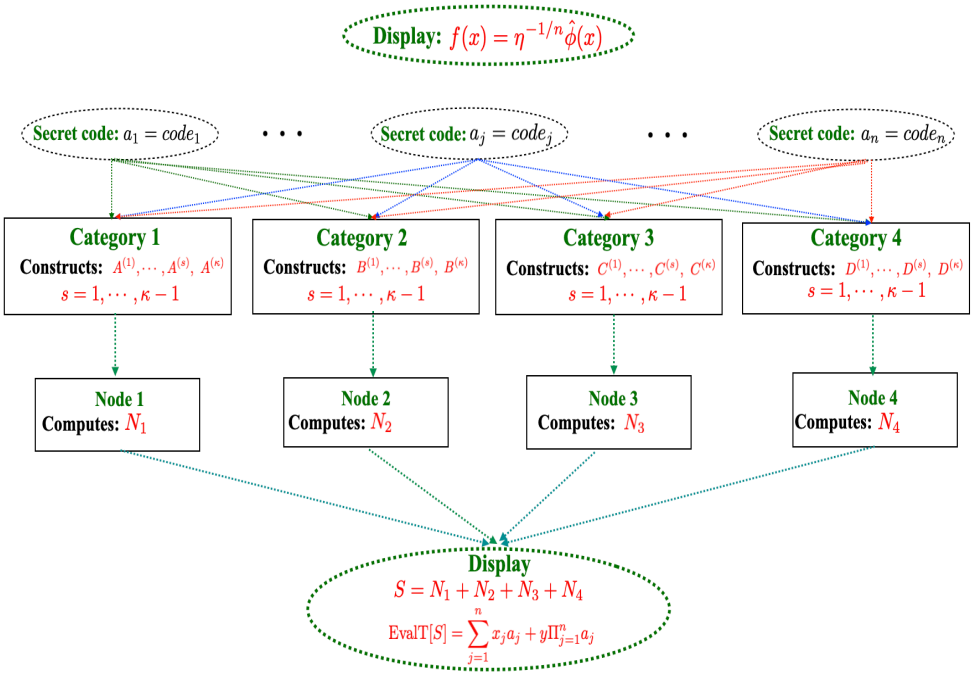

Fig. 10. Display of the expression $\sum_{j=1}^{n} x_j a_j + y\Pi_{j=1}^{n} a_j$ by using $N = 3f + 1$ nodes

We can also naturally embed our system into sharded scalable blockchains, as exemplified by Chainspace [ABSB$^+$18]. In these systems, transactions are distributed and executed on 'shards' of authorities, whose membership and public keys are known. Our computing nodes can naturally coincide with the authorities within a shard—a special object in Chainspace can signal to them to perform a private computation (rather than the default public computations). The authorities then attach their output to the transaction they are processing anyway. The trust assumptions of sharded blockchains naturally compose with the requirements of our computing nodes: each shard simply contains a node of every one of our four categories.

VIII. Related Works

There are two main constructions of multiparty protocols: circuit garbling and secret-sharing. Circuit garbling involves encrypting keys in a specific order to simulate a circuit evaluation [AIK14]; secret-sharing based protocol (as the one described in this paper) breaks the inputs among all nodes who use their shares to evaluate some function through local computations [BDOZ11], [DZ13], [NNOB12], [LPSY15].

SPDZ [DPSZ12] is one of the most notorious secret-sharing based multiparty computation protocols scaling to an arbitrary number of users; SPDZ is secure against active adversaries using MACs to verify the integrity of computations, and does not require any kind of trusted third parties; it requires however expensive somewhat homomorphic encryption (SHE) to generate the triples used to compute multiplication of secrets. SPDZ2 [DKL$^+$13] offers various improvements to the offline phase of SPDZ and allows the MACs to be checked without revealing their key, thus allowing the MAC to be re-used after it is checked. Mascot [KOS16] uses oblivious transfer rather than SHE to further improve performances of the offline phase and generate triples.

The literature following SPDZ mainly improves the offline phase, while our system innovates on both the offline and online phases. Most multiparty protocols for arithmetic circuits based on secret-sharing that scale to an arbitrary number of users is based on the algebra introduced by Donald Beaver [Bea91]. They thus require triples to compute the multiplication of secrets and impose communication between nodes during the online phase; their online latency, therefore, increases with the number of multiplications to evaluate. Our system comes with a different trade-off: our nodes do not communicate during the online phase and thus enjoy constant (and low) online latency in the size of the circuit, at the cost of not supporting the composition of operations (see Section IX). Established secret-sharing protocols face a trade-off between security and online latency—adding nodes improves security but increases latency. Our protocol forgoes this trade-off since multiplications do not require communication between the nodes.

IX. Limitations and Future Work

Our system has several limitations that are beyond the scope of this work and deferred to future work. We do not support the composition of operations. That is, while most established scheme [DPSZ12], [DKL$^+$13], [KOS16] can evaluate expressions like $(a + b)(c + d)$ with two additions and one multiplication, we need to distribute the operation and evaluate $(ac + ad + bc + bd)$. We also defer future work adapting our

scheme to withstand active adversaries, potentially adapting the MAC-based approach introduced by SPDZ [DPSZ12].

As we have seen, as an example of a solution, for the case of $3f + 1$ nodes we have proposed a protocol that includes four categories with the addition of 4 nodes in the *second level* of calculation. We have imposed the condition that at least three nodes, belonging to three different categories, are not corrupted and that at least one of the nodes belonging to the *second level* of calculation is not corrupted. This is one of the ways to solve the problem, but we imagine that simpler and less restrictive protocols may be proposed. This too will be the subject of future work.

X. Discussion

This work aims to solve the following problem: *Propose a method able to show publicly a general mathematical expression while keeping the players' codes secret*. The problem has been solved by using the so-called *Masks' method*. In short, this method consists in hiding the codes of the players within the parameters of the functions and being able to show only the numerical value taken by the mathematical expression by using the *(generalized) Parseval identity*. The problem has been solved by using the method of *additive masks* and by applying the $\Theta^{[n]}$ algebra. Calculations are simplified by choosing an even main function $f(x)$. The codes remain secret if at least three out of four nodes are not corrupted. The values of a general mathematical expression are obtained by using interpolating polynomials. When a given function *Expr.* is replaced by Chebyshev interpolating polynomials, thanks to *theorem 1*, we can minimize the error, until reaching the "level of precision" allowed by the computers, using a sufficiently high number of polynomials. Moreover, *theorem 2* guarantees that Chebyshev method is the best among the other methods based on the 2nd degree of equations as its convergence rate is equal to 3. It is worth noting that the Chebyshev polynomials method is particularly convenient also in physical or engineering applications where the exact mathematical expression *Expr.* is not known, but only the (experimental) values that *Expr.* takes in the nodes are known.

For treating simply the case of $3f + 1$ *nodes*, we adopted the method of *multiplicative secondary masks* that, for simplicity, may be *arbitrary positive real numbers*. Even in this case, calculations are simplified by choosing an *even main function* $f(x)$. The nodes are organized in *four categories*, each of which contains κ nodes. The nodes are labeled by the index ι, which takes natural numbers $\iota = 0, 1, 2, \cdots$. So, the numbers of nodes are $N = 4 + 12\iota$, $f = 1 + 4\iota$ and each category contains $\kappa = 3\iota$ nodes. Hence, the total number of nodes is the nodes contained in the four categories plus 4. The simplest case corresponds to $\iota = 0$, i.e., the categories are empty. The participants' privacy is guaranteed if *at least three nodes belonging to three different categories are not corrupted* and one node belonging to the second level of calculation is not corrupted. In this case, no one can determine the codes of the players: the nodes are not able to determine the codes of the users and the users are not able to determine the codes of the other users.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgements

We thank George Danezis and Ioannis Psaras for helpful suggestions on the early manuscript and valuable advice.

References

[ABSB+18] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. Chainspace: A Sharded Smart Contracts Platform. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.

[AIK14] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. How to garble arithmetic circuits. *SIAM Journal on Computing*, 43(2):905–929, 2014.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, 2011.

[Bea91] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Proceedings of Crypto*, 1991.

[Cac16] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

[Die18] Andreas Dieckmann. Collection of Infinite Product and Series. http://www-elsa.physik.uni-bonn.de/~dieckman/InfProd/InfProd.html, 2018.

[DKL+13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P Smart. Practical covertly secure MPC for dishonest majority–or: breaking the SPDZ limits. In *Proceedings of the European Symposium on Research in Computer Security*, 2013.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Proceedings of Crypto*, 2012.

[DZ13] Ivan Damgård and Sarah Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *Proceedings of the Theory of Cryptography Conference*, 2013.

[GR14] Izrail Solomonovich Gradshteyn and Iosif Moiseevich Ryzhik. *Table of integrals, series, and products*. Academic press, 2014.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the SIGSAC Conference on Computer and Communications Security*, 2016.

[LPSY15] Yehuda Lindell, Benny Pinkas, Nigel P Smart, and Avishay Yanai. Efficient constant round multi-party computation combining BMR and SPDZ. In *Proceedings of Crypto*, 2015.

[MH02] John C Mason and David C Handscomb. *Chebyshev polynomials*. Chapman and Hall/CRC, 2002.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Proceedings of Crypto*, 2012.

[Son19] Alberto Sonnino. Fmpc: Secure multiparty computation from fourier series and parseval's identity. *arXiv preprint arXiv:1912.02583*, 2019.

[Woo17] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger eip-150 revision. "http://gavwood.com/paper.pdf", 2016 (visited August 9, 2017).

APPENDIX

$$\sum_{m=1}^{\infty} \frac{(-1)^m}{m^2-\tau^2} = \frac{1}{2\tau^2}\left(1-\frac{\pi\tau}{\sin(\pi\tau)}\right) \quad ; \quad \sum_{m=1}^{\infty} \frac{1}{m^2-\tau^2} = \frac{1}{2\tau^2}\left(1-\pi\tau\cot(\pi\tau)\right) \tag{40}$$

The sum of powers of these expressions may be obtained by performing the derivatives with respect to parameter τ e.g.,

$$\sum_{m=1}^{\infty} \frac{1}{(m^2-\tau^2)^2} = -\frac{1}{2\tau^4} + \frac{\pi\sin(2\pi\tau)+2\pi^2\tau}{8\tau^3\sin^2(\pi\tau)}$$

If, for example, we chose $f(x)=\eta^{-1/n}\cos(\pi\tau x/l)$ with $\eta=\mathbb{C}^{-1}$, then

$$\alpha^{(0)} = 2\eta^{-1/n}\frac{\sin(\pi\tau)}{\pi\tau} \quad \text{and} \quad \alpha^{(m)} = \alpha^{(0)}\tau^2\frac{(-1)^m}{\tau^2-m^2}$$

For $n=2$ we get

$$\begin{aligned}
&\eta = 1+\frac{\sin(2\pi\tau)}{2\pi\tau}\\
&f(x) = \left(1+\frac{\sin(2\pi\tau)}{2\pi\tau}\right)^{-1/2}\cos(\pi\tau x/l)\\
&\alpha^{(0)} = 2\left(1+\frac{\sin(2\pi\tau)}{2\pi\tau}\right)^{-1/2}\frac{\sin(\pi\tau)}{\pi\tau}\\
&\alpha^{(m)} = \alpha^{(0)}\tau^2\frac{(-1)^m}{\tau^2-m^2}
\end{aligned}$$

In this Section, we prove the validity of the generalized Parseval's identity given by Eq. (25). To this aim, let us first consider the constant $\mathbb{C}$ defined in Eq. (20) and the *cosine main functions* defined in Eq. (18). The convolution operation between two *cosine main functions* $\phi_c^{(\kappa_1)}(x)$ and $\phi_c^{(\kappa_2)}(x)$, with κ_1 and κ_2 integer numbers subject to the conditions $1\le\kappa_1,\kappa_2\le n\ ;\ \kappa_1\neq\kappa_2$, reads

$$(\phi_c^{(\kappa_1)}\star\phi_c^{(\kappa_2)})(x) = \frac{1}{2}\alpha_0^{(\kappa_1)}\alpha_0^{(\kappa_2)} + \sum_{m=1}^{\infty}\alpha_m^{(\kappa_1)}\alpha_m^{(\kappa_2)}\cos\Big(\frac{m\pi}{l}x\Big)$$

The convolution operation between this function and another cosine main function, say $\phi_c^{(\kappa_3)}(x)$ with κ_3 integer number subject to the conditions $1\le\kappa_3\le n\ ;\ \kappa_3\neq\kappa_1\neq\kappa_2$, gives

$$\begin{aligned}
&((\phi_c^{(\kappa_1)}\star\phi_c^{(\kappa_2)})\star\phi_c^{(\kappa_3)})(x)\\
&= \frac{1}{2}\alpha_0^{(\kappa_1)}\alpha_0^{(\kappa_2)}\alpha_0^{(\kappa_3)} + \sum_{m=1}^{\infty}\alpha_m^{(\kappa_1)}\alpha_m^{(\kappa_2)}\alpha_m^{(\kappa_3)}\cos\Big(\frac{m\pi}{l}x\Big)
\end{aligned} \tag{41}$$

Hence, by performing $n-2$ convolution operations, sequentially, starting from $\phi_c^{(1)}(x)$ to $\phi_c^{(n-1)}(x)$ we get

$$\begin{aligned}
&((\cdots(((\phi_c^{(1)}\star\phi_c^{(2)})\star\phi_c^{(3)})\star\phi_c^{(4)})\star\cdots)\star\phi_c^{(n-1)})(x)\\
&\equiv Input1_C(x)\\
&= \frac{1}{2}\Pi_{j=1}^{n-1}\alpha_0^{(j)} + \sum_{m=1}^{\infty}\Pi_{j=1}^{n-1}\alpha_m^{(j)}\cos\Big(\frac{m\pi}{l}x\Big)
\end{aligned}$$

By integrating Eq. (42) with $\phi_c^{(n)}(x)$, we finally have

$$\begin{aligned}
\mathbb{C} &= \frac{1}{l}\int_{-l}^{l}\phi_c^{(n)}(x)Input1_C(x)dx\\
&= \frac{1}{2}\Pi_{j=1}^{n}\alpha_0^{(j)} + \sum_{m=1}^{\infty}\Pi_{j=1}^{n}\alpha_m^{(j)}
\end{aligned} \tag{42}$$

If we now perform the convolution operation between two *sine main functions* defined in Eq. (18), say $\phi_s^{(\kappa_1)}(x)$ and $\phi_s^{(\kappa_2)}(x)$ with κ_1 and κ_2 integer numbers subject to the conditions $1\le\kappa_1,\kappa_2\le n\ ;\ \kappa_1\neq\kappa_2$, we get

$$(\phi_s^{(\kappa_1)}\star\phi_s^{(\kappa_2)})(x) = -\sum_{m=1}^{\infty}\beta_m^{(\kappa_1)}\beta_m^{(\kappa_2)}\cos\Big(\frac{m\pi}{l}x\Big)$$

Hence, if n is an even number, we may pair up two by two $n-2$ input functions and we may perform $n-3$ convolution operations between each pair by getting

$$\begin{aligned}
&((\phi_s^{(1)}\star\phi_s^{(2)})\star(\phi_s^{(3)}\star\phi_s^{(4)})\star\cdots\star(\phi_s^{(n-3)}\star\phi_s^{(n-2)}))(x)\\
&\equiv Input1_{S_e}(x)\\
&Input1_{S_e}(x) = -\sum_{m=1}^{\infty}\Pi_{j=1}^{n-2}\beta_m^{(j)}\cos\Big(\frac{m\pi}{l}x\Big)
\end{aligned}$$

Since

$$(\phi_s^{(n-1)}\star\phi_s^{(n)})(x) = -\sum_{m=1}^{\infty}\beta_m^{(n-1)}\beta_m^{(n)}\cos\Big(\frac{m\pi}{l}x\Big)$$

we finally obtain

$$\begin{aligned}
\mathbb{S}_e &= \frac{1}{l}\int_{-l}^{l}(\phi_s^{(n-1)}\star\phi_s^{(n)})(x)Input1_{S_e}(x)dx\\
&= \sum_{m=1}^{\infty}\Pi_{j=1}^{n}\beta_m^{(j)}
\end{aligned} \tag{43}$$

Similarly, if n is an odd number, firstly we note that

$$(\phi_s^{(n)}\wedge\mathbb{K})(x) = \sum_{m=1}^{\infty}\beta_m^{(n)}\cos\Big(\frac{m\pi}{l}x\Big)$$

Since

$$\begin{aligned}
&((\phi_s^{(1)}\star\phi_s^{(2)})\star(\phi_s^{(3)}\star\phi_s^{(4)})\star\cdots\star(\phi_s^{(n-2)}\star\phi_s^{(n-1)}))(x)\\
&\equiv Input1_{S_o}(x)\\
&= -\sum_{m=1}^{\infty}\Pi_{j=1}^{n-1}\beta_m^{(j)}\cos\Big(\frac{m\pi}{l}x\Big)
\end{aligned}$$

we finally get

$$\mathbb{S}_o = -\frac{1}{l}\int_{-l}^{l}(\phi_s^{(n)}\wedge\mathbb{K})(x)Input1_{S_o}(x)dx = \sum_{m=1}^{\infty}\Pi_{j=1}^{n}\beta_m^{(j)} \tag{44}$$

Adding Eq. (42) to Eq. (43) (if n is an even number), or adding Eq. (42) to Eq. (44) (if n is an odd number), we obtain

the *generalised Parseval's identity* for n inputs $\phi^{(j)}(x)$ (with $j = 1, 2, \cdots, n$)

$$\begin{aligned} &\mathbb{C} + \mathbb{S}_\kappa \\ &= \frac{1}{2}\Pi_{j=1}^n \alpha_0^{(j)} + \sum_{m=1}^{\infty}(\Pi_{j=1}^n \alpha_m^{(j)}) + \sum_{m=1}^{\infty}(\Pi_{j=1}^n \beta_m^{(j)}) \quad \text{with} \\ &\mathbb{S}_\kappa = \begin{cases} \mathbb{S}_e & \text{if n=even number} \\ \mathbb{S}_o & \text{if n=odd number} \end{cases} \end{aligned} \tag{45}$$

This theorem allows computing the normalization constant η:

$$\eta = (\mathbb{C} + \mathbb{S}_\kappa)^{-1/n}$$

and the *normalised main functions* $f^{(j)}(x)$ read

$$f^{(j)}(x) = \eta\phi^{(j)}(x) \quad \text{with} \quad j = 1, 2, \cdots, n$$

If we use the main functions $f^{(j)}(x)$ the generalised Parseval's identity reads

$$\frac{1}{2}\Pi_{j=1}^n \widetilde{\alpha}_0^{(j)} + \sum_{m=1}^{\infty}(\Pi_{j=1}^n \widetilde{\alpha}_m^{(j)}) + \sum_{m=1}^{\infty}(\Pi_{j=1}^n \widetilde{\beta}_m^{(j)}) = 1 \tag{46}$$

with $\widetilde{\alpha}_0^{(j)}$, $\widetilde{\alpha}_m^{(j)}$, and $\widetilde{\beta}_m^{(j)}$ denoting the Fourier coefficients of $f^{(j)}(x)$. The schemes of calculation for coefficients $\mathbb{C}$, $\mathbb{S}_e$ and $\mathbb{S}_o$ are illustrated in Figure 4, 5, and 6, respectively.

- $\Theta^{[n]}$ algebra

i)

$$\begin{aligned} &\Theta^{[1]} * \Theta^{[1]} * \Theta^{[1]} * \cdots * \Theta^{[1]} \equiv \Theta^{[n]} \quad (n \text{ times}); \\ &\widehat{\Pi}_{j=1}^n h_i \equiv h_1 * h_2 * \cdots * h_n \end{aligned} \tag{47}$$

ii)

$$\begin{aligned} &\Theta^{[n]} * 1 = 1 * \Theta^{[n]} = \Theta^{[n]}; \\ &\Theta^{[n]} * 0 = 0 * \Theta^{[n]} = 0 \end{aligned} \tag{48}$$

iii)

$$\begin{aligned} \Theta^{[m]} * (\Theta^{[n]} * \Theta^{[\kappa]}) &= (\Theta^{[m]} * \Theta^{[n]}) * \Theta^{[\kappa]} \\ &= \Theta^{[m]} * \Theta^{[n]} * \Theta^{[\kappa]} \\ &= \Theta^{[m+n+\kappa]} \end{aligned} \tag{49}$$

with m, n and k denoting *positive natural numbers*.

iv)

$$\begin{aligned} &\Theta^{[m]} * \Theta^{[n]} = \Theta^{[n]} * \Theta^{[m]} \\ &= \Theta^{[m+n]}; \; (\Theta^{[n]})^m \equiv \Theta^{[nm]} \end{aligned} \tag{50}$$

v)

$$\begin{aligned} &\Theta^{[n]} * (x + y) = \Theta^{[n]} * x + \Theta^{[n]} * y \\ &= \Theta^{[n]} x + \Theta^{[n]} y = x\Theta^{[n]} + y\Theta^{[n]} \quad (x * y \equiv xy) \end{aligned} \tag{51}$$

with x and y denoting *two complex numbers*.

vi)

$$\left((x_1 + x_2\Theta^{[n]})^{1/\kappa}\right)^{\kappa} \equiv x_1 + x_2\Theta^{[n]} \tag{52}$$

with x_1, x_2 denoting *numbers* and κ is a *real number*, respectively.

- The command EvalT[...]

vii)

$$\begin{aligned} &\text{EvalT}[\Theta^{[n]}] \\ &\equiv \exp\left(\frac{\pi}{4}(3 + (-1)^n)\right) \end{aligned} \tag{53}$$

with n denoting *positive natural numbers*.

viii)

$$\begin{aligned} &(\text{EvalT}[\Theta^{[n]}])^m \\ &\equiv \exp\left(\frac{m\pi}{4}(3 + (-1)^n)\right) \end{aligned} \tag{54}$$

with m denoting a *number*

ix)

$$\begin{aligned} &\text{EvalT}[f(\mathbf{x}_1, \Theta^{[n]}) + g(\mathbf{x}_2, \Theta^{[n]})] \\ &\equiv f(\mathbf{x}_1, \text{EvalT}[\Theta^{[n]}]) + g(\mathbf{x}_2, \text{EvalT}[\Theta^{[n]}]) \end{aligned} \tag{55}$$

with x_1 and x_2 denoting *numbers*.

x)

$$\begin{aligned} &\text{EvalT}[f(\mathbf{x}_1, \Theta^{[n]})g(\mathbf{x}_2, \Theta^{[n]})] \\ &\equiv f(\mathbf{x}_1, \text{EvalT}[\Theta^{[n]}])g(\mathbf{x}_2, \text{EvalT}[\Theta^{[n]}]) \end{aligned} \tag{56}$$

with f and g denoting *general functions* with x, y denoting *numbers*.

xi)

$$\text{EvalT}[f(\mathbf{x}, (\Theta^{[n]})^y)] \equiv f(\mathbf{x}, \text{EvalT}[\Theta^{[ny]}] \tag{57}$$

The basic rule is to perform the operation $*$ first and then apply the command **EvalT [...]** by taking into account the properties **i)** - **x)** above.

- Identities and Differences

$(\text{EvalT}[\Theta^{[2n-1]}])^2 = \text{EvalT}[\Theta^{[2m]}] = -1$ with m and n denoting *integer numbers*.

$$\text{EvalT}[\Theta^{[n]}]\text{EvalT}[\Theta^{[m]}] \neq \text{EvalT}[\Theta^{[n+m]}] \tag{58}$$

$$(\text{EvalT}[\Theta^{[n]}])^y \neq \text{EvalT}[\Theta^{[ny]}] \tag{59}$$

$$\begin{aligned} &\text{EvalT}[f(\mathbf{x}_1, \Theta^{[n]}) * g(\mathbf{x}_2, \Theta^{[n]})] \\ &\neq \text{EvalT}[f(\mathbf{x}_1, \Theta^{[n]})]\text{EvalT}[g(\mathbf{x}_2, \Theta^{[n]})] \end{aligned} \tag{60}$$

- Examples

$$\begin{aligned} &(x_1 + i\Theta^{[1]}y_1) * (x_2 + i\Theta^{[2]}y_2) \\ &= x_1x_2 + i\Theta^{[1]}x_2y_1 + i\Theta^{[2]}x_1y_2 - \Theta^{[3]}y_1y_2 \end{aligned} \tag{61}$$

with x_1, x_2, y_1, and y_2 denoting *numbers*.

$$\begin{aligned} &\text{EvalT}[(x_1 + i\Theta^{[1]}y_1) * (x_2 + i\Theta^{[2]}y_2)] \\ &= \text{EvalT}[x_1x_2 + i\Theta^{[1]}x_2y_1 + i\Theta^{[2]}x_1y_2 - \Theta^{[3]}y_1y_2] \\ &= x_1x_2 - x_2y_1 - i(x_1y_2 + y_1y_2) \end{aligned} \tag{62}$$