FMPC: Secure Multiparty Computation from Fourier Series and Parseval's Identity

Alberto Sonnino
University College London (UCL)
alberto.sonnino@ucl.ac.uk

***Abstract*—FMPC is a novel multiparty computation protocol of arithmetic circuits based on secret-sharing, capable of computing multiplication of secrets with no online communication; it thus enjoys constant online communication latency in the size of the circuit. FMPC is based on the application of Fourier series to Parseval's identity, and introduces the first generalization of Parseval's identity for Fourier series applicable to an arbitrary number of inputs. 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, but is only suitable for the evaluation of low-depth arithmetic circuits. FMPC relies on an offline phase consisting of traditional preprocessing as introduced by established protocols like SPDZ, and innovates on the online phase that mainly consists of each node locally evaluating specific functions. FMPC paves the way for a new kind of multiparty computation protocols capable of computing multiplication of secrets as an alternative to circuit garbling and the traditional algebra introduced by Donald Beaver in 1991.**

I. Introduction

Multiparty computation protocols allow multiple users to compute some function of their combined secret inputs without revealing any additional information about their inputs other than the output of the function. FMPC is a secret-sharing based protocol for arithmetic circuits [3]; it operates in a setting where users wish to compute a function over some secrets by submitting the computation to a set of nodes, and is only suitable for circuits with a low number of multiplications. The users first secret-share their inputs by breaking them into multiple shares, and provide each node with one each. The nodes then perform additions and multiplications on these shares by local computations, and finally output the result of the computation. FMPC focuses on the computation of multiplication of secrets, and assumes that additions can be performed using traditional algebra as described by SPDZ [3].

As previous secret-sharing based protocols [3], [2], [5], FMPC divides execution into an *offline phase* and an *online phase*. The offline phase is performed ahead of time and does not involve any users secret input; the output of the computation is then evaluated during the online phase. Traditional secret-sharing based protocols are efficient to compute additions of secrets, but computing multiplication is expensive [3]; these are based on the algebra introduced by Donald Beaver [1] relying on the existence of some additional secret-shared values called *triples*, that are generated during the offline phase. Each node then broadcasts their shares of secrets blinded with these triples value. This causes high communication complexity during the online phase, especially for computations requiring many multiplications; their latency increases with the number of multiplications to evaluate.

FMPC is a novel secret-sharing technique to compute multiplication of secrets without requiring nodes to communicate with each other at all during the online phase; FMPC thus enjoys constant (and low) online communication latency in the size of the circuit. This is achieved through the application of Fourier series to Parseval's identity. On the downside, FMPC cannot compose operations and is therefore only suitable to evaluate circuits with a small number of multiplications (see Section VII). FMPC relies on established preprocessing techniques for the offline phase, and makes the following contributions to the online phase:

- Section IV presents the mathematical construction behind FMPC by taking the example of a two-user computation.
- Section V provides a concrete instantiation of FMPC and shows a practical protocol execution.
- Section VI introduces the first generalization of Parseval's identity for Fourier series applicable to an arbitrary number of inputs, and uses it to extend the two-user computation scheme presented in Section IV to a scheme supporting an arbitrary number of users. At the best of our knowledge, this is the first secret-sharing multiparty computation protocol scaling to an arbitrary number of inputs that enables multiplication of secrets with no online communication.

FMPC is a first of its kind attempt to analytically model MPC and aims to trigger further debates towards a working system.

II. Threat Model and Goals

The following actors participate in a FMPC computation:

- **Users:** End-user devices submit a computation over some secret inputs to a set of nodes; they wish to publish the output of a computation without revealing their secret inputs to anybody. Without loss of generality, we assume that each user hold one secret input.
- **Nodes:** Infrastructure executing the computation submitted by the users.

We model the offline phase as executed by a trusted authority responsible to generate some scheme parameters and communicate them to the users; this offline phase can be distributed using traditional techniques introduced by SPDZ [3] (see Section IV-B). FMPC assumes passive adversaries who follow the protocol specification but try to learn more than allowed about the users secret inputs[1]. Nodes can collude with each other as long as there is at least one honest non-colluding node.

[1]We leave the extension of FMPC to active adversaries as future work; potentially adapting the MAC-based approach introduced by SPDZ [3].

Under the above threat model, FMPC achieves the following design goals:

- **Private Computation** - Parties only learn the output of the computation.
- **Non-Interactivity** - Nodes do not communicate with each other during the online phase to perform computations.

III. Background

We recall the theory of Fourier series and Parseval's identity, and the expression of some useful convergent sums analytically; The long version of the paper[2] shows how to compute them numerically using finite fields.

A. Convolution of Fourier Series

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

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{+\infty} a_n \cos\Big(\frac{n\pi x}{l}\Big) + \sum_{n=1}^{+\infty} b_n \sin\Big(\frac{n\pi x}{l}\Big) \quad (1)$$

$$g(x) = \frac{\alpha_0}{2} + \sum_{n=1}^{+\infty} \alpha_n \cos\Big(\frac{n\pi x}{l}\Big) + \sum_{n=1}^{+\infty} \beta_n \sin\Big(\frac{n\pi x}{l}\Big)$$

where the Fourier coefficients $(a_0,\ a_n,\ b_n)$ and $(\alpha_0,\ \alpha_n,\ \beta_n)$ (for $n = 1, 2, \cdots$) are given below:

$$a_0 = \frac{1}{l}\int_{-l}^{l} f(x)dx\ ; \quad a_n = \frac{1}{l}\int_{-l}^{l} f(x)\cos\Big(\frac{n\pi x}{l}\Big)dx \quad (2)$$

$$b_n = \frac{1}{l}\int_{-l}^{l} f(x)\sin\Big(\frac{n\pi x}{l}\Big)dx$$

$$\alpha_0 = \frac{1}{l}\int_{-l}^{l} g(x)dx\ ; \quad \alpha_n = \frac{1}{l}\int_{-l}^{l} g(x)\cos\Big(\frac{n\pi x}{l}\Big)dx$$

$$\beta_n = \frac{1}{l}\int_{-l}^{l} g(x)\sin\Big(\frac{n\pi x}{l}\Big)dx$$

The convolution function between $f(x)$ and $g(x)$ is defined as

$$\Phi(x) = \Big(f \star g\Big)(x) \equiv \frac{1}{l}\int_{-l}^{l} f(y)g(x-y)dy \quad (3)$$

By inserting Equation (1) into Equation (3), and by taking into account the following identities

$$\frac{1}{l}\int_{-l}^{l} \sin\big(\frac{n'\pi}{l}x\big)\sin\big(\frac{n\pi}{l}x\big)dx = \delta_{nn'} \quad (4)$$

$$\frac{1}{l}\int_{-l}^{l} \cos\big(\frac{n'\pi}{l}x\big)\cos\big(\frac{n\pi}{l}x\big)dx = \delta_{nn'}$$

$$\int_{-l}^{l} \sin\big(\frac{n'\pi}{l}x\big)\cos\big(\frac{n\pi}{l}x\big)dx = 0$$

where $\delta_{nn'}$ denotes Kronecker's delta, we obtain the Fourier series of the convolution between two functions:

$$\Phi(x) = \frac{c_0}{2} + \sum_{n=1}^{\infty} c_n \cos\big(\frac{n\pi}{l}x\big) + \sum_{n=1}^{\infty} d_n \sin\big(\frac{n\pi}{l}x\big) \quad (5)$$

[2]https://arxiv.org/abs/1912.02583

where

$$c_0 = \frac{a_0\alpha_0}{2} \quad ; \quad c_n = a_n\alpha_n - a_n\beta_n \quad ; \quad d_n = b_n\alpha_n + b_n\beta_n$$

We also recall that the convolution operation satisfies commutativity and associativity; these properties are used in Section VI to scale FMPC to an arbitrary number of inputs.

B. Parseval's Identity

Let's assume two functions $f(x)$ and $g(x) \in \mathbb{L}^2[-l, l]$ as defined in Equation (1); defining the four vectors $\mathbf{A}$, $\mathbf{B}$, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (for $n = 1, 2, \cdots$) as below,

$$\mathbf{A} = \{a_0/\sqrt{2},\ a_n\} \ ; \ \mathbf{B} = \{b_n\} \quad (6)$$
$$\boldsymbol{\alpha} = \{\alpha_0/\sqrt{2},\ \alpha_n\} \ ; \ \boldsymbol{\beta} = \{\beta_n\}$$

Parseval's identity [4] holds for $f(x)$ and $g(x)$:

$$\mathbf{A}\cdot\boldsymbol{\alpha} + \mathbf{B}\cdot\boldsymbol{\beta} = \frac{1}{l}\int_{-l}^{l} f(x)g(x)dx \quad \text{or} \quad (7)$$

$$\Big(\frac{a_0\alpha_0}{2} + \sum_{n=1}^{\infty} a_n\alpha_n\Big) + \Big(\sum_{n=1}^{\infty} b_n\beta_n\Big) = \frac{1}{l}\int_{-l}^{l} f(x)g(x)dx$$

Parseval's identity only applies to two functions; Section VI-A presents our generalization of Parseval's identity that applies to an arbitrary number of functions used to extend FMPC to an arbitrary number of inputs.

C. Convergent Sums

FMPC requires the computation of scalar products of vectors with infinite components. It is therefore crucial that the infinite series produced by these scalar products are convergent, and that the results of these series can be computed efficiently and exactly (*i.e.,* analytically). For example, in case of two users, FMPC requires the evaluation of the following convergent sums (see Section V):

$$\sum_{n=1}^{\infty} \frac{1}{(\gamma^2 - n^2)(\delta^2 - n^2)} \quad ; \quad \sum_{n=1}^{\infty} \frac{n^2}{(\gamma^2 - n^2)(\delta^2 - n^2)} \quad (8)$$

These expressions can be easily calculated from the following well-known identity [4]

$$\Omega(\gamma) = \sum_{n=1}^{\infty} \frac{1}{(\gamma^2 - n^2)} = \frac{\pi}{2\gamma}\cot(\pi\gamma) - \frac{1}{2\gamma^2} \quad (9)$$

as below:

$$\sum_{n=1}^{\infty} \frac{1}{(\gamma^2 - n^2)(\delta^2 - n^2)} = \frac{1}{\delta^2 - \gamma^2}\Big(\Omega(\gamma) - \Omega(\delta)\Big)$$
$$= \frac{\pi}{2(\delta^2 - \gamma^2)}\Big(\frac{\cot(\pi\gamma)}{\gamma} - \frac{\cot(\pi\delta)}{\delta}\Big) - \frac{1}{2\gamma^2\delta^2}$$

$$\sum_{n=1}^{\infty} \frac{n^2}{(\gamma^2 - n^2)(\delta^2 - n^2)} = \frac{1}{\delta^2 - \gamma^2}\Big(\gamma^2\Omega(\gamma) - \delta^2\Omega(\delta)\Big)$$
$$= \frac{\pi}{2(\delta^2 - \gamma^2)}\Big(\gamma\cot(\pi\gamma) - \delta\cot(\pi\delta)\Big)$$

Section V illustrates that a convenient choice of the mask functions allows evaluating the infinite series (*i.e.,* the scalar products) analytically.

IV. Two-users FMPC Construction

We present the mathematical constructions behind FMPC by illustrating a two-users computation protocol; Section V provides a concrete instantiation of this construction.

A. Mathematical Construction

Figure 1 presents a two-users FMPC computation. We consider two users, *Alice* holding a secret input a and *Bob* holding a secret input b, wishing to compute the product ab without revealing their secret inputs. The protocol operates on the public parameters l and q (with $0 < q < 1$); and on the two parametric functions $\phi_{\tau'}, \psi_{\sigma'} \in \mathbb{L}^2[-l, l]$ whose parameters are generated by the trusted authority *Trusty*; we refer to those functions as *mask functions*. The protocol is divided in two phases: an *offline phase* consisting of pre-computations that can be performed ahead of time as it is independent on the secret inputs, and an *online phase* producing the output ab.

a) Offline phase: We model the offline phase as executed by a trusted authority *Trusty* (Section IV-B shows how to distribute the offline phase). *Trusty* generates at random τ and σ, and computes the *normalization coefficient* η given by

$$\eta^{-1}(\tau, \sigma, l) = \frac{1}{l} \int_{-l}^{l} \phi_\tau(x)\psi_\sigma(x)dx \tag{10}$$

and computes the following *normalized mask-functions*:

$$\widetilde{\phi}_\lambda(x) = \eta^q \phi_\tau(x) \quad \widetilde{\psi}_\lambda(x) = \eta^{1-q}\psi_\sigma(x)$$

where λ indicates the set of parameters $\lambda = (\tau, \sigma, l, q)$ (❶). Contrarily to traditional secret-sharing protocols like SPDZ [3], FMPC pushes the complexity at the edges by offloading the offline phase to the users.

b) Online phase: Trusty sends $\tilde{\phi}_\lambda$ to *Alice* and $\tilde{\psi}_\lambda$ to *Bob*, who respectively compute f and g:

$$f(x) = a\widetilde{\phi}_\lambda(x) \quad ; \quad g(x) = b\widetilde{\psi}_\lambda(x) \tag{11}$$

Alice computes the vectors $\mathbf{A}$ and $\mathbf{B}$ from $f(x)$, and *Bob* computes the vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ from $g(x)$ as defined by Equations (2) and (6) (❷). *Alice* sends $\mathbf{A}$ to $node_1$ and $\mathbf{B}$ to $node_2$; and *Bob* sends $\boldsymbol{\alpha}$ to $node_1$ and $\boldsymbol{\beta}$ to $node_2$. As a result, $node_1$ gathers the constant and cosine component of the Parseval's identity, and $node_2$ gathers the sine component of the Parseval's identity (❷). $Node_1$ outputs $(\boldsymbol{A}\cdot\boldsymbol{\alpha})$, and $node_2$ outputs $(\boldsymbol{B}\cdot\boldsymbol{\beta})$ (❹); anyone can compute $(\boldsymbol{A}\cdot\boldsymbol{\alpha})+(\boldsymbol{B}\cdot\boldsymbol{\beta}) = ab$ according to Equation (7). The intuition behind the scheme is to decompose the product ab into two components that are eventually added together to compute the final result; this reduces the problem of multiplication of secret to an addition, which is enabled by Parseval's identity. Section V presents an end-to-end example, with practical choices of mask functions.

B. Decentralization of the Offline Phase

We do not innovate on the offline phase, and rely on existing established solutions. The offline phase of FMPC randomly generates the parameters of the mask functions and computes the normalization coefficient. FMPC may employ the same technique used by SPDZ [3] to generate multiplicative triples, which relies on somewhat homomorphic encryption; despite the simplicity of this approach, it incurs expensive public key cryptography and may lead to high cost. Mascot [5] overcomes this limitation by using oblivious transfer to generate the triples values during the offline phase. Section V-D shows how to use the offline phase of those protocols to instantiate a practical FMPC computation. Alternatively, FMPC may rely on a semi-trusted authority to run the offline phase; the authority is then trusted to correctly generate those parameters and to not collude with the nodes, but never learns any information about the users inputs.

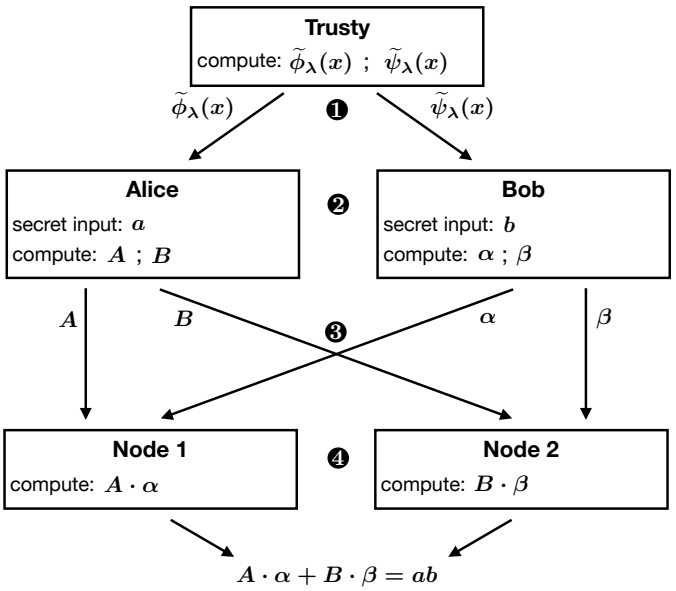

Fig. 1. Overview of FMPC execution. *Trusty* sends $\widetilde{\phi}_\lambda(x)$ to *Alice* and $\widetilde{\psi}_\lambda(x)$ to *Bob* (❶). *Alice* computes $\mathbf{A}$ and $\mathbf{B}$, and *Bob* computes $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ according to Equation (6) (❷); *Alice* sends $\mathbf{A}$ to $node_1$ and $\mathbf{B}$ to $node_2$, and *Bob* sends $\boldsymbol{\alpha}$ to $node_1$ and $\boldsymbol{\beta}$ to $node_2$ (❸). $Node_1$ outputs $(\boldsymbol{A}\cdot\boldsymbol{\alpha})$ and $node_2$ outputs $(\boldsymbol{B}\cdot\boldsymbol{\beta})$ (❹); anyone can compute $\boldsymbol{A}\cdot\boldsymbol{\alpha} + \boldsymbol{B}\cdot\boldsymbol{\beta} = ab$ according to Equation (7).

V. Instantiation of Two-users FMPC Computation

We illustrate a practical example of FMPC computation considering the following mask-functions:

$$\begin{aligned} \phi_\tau(x) &= \tau_1 \sin(\tau_2 x) + \tau_3 \cos(\tau_4 x) \\ \psi_\sigma(x) &= \sigma_1 \sin(\sigma_2 x) + \sigma_3 \cos(\sigma_4 x) \end{aligned} \tag{12}$$

for parameters $\tau = (\tau_1, \tau_2, \tau_3, \tau_4)$ and $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$. For simplicity, we set $q = 1/2$, and

$$\tau_2 = \sigma_4 = \frac{\pi}{4l} \quad ; \quad \tau_4 = \sigma_2 = \frac{3\pi}{4l} \tag{13}$$

to obtain the following normalized mask-functions:

$$\begin{aligned} \widetilde{\phi}_\tau(x) &= \eta^{1/2}\Big(\tau_1 \sin\Big(\frac{\pi}{4l}x\Big) + \tau_3 \cos\Big(\frac{3\pi}{4l}x\Big)\Big) \\ \widetilde{\psi}_\sigma(x) &= \eta^{1/2}\Big(\sigma_1 \sin\Big(\frac{3\pi}{4l}x\Big) + \sigma_3 \cos\Big(\frac{\pi}{4l}x\Big)\Big) \\ \text{with} \quad \eta &= \frac{\pi}{2(\tau_1\sigma_1 + \tau_3\sigma_3)} \end{aligned} \tag{14}$$

A. Protocol Execution

We show how the protocol illustrated in Figure 1 executes using the mask functions given by Equation (12).

Algorithm 1 FMPC example computation – Offline phase

procedure TRUSTY
  $\tau_1, \tau_3, \sigma_1, \sigma_3 \leftarrow random$ from $\mathbb{F}_{p^m}$
  compute $\eta = \frac{\pi}{2(\tau_1\sigma_1+\tau_3\sigma_3)}$ mod N
  send (τ_1, τ_3, η) to *Alice*
  send $(\sigma_1, \sigma_3, \eta)$ to *Bob*

Algorithm 2 FMPC example computation – Online phase

procedure ALICE(τ_1, τ_3, η)
  compute $a_0 = \frac{4\sqrt{2}}{3\pi} a\tau_3\eta^{1/2}$ mod N
  compute $\widehat{a_0} = \frac{4\sqrt{2}}{3\pi} a\tau_1\eta^{1/2}$ mod N
  send a_0 to $node_1$ and $\widehat{a_0}$ to $node_2$
procedure BOB(σ_1, σ_3, η)
  compute $\alpha_0 = \frac{4\sqrt{2}}{\pi} b\sigma_3\eta^{1/2}$ mod N
  compute $\widehat{\alpha}_0 = \frac{4\sqrt{2}}{\pi} b\sigma_1\eta^{1/2}$ mod N
  send α_0 to $node_1$ and $\widehat{\alpha}_0$ to $node_2$
procedure NODE1(a_0, α_0)
  output $s_1 = (3\pi/16)a_0\alpha_0$ mod N
procedure NODE2($\widehat{a_0}, \widehat{\alpha}_0$)
  output $s_2 = (3\pi/16)\widehat{a}_0\widehat{\alpha}_0$ mod N

a) Offline phase: Algorithm 1 illustrates the offline phase; *Trusty* generates at random $(\tau_1, \tau_3, \sigma_1, \sigma_3)$; computes the normalization coefficients η; and sends (τ_1, τ_3, η) to *Alice* and $(\sigma_1, \sigma_3, \eta)$ to *Bob* (❶).

b) Online phase: Algorithm 2 illustrates the online phase; *Alice* computes $(a_0, \widehat{a}_0)$, and *Bob* computes $(\alpha_0, \widehat{\alpha}_0)$ (❷). *Alice* sends a_0 to $node_1$ and $\widehat{a}_0$ to $node_2$; *Bob* sends α_0 to $node_1$ and $\widehat{\alpha}_0$ to $node_2$ (❸). $Node_1$ has all information it needs to compute a_n and α_n (see Equation (15) of Section V-B)—those can be computed from the mere knowledge of a_0 and α_0—and outputs $s_1 = a_0\alpha_0/2 + \sum_{n=1}^{\infty} a_n\alpha_n$. In practice, $node_1$ only evaluates and outputs $s_1 = (3\pi/16)a_0\alpha_0$ (see Equation (17) of Section V-B). Similarly, $node_2$ has all information to compute b_n and β_n (those can be computed from $\widehat{a}_0$ and $\widehat{\alpha}_0$), and outputs $s_2 = \sum_{n=1}^{\infty} b_n\beta_n$; in practice, $node_2$ simply outputs $s_2 = (3\pi/16)\widehat{a}_0\,\widehat{\alpha}_0$ (see Equation (18) of Section V-B). Anyone can compute $s_1 + s_2 = ab$, which follows from Equation (7) (❹).

All operations are performed over a finite field $\mathbb{F}_{p^m}^N$ where p is prime, m is integer and $N = p^m - 1$; addition, multiplication, and the modular inverse are implemented by modular arithmetic $mod\ N$, that is $0 \leq n \leq N-1$. These operations use the equations recalled in Section III, and the long version of the paper[3] recalls how to write them numerically over $\mathbb{F}_{p^m}^N$.

B. Correctness of the Computation

We compute the normalization coefficients and the normalized mask functions according to Equation (10) and (11); and the functions $f(x)$ and $g(x)$ according to Equation (11). All computations are performed using Wolfram Mathematica[4] 11.2, we release our script as open source[5]. The Fourier coefficients $(a_0,\ a_n,\ b_n)$ and $(\alpha_0,\ \alpha_n,\ \beta_n)$ are then given below (for $n = 1, 2, \cdots$):

$$a_0 = \frac{4\sqrt{2}}{3\pi} a\tau_3\eta^{1/2} \quad ; \quad a_n = \frac{9(-1)^n}{9-16n^2} a_0 \tag{15}$$
$$b_n = \frac{12(-1)^n n}{1-16n^2}\widehat{a}_0$$
$$\alpha_0 = \frac{4\sqrt{2}}{\pi} b\sigma_3\eta^{1/2} \quad ; \quad \alpha_n = \frac{(-1)^n}{1-16n^2}\alpha_0$$
$$\beta_n = \frac{4(-1)^n n}{9-16n^2}\widehat{\alpha}_0$$

where:

$$\widehat{a}_0 = \frac{4\sqrt{2}}{3\pi} a\tau_1\eta^{1/2} \quad ; \quad \widehat{\alpha}_0 = \frac{4\sqrt{2}}{\pi} b\sigma_1\eta^{1/2} \tag{16}$$
$$\eta = \frac{\pi}{2(\tau_1\sigma_1 + \tau_3\sigma_3)}$$

We can easily check Parseval's identity; $node_1$ computes

$$\frac{a_0\alpha_0}{2} + \sum_{n=1}^{\infty} a_n\alpha_n = \tag{17}$$
$$a_0\alpha_0\Big(\frac{1}{2} + \frac{9}{256}\sum_{n=1}^{\infty}\frac{1}{[(3/4)^2 - n^2][(1/4)^2 - n^2]}\Big) = \frac{3\pi}{16}a_0\alpha_0$$

and $node_2$ computes

$$\sum_{n=1}^{\infty} b_n\beta_n = \tag{18}$$
$$\frac{3}{16}\widehat{a}_0\widehat{\alpha}_0\sum_{n=1}^{\infty}\frac{n^2}{[(3/4)^2 - n^2][(1/4)^2 - n^2]} = \frac{3\pi}{16}\widehat{a}_0\widehat{\alpha}_0$$

Equation (17) and Equation (18) are computed by evaluating the convergent sums given by Equation (8) of Section III. By adding Equation (17) to Equation (18), we finally get:

$$\frac{a_0\alpha_0}{2} + \sum_{n=1}^{\infty} a_n\alpha_n + \sum_{n=1}^{\infty} b_n\beta_n = \frac{3\pi}{16}(a_0\alpha_0 + \widehat{a}_0\widehat{\alpha}_0) = ab \tag{19}$$

C. Security Analysis

We show that no adversary can retrieve the secret inputs a and b from the knowledge of $(a_0, \alpha_0, \widehat{a}_0, \widehat{\alpha}_0, \eta)$. We assume passive adversaries; *i.e.,* they follow the protocol specification but try to learn more than allowed (see Section II). Informally, the adversary possesses five equations, *i.e.,* the expressions of $(a_0, \alpha_0, \widehat{a}_0, \widehat{\alpha}_0, \eta)$, and six unknown, *i.e.,* $(a, b, \tau_1, \tau_3, \sigma_1, \sigma_3)$. The adversary thus holds fewer equations than unknowns, which make it information-theoretically impossible to recover any unknown value. Theorem 1 presents this result formally.

Theorem 1. *The scheme presented in Section V-A achieves perfect secrecy against a passive adversary holding $\theta = (a_0, \alpha_0, \widehat{a}_0, \widehat{\alpha}_0, \eta) \in \mathbb{F}_{p^m}$; i.e., for all distribution of $a, b \in \mathbb{F}_{p^m}$ and for all $\theta = (a_0, \alpha_0, \widehat{a}_0, \widehat{\alpha}_0, \eta) \in \mathbb{F}_{p^m}$, we have $Pr[a|\theta] = Pr[a]$ and $Pr[b|\theta] = Pr[b]$.*

The proof can be found in the long version of the paper[6].

This implies that nodes are not able to recover the users inputs even if they collude (but multiple nodes are still required to handle additions of secrets, as in SPDZ [3]).

[3] https://arxiv.org/abs/1912.02583
[4] http://www.wolfram.com/mathematica/
[5] https://gist.github.com/asonnino/7d3abd570736d13bddf61fa429692983
[6] https://arxiv.org/abs/1912.02583

D. Discussion

We discuss convenient choice of mask functions, distribution of the offline phase, and extension to multiple nodes.

a) Convenient choice of mask functions: Even though FMPC applies to any kind of square-integrable functions, a convenient choice of family of mask functions (in the case of two users) is $\{\xi_1^{(j)}\sin(\xi_2^{(j)}x) + \xi_3^{(j)}\cos(\xi_4^{(j)}x)\}$ where parameters $\xi_i^{(j)}$ are randomly chosen (with $i = 1,\ 2,\ 3,\ 4$ and users $j = 1, 2$). The parameters $\xi_2^{(j)}$ and $\xi_4^{(j)}$ (with $j = 1, 2$) are public, and it is convenient to set them to $\xi_2^{(1)} = \xi_4^{(2)} = \pi/(4l)$ and $\xi_4^{(1)} = \xi_2^{(2)} = (3\pi)/(4l)$ (see Equation (14)). The main advantage of this family of mask-functions is that they forgo the need to resort to numerical calculations to compute the contributions of Parseval's identity—calculating the numerical sums of the Parseval's identity is never needed—users simply evaluate them using the analytic expressions provided in Section III-C. We can easily observe that it is possible to select mask-functions allowing to perform all calculations analytically even for a large number of users; mask functions composed of sums of sine and cosine ensures convergence, and can be evaluated using expressions similar to those given in Section III-C.

b) Distribution of the offline phase: Established protocols like SPDZ require the generation of multiplicative triplets during the offline phase; *i.e.,* they provide a functionality to generate three elements (x, y, z) such that $xy = z$ in a distributed manner. FMPC may execute twice this functionality to generate (τ_1, σ_1) such that $\tau_1\sigma_1 = \eta_1$, and (τ_3, σ_3) such that $\tau_3\sigma_3 = \eta_3$; and then simply compute:

$$\eta = \frac{\pi}{2(\eta_1 + \eta_3)} = \frac{\pi}{2(\tau_1\sigma_1 + \tau_3\sigma_3)} \tag{20}$$

VI. Extension to Multiple Players

We introduces the first generalization of Parseval's identity for Fourier series applicable to an arbitrary number of inputs, and uses it to extend the two-user computation scheme presented in Section IV to an arbitrary number of users.

A. Generalization of Parseval's Identity

We present the generalization of Parseval's identity for Fourier series applicable to n inputs. Parseval's identity traditionally applies only to two functions; we overcome this drawback by using the convolution operation between two functions. We illustrate Parseval's identity for three inputs, which can easily be generalized for an arbitrary number of inputs. Section VI-B leverages these considerations to build the n-users FMPC protocol.

Firstly we observe that in the case of two users, Parseval's identity may be cast into the following form

$$\frac{1}{2}a_0\alpha_0 + \frac{1}{2}(\boldsymbol{A}+\boldsymbol{B})\cdot(\boldsymbol{\alpha}+\boldsymbol{\beta}) + \frac{1}{2}(\boldsymbol{A}-\boldsymbol{B})\cdot(\boldsymbol{\alpha}-\boldsymbol{\beta}) = \frac{1}{l}\int_{-l}^{l} f(x)g(x)dx \tag{21}$$

Let's now consider three inputs, $f(x)$, $g(x)$ and $h(x)$ with Fourier series representations given by Equation (1) and by

$$h(x) = \frac{\gamma_0}{2} + \sum_{n=1}^{\infty}\gamma_n\cos\big(\frac{n\pi}{l}x\big) + \sum_{n=1}^{\infty}\varrho_n\sin\big(\frac{n\pi}{l}x\big) \tag{22}$$

respectively; the generalized Parseval's identity reads:

$$\begin{aligned}&\frac{1}{2}a_0\alpha_0\gamma_0 + \frac{1}{2}(\boldsymbol{A}+\boldsymbol{B})\cdot(\boldsymbol{\alpha}+\boldsymbol{\beta})\cdot(\boldsymbol{\gamma}+\boldsymbol{\varrho}) + \\ &\frac{1}{2}(\boldsymbol{A}-\boldsymbol{B})\cdot(\boldsymbol{\alpha}-\boldsymbol{\beta})\cdot(\boldsymbol{\gamma}-\boldsymbol{\varrho}) = \\ &\frac{1}{l}\int_{-l}^{l} f(x)(g\star h)(x)dx + \frac{1}{l}\int_{-l}^{l} g(x)(f\star h)(x)dx + \\ &\frac{1}{l}\int_{-l}^{l} h(x)(f\star g)(x)dx - \frac{2}{l}\int_{-l}^{l}\hat{h_c}(x)(\hat{f}_c\star\hat{g}_c)(x)dx\end{aligned} \tag{23}$$

or

$$\begin{aligned}&\frac{a_0\alpha_0\gamma_0}{2} + \frac{1}{2}\sum_{n=1}^{\infty}(a_n+b_n)(\alpha_n+\beta_n)(\gamma_n+\varrho_n) + \\ &\frac{1}{2}\sum_{n=1}^{\infty}(a_n-b_n)(\alpha_n-\beta_n)(\gamma_n-\varrho_n) = \\ &\frac{1}{l}\int_{-l}^{l} f(x)(g\star h)(x)dx + \\ &\left(\frac{1}{l}\int_{-l}^{l} g(x)(f\star h)(x)dx - \frac{1}{l}\int_{-l}^{l}\hat{g_c}(x)(\hat{f}_c\star\hat{h_c})(x)dx\right) + \\ &\left(\frac{1}{l}\int_{-l}^{l} h(x)(f\star g)(x)dx - \frac{1}{l}\int_{-l}^{l}\hat{h_c}(x)(\hat{f}_c\star\hat{g}_c)(x)dx\right)\end{aligned} \tag{24}$$

Vectors $\mathbf{A}$, $\mathbf{B}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ and $\boldsymbol{\varrho}$ are respectively defined as

$$\begin{aligned}\mathbf{A} = \{a_n\} \ ;\ \mathbf{B} = \{b_n\} \quad (n = 1, 2, \cdots)\\ \boldsymbol{\alpha} = \{\alpha_n\} \ ;\ \boldsymbol{\beta} = \{\beta_n\} \quad (n = 1, 2, \cdots)\\ \boldsymbol{\gamma} = \{\gamma_n\} \ ;\ \boldsymbol{\varrho} = \{\varrho_n\} \quad (n = 1, 2, \cdots)\end{aligned} \tag{25}$$

and

$$\begin{aligned}&\hat{f}_c(x) \equiv \frac{1}{2}\Big(f(x) + f(-x)\Big) \ ;\ \hat{g}_c(x) \equiv \frac{1}{2}\Big(g(x) + g(-x)\Big)\\ &\hat{h}_c(x) \equiv \frac{1}{2}\Big(h(x) + h(-x)\Big)\end{aligned} \tag{26}$$

We simply include γ_0, $(\boldsymbol{\gamma}+\boldsymbol{\varrho})$, and $(\boldsymbol{\gamma}-\boldsymbol{\varrho})$ in the left-hand side of the equation, and adapt the right-end side to match calculations. A mathematical formula for an arbitrary number of inputs can easily be obtained following the same logic.

B. Mathematical Construction

We extend the two-user FMPC scheme presented in Section IV to a scheme supporting n-users.

a) Offline phase: Trusty generates at random the parameters of n mask-functions $(\phi_1, \cdots, \phi_n)$; it then computes the n normalization coefficients similarly to Equation (10), and uses them to compute the normalized mask-functions $(\widetilde{\phi_1}, \cdots, \widetilde{\phi_n})$ as shown in Equation (11). This is analogue to the offline phase of the protocol presented in Section IV-A, except that we now consider n mask-functions instead of two.

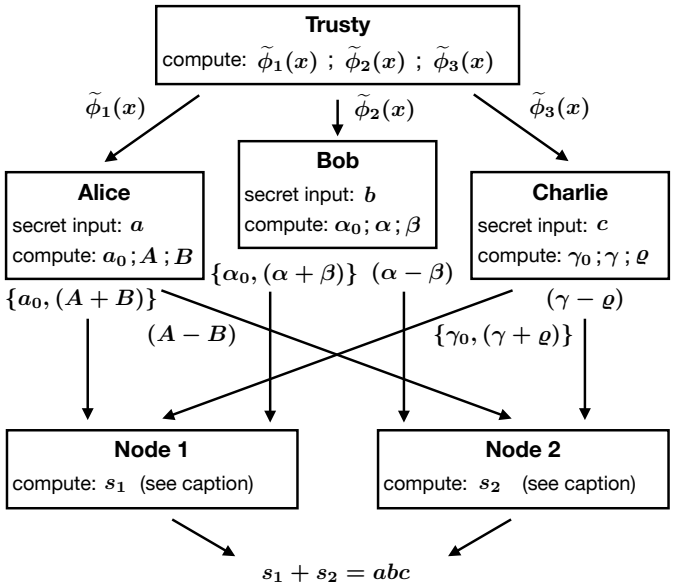

Fig. 2. Example execution of FMPC for 3 users and 2 nodes. Each user receives a normalized mask function from *Trusty*. *Alice* computes and sends $\{a_0, (\mathbf{A}+\mathbf{B})\}$ to $node_1$ and $(\mathbf{A}-\mathbf{B})$ to $node_2$; *Bob* sends $\{\alpha_0, (\boldsymbol{\alpha}+\boldsymbol{\beta})\}$ to $node_1$ and $(\boldsymbol{\alpha}-\boldsymbol{\beta})$ to $node_2$; and *Charlie* sends $\{\gamma_0, \boldsymbol{\gamma}+\boldsymbol{\varrho}\}$ to $node_1$ and $(\boldsymbol{\gamma}-\boldsymbol{\varrho})$ to $node_2$. $Node_1$ outputs $s_1 = 1/2(a_0\alpha_0\gamma_0) + 1/2(\boldsymbol{A}+\boldsymbol{B})\cdot(\boldsymbol{\alpha}+\boldsymbol{\beta})\cdot(\boldsymbol{\gamma}+\boldsymbol{\varrho})$, and $node_2$ outputs $s_2 = 1/2(\boldsymbol{A}-\boldsymbol{B})\cdot(\boldsymbol{\alpha}-\boldsymbol{\beta})\cdot(\boldsymbol{\gamma}-\boldsymbol{\varrho})$; according to Equation (23), anyone can compute $s_1 + s_2 = abc$.

b) Online phase: Trusty sends a normalized mask function to each user; they compute $(f_1, \cdots, f_n)$ using their secret inputs $(a_1, \cdots, a_n)$, and their Fourier coefficients according to Equation (11) and Equation (2). Similarly to Section IV-A, users send the constant and cosine component of Parseval's identity to $node_1$, and the sine component to $node_2$; therefore the protocol can always be executed with two nodes. Each node then computes and outputs the scalar product of the users coefficients vector, and the product $\prod_{i=1}^{n} a_i$ is computed by summing the output of each node according to the generalized Parseval's identity presented in Section VI-A.

Figure 2 shows an example of execution of FMPC for three users. Each user, *Alice*, *Bob* and *Charlie* receives a normalized mask-function from *Trusty*. In this case the normalization coefficient is given by

$$\eta^{-1} = \frac{1}{l}\int_{-l}^{l}\Big(\phi_1(x)(\phi_2 \star \phi_3)(x) + \phi_2(x)(\phi_1 \star \phi_3)(x) + \phi_3(x)(\phi_1 \star \phi_2)(x) - 2\hat{\phi}_{3c}(x)(\hat{\phi}_{1c} \star \hat{\phi}_{2c})(x)\Big)dx \tag{27}$$

with

$$\hat{\phi}_{ic} \equiv \frac{1}{2}\Big(\phi_i(x) + \phi_i(-x)\Big) \quad \text{with} \quad (i = 1, 2, 3) \tag{28}$$

and the three normalized mask-functions read

$$\widetilde{\phi}_1(x) = \eta^{q_1}\phi_1(x)\ \ \widetilde{\phi}_2(x) = \eta^{q_2}\phi_2(x)\ \ \widetilde{\phi}_3(x) = \eta^{1-q_1-q_2}\phi_3(x)$$

where q_1 and q_2 are two positive real numbers subject to the condition $0 < q_1 + q_2 < 1$. *Alice* locally computes $\{a_0, \mathbf{A}\}$ and $\mathbf{B}$; *Bob* computes $\{\alpha_0, \boldsymbol{\alpha}\}$ and $\boldsymbol{\beta}$; and *Charlie* computes $\{\gamma_0, \boldsymbol{\gamma}\}$ and $\boldsymbol{\varrho}$ similarly to Equation (2) and Equation (6). *Alice* sends $\{a_0, \mathbf{A}+\mathbf{B}\}$ to $node_1$ and $(\mathbf{A}-\mathbf{B})$ to $node_2$; *Bob* sends $\{\alpha_0, (\boldsymbol{\alpha}+\boldsymbol{\beta})\}$ to $node_1$ and $(\boldsymbol{\alpha}-\boldsymbol{\beta})$ to $node_2$; and *Charlie* sends $\{\gamma_0, (\boldsymbol{\gamma}+\boldsymbol{\varrho})\}$ to $node_1$ and $(\boldsymbol{\gamma}-\boldsymbol{\varrho})$ to $node_2$. Finally, $node_1$ outputs $s_1 = \frac{1}{2}a_0\alpha_0\gamma_0 + \frac{1}{2}(\boldsymbol{A}+\boldsymbol{B})\cdot(\boldsymbol{\alpha}+\boldsymbol{\beta})\cdot(\boldsymbol{\gamma}+\boldsymbol{\varrho})$, and $node_2$ outputs $s_2 = \frac{1}{2}(\boldsymbol{A}-\boldsymbol{B})\cdot(\boldsymbol{\alpha}-\boldsymbol{\beta})\cdot(\boldsymbol{\gamma}-\boldsymbol{\varrho})$; following Equation (23), anyone can compute $s_1 + s_2 = abc$.

VII. Limitations and Future Work

FMPC has several limitations that are beyond the scope of this work, and deferred to future work. FMPC (*i*) does not support composition of operations. That is, while most established scheme [3], [2], [5] can evaluate expressions like $(a+b)(c+d)$ with two additions and one multiplication, FMPC needs to distribute the operation and evaluate $(ac+ad+bc+bd)$. This limitation is problematic for large computations and makes FMPC suitable only to evaluate circuits with a relatively small number of multiplications. Other limitations are (*ii*) that the security and efficiency of the scheme rely on the choice of the mask functions. We also defer as future work (*iv*) adapting our scheme to withstand active adversaries, potentially adapting the MAC-based approach introduced by SPDZ [3].

VIII. Conclusions

FMPC is a novel secret-sharing multiparty computation protocol of arithmetic circuits that requires no online communication between nodes to compute multiplication of secrets; FMPC innovates on the online phase by applying Fourier series to Parseval's identity. FMPC enjoys of constant latency in the size of the circuit, but is only suitable to evaluate low-depth circuits. We introduce the first generalization of Parseval's identity for Fourier series applicable to an arbitrary number of inputs, and use it to allow FMPC to operate on an arbitrary number of inputs. FMPC paves the way for new kind of multiparty computation protocols, hopefully encouraging discussions and spurring new directions to explore.

Acknowledgements

This work was supported by the EU H2020 DECODE project under grant agreement number 732546 as well as `chainspace.io`. We thank George Danezis for helpful suggestions on early manuscript and valuable advice, and Ioannis Psaras for comments and proofreading.

References

[1] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Proceedings of Crypto*, 1991.

[2] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. 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.

[3] I. Damgård, V. Pastro, N. Smart, and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in *Proceedings of Crypto*, 2012.

[4] I. S. Gradshteyn and I. M. Ryzhik, *Table of integrals, series, and products*. Academic press, 2014.

[5] M. Keller, E. Orsini, and P. Scholl, "MASCOT: faster malicious arithmetic secure computation with oblivious transfer," in *Proceedings of the SIGSAC Conference on Computer and Communications Security*, 2016.