# Application of Randomness for Security and Privacy in Multi-Party Computation

Rahul Saha[1,2], Gulshan Kumar[1,2], G. Geetha[3], Mauro Conti[1], William J Buchanan[4]

***Abstract*—A secure Multi-Party Computation (MPC) is one of the distributed computational methods, where it computes a function over the inputs given by more than one party jointly and keeps those inputs private from the parties involved in the process. Randomization in secret sharing leading to MPC is a requirement for privacy enhancements; however, most of the available MPC models use the trust assumptions of sharing and combining values. Thus, randomization in secret sharing and MPC modules is neglected. As a result, the available MPC models are prone to information leakage problems, where the models can reveal the partial values of the sharing secrets.**

**In this paper, we propose the first model of utilizing a random function generator as an MPC primitive. More specifically, we analyze our previous development of the Symmetric Random Function Generator (SRFG) for information-theoretic security, where the system is considered to have unconditional security if it is secure against adversaries with unlimited computing resources and time. Further, we apply SRFG to eradicate the problem of information leakage in the general MPC model. Through a set of experiments, we show that SRFG is a function generator that can generate the combined functions (combination of logic GATEs) with $n/2$ -private to $n$-private norms. As the main goal of MPC is privacy preservation of the inputs, we analyze the applicability of SRFG properties in secret sharing and MPC and observe that SRFG is eligible to be a cryptographic primitive in MPC developments. We also measure the performance of our proposed SRFG-based MPC framework with the other randomness generation-based MPC frameworks and analyze the comparative attributes with the state-of-the-art models. We observe that our posed SRFG-based MPC is $\approx 30\%$ better in terms of throughput and also shows $100\%$ privacy attainment.**

***Index Terms*—Cryptography, Privacy, Security, Information, Randomness, Function, Computation, Multi-party**

## I. Introduction

The paradigm of the Internet of Things (IoTs) drives the present age of technology. The present computing paradigm shifted from centralized to decentralization and now, multi-party computing is showing its potential in every domain of applications. Multi-Party Computation (MPC) is a cryptographic tool that allows multiple parties to compute a function using their combined data, without revealing their input. With the increasing demand for IoT applications, the security and privacy of the data become the primary focus of security protocols and methods. Even though Yao's seminal work introduced MPC in 1982, the increasing demand for IoT applications and distributed architecture of computing resulted in a re-birth of MPC's popularity [1]. The adoption of MPC in IoT provides the required processing power to provide smart services in the shortest time. In the following, we discuss the basic understanding of MPC functionality and information-theoretic security in MPC connection.

### A. MPC functionality

In the context of digital assets, MPC replaces individual private keys for the signing of transactions [2]. MPC distributes the signing process among multiple systems. Each system possesses a piece of private data representing a share of the key (secret), and together they cooperate to sign transactions in a distributed way. There are several possible applications of this technology. Being able to store secrets with different systems, rather than in a single system leading to the central failure, would make the computation much harder for a potential intruder. By using multiparty computation, legitimate users can still use relevant information to verify the legitimacy of the operation.

In an MPC, we consider a given number of participants as $P_1, P_2, ..., P_N$. Each participant has a private data: $D_1, D_2, ..., D_N$. All the participants want to compute the value of a public function on that private data given as $F(D_1, D_2, ..., D_N)$ while keeping their own inputs secret. We can show a mathematical notion of MPC as in Equation 8. In this equation, $F$ denotes a public function and $\bigcirc$ represents any operation in the function $F$. We show a logical representation of MPC in Figure 1. The figure shows that MPC consists of three functions: sharing, computation, and reconstruction. In sharing, the participants share their private data with the public function. In computation, the public function computes with the given private inputs. In reconstruction, the output is again shared with all the participants, where the output is public without inferring the knowledge about the input private data. Note that, the public function considers the private data of all the participants and jointly computes to output the result [3] [4]. The general expression is shown in Equation 1.

$$\{P_1, P_2, ..., P_N\} \rightarrow \{D_1, D_2, ..., D_N\} = F(D_1 \bigcirc D_2 \bigcirc, ..., \bigcirc D_N). \tag{1}$$

There are two basic properties, which an MPC protocol aims to ensure: input privacy and correctness.

*a) Input privacy:* The inputs given to the public function $F$ are private to the individual. No information about the private data held by the participants is inferential from the messages sent during the execution of the MPC protocol. The

[1]Department of Mathematics, University of Padua, Italy
[2]School of Computer Science and Engineering, Lovely Professional University, India
[3] School of Computer Science and Information Technology, Jain Deemed to be University, India
[4] Blockpass ID Lab, Edinburgh Napier University, United Kingdom

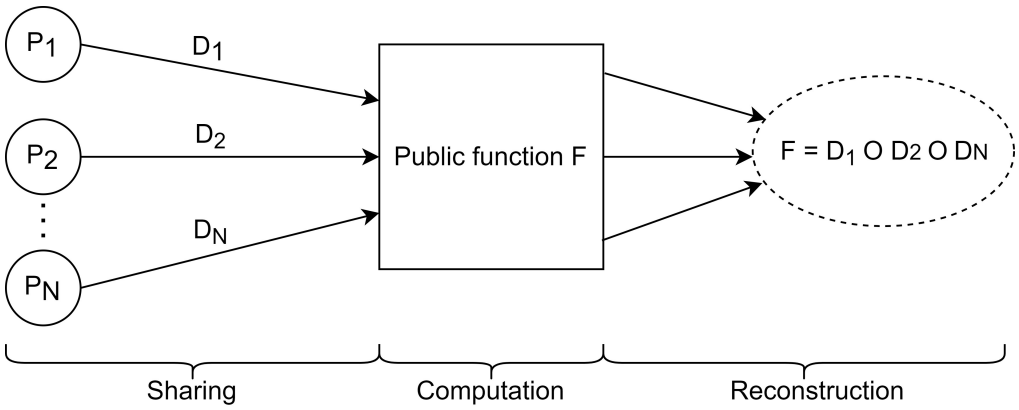


Fig. 1. Logical interpretation of MPC

maximum information that can be inferred from an MPC is only by observing the output of the public function.

*b) Correctness:* The essence of this property of MPC lies in the impact of the adversarial colluding parties. In such a real-world scenario with the adversaries, the correctness property uses two dimensions: either the honest parties are guaranteed to compute the correct output, or they abort if they find an error. We call the former one a "robust" protocol and the latter one is an MPC protocol with abort.

### B. Information theoretic security

A cryptosystem is information-theoretic secure if the system is secure against adversaries with unlimited computing resources and time [5]. Information-theoretically secure protocols are resistant to future developments in computing, more specifically for quantum-safe cryptographic protocols [6]. Different cryptographic tasks use information-theoretic security in a meaningful way to accomplish some useful security requirements. Secret sharing schemes such as Shamir's are information-theoretically secure with a threshold number of shares. Secure MPC protocols often have information-theoretic security. Information-theoretic privacy for the user's query in Private Information Retrieval (PIR) is another dimension of research. Reductions between cryptographic primitives urge the need for achieving information-theoretic security. Symmetric encryption can be constructed under an information-theoretic notion of security called entropic security, which assumes that the adversary knows almost nothing about the message being sent.

We show a mathematical understanding of information-theoretic security in the following. We consider a adversary game protocol $\pi$, where $x$ is an integer within $[0, n-1]$. We execute a game of random bit selection $b$ and try to identify whether $b$ is 0 or 1. If $b = 0$, we send a random value $r \leftarrow X$ to the (unbounded) adversary. We sample $r$ from a set $R$. If $b = 1$, we sample a random value $r \leftarrow R$ and send $x + r$ to the adversary. We fix a security parameter $k$ and consider that $\pi$ has the perfect security if the adversary has a probability of exactly $1/2$ of guessing the value of $b$ given the input. On the other hand, the protocol has information-theoretic security if the adversary has the probability $1/2 + \mu(k)$ of guessing the value of $b$, where $\mu$ is a negligible error function. From the entropy point of view, we can show that information-theoretic security must follow the condition $H(K|C) < H(K)$, where $H()$ is the entropy function, $K$ is the information of key, and $C$ is the information of cipher text.

As MPC avoids the signing of encryption keys, the secret sharing with multiple systems must obtain information-theoretic security. Previous research also emphasize on this particular requirement of MPC and analyzes secure MPC to have a certain amount of information-theoretic security [7].

### C. Motivation and contribution

Even though information-theoretic security and MPC are available in existing research work dated back to the 1990s, the popularity of these two aspects revokes the progress of IoTs and their various applications. Besides, the researchers also negotiate with the construction of a set of binary strings $R$ to where the functions that try to fool an MPC protocol are efficient. However, the randomness from a non-Boolean circuit is also a question to solve for information-theoretic security. On the other hand, our previously developed Symmetric Random Function Generator (SRFG) shows the capability of posing information-theoretic security. A symmetric random function, acting as a random number generator in MPC, ensures the independence and unpredictability of shared random values. This is crucial for preserving privacy in computations where parties collaborate on private inputs. By incorporating the symmetric random function, MPC addresses existing randomness and privacy challenges. The function's pseudorandom outputs obscure patterns and correlations, fortifying the protocol against adversarial attacks. It promotes fairness, consistency, and efficiency across parties, offering a versatile tool to generate secure and unpredictable values, ultimately mitigating privacy concerns within the MPC framework. Thus, the urge for randomness in MPC secret sharing and the scope of SRFG motivate us to analyze the function generator for multi-party computing assuring information-theoretic security. To this end, the major contribution of our present work is as follows.

- **Random function generator for MPC**: Randomization is a requirement in information-theoretic security and privacy in connection with MPC. However, the existing

literature neglects this fact. We pioneer in this direction and use a symmetric random function generator in MPC as a primitive for secret sharing.

- **Information-theoretic security and privacy of a random function**: We show a detailed analysis of the application of the random function generator for the assurance of information theoretic security and privacy. We observe that our random function generator is able to provide a negligible probability of information leakage.
- **$t$-private notion of privacy in random function generator**: We pioneer the analysis of the privacy properties in a symmetric random function generator. The application of this random function generator in MPC shows that the functions from this generator achieve the strength of $n$-private norms for $n$-bit variable.

### D. Organization of the paper

We organize the rest of the paper as follows. Section II shows some contributory works in the domain of information-theoretic security and privacy in connection with MPC. Section III discusses the basic technical construction of SRFG as a cryptographic primitive. Section IV analyzes the properties of SRFG for being adaptive to information-theoretic security and its utilization is MPC secret sharing. Section V analyzes the security and privacy of SRFG applications in secret sharing and MPC. Section VI discusses the experimental methodology and related results. Finally, Section VII concludes our work.

## II. Related work

In this section, we discuss some of the important works in the direction of secret sharing, which is an enabler method in MPC.

Threshold secret sharing or $(t, n)$ threshold secret sharing is a popular use in MPC for providing secrecy and robustness services for various cryptographic protocols. To enhance the security of $n$ distributed shares with $t$ threshold of secrecy, [8] shows a secure secret reconstruction. Alike the traditional secret sharing schemes [9], the proposed work mitigates the outsider attack on the shares by leveraging the need to know all the released shares. However, the limitation of the work is that the method of the secret sharing shown in [8] is unable to prevent the outsiders from learning the secret if the outsiders intercept all the released shares. The authors also address this limitation by making the reconstructed secret accessible only to shareholders, but not to outsiders. A traditional secret sharing has the risk of an adversary without a valid share may obtain the secret when more than $t$ shareholders participate in the secret reconstruction. To mitigate this risk, a Group-Oriented Secret Sharing (GOSS) uses $(t, m, n)$ parameters based on the Chinese remainder theorem [10]. Without any share verification or user authentication, the scheme uses Randomized Components (RCs) to bind all participants into a tightly coupled group, and ensures that the secret is recoverable only if all $m(m \geq t)$ participants in the group have valid shares and release valid RCs honestly. Recent work in this direction of GOSS [11] shows that the group-orientation property in [10] is invalid and concrete attacks on GOSS allow an unauthenticated adversary with no valid share to participate in the reconstruction phase and obtain the secret.

A secure secret sharing scheme based on symmetric bivariate polynomials and its extended version to an asymmetric one is available in recent literature [12]. Secret Sharing Scheme (SSS) is also compatible with a Group-Characterizable (GC) random variable [13], where the scheme works as an entropy function between the group $G$ and the subgroups $G_1, G_2, ..., G_N$. Such discussion also shows that Homomorphic SSS (HSSS) is equivalent to GC-SSS, whose subgroups are normal in the main group $G$.

Verifiable Secret Sharing (VSS) or Verifiable Multi-Secret Sharing (VMSS) is another constituent of secret sharing schemes. We have gone through a survey of such schemes in [14]. Some open problems notified by this survey include broadcast complexity, communication complexity, and lower bounds of the schemes. The work in the paper [15] proposes two VMSS schemes, which by add new validity checks in the verification phase to overcome the problems of malicious behaviour of the dealer. The schemes use XTR public key system and realizes $GF(p^6)$ security by computations in $GF(p^2)$ without explicit constructions of $GF(p^6)$, where $p$ is a prime. The method uses the trace function to provide short parameters for the requirements assuring a high level of security. In addition, the two schemes are dynamic and threshold changeable.

Random functions are playing a great role in secret sharing schemes. For example, a recent work proposes a $(t, n)$–threshold verifiable secret sharing scheme with changeable parameters based on a trapdoor one-way function [16]. This scheme consists of a generation phase, a distribution phase, an encoding phase, and a reconstruction phase. The generation and distribution phases are, respectively, based on Shamir's and Feldman's approaches, while the encoding phase uses a novel trapdoor one-way function. In the reconstruction phase, the shares and reconstructed secrets are validated using a cryptographic hash function. SMPC solutions use random numbers to mask the shared secrets followed by encryption. To ensure correct decryption of the final result, it is required that these random numbers sum to a publicly known value. The work in [17] proposes two novel protocols for joint random number generation with very low computational and communication overhead. The protocol relies on bit-wise sharing of individually generated random numbers, allowing parties to adapt random numbers to yield a public sum. Second, we propose a protocol that uses the sign function to generate a random number from broadcast numbers. Recently, the authors in the paper [18] show a partially synchronous protocol that allows a system of $N$ processes to produce an unpredictable common random number shared by correct participants. The protocol claims to be optimally resilient, as it allows up to $f = b_{N-1}$ the processes to behave arbitrarily and ensures deterministic termination.

Furthermore, we review several current research works related to secret share and multi-party computation to validate that the randomness provided by SRFG and applied to MPC is quite efficient in terms of privacy and computational performance. For example, entangled polynomial codes

do not consistently outperform PolyDot codes in the MPC setting, contrary to their superiority in coded computation. Adaptive Gap Entangled (AGE) polynomial codes for MPC prove their superiority through analysis and simulations in terms of worker requirement and computational, storage, and communication overhead [19]. The work in [20] introduces a novel secure computation approach based on a client-server model, utilizing (k,n) threshold secret sharing to protect inputs distributed across multiple clients. Unlike conventional methods, this approach minimizes communication during secure computation, concentrating communication in the preprocessing phase. Another work shown in [21] introduces a secure multiparty federated learning control system to prevent data leakage in federated learning. It includes secure training and prediction processes where data providers collaborate without revealing local data, and users receive prediction services without accessing the model, ensuring privacy and security for all parties involved. A recent study introduces the problem of private randomness agreement (PRA), where two participants aim to agree on a random string unknown to an adversary, utilizing a public, authenticated channel alongside main channels [22]. It demonstrates that PRA cannot be solved in a single round, but presents efficient solutions requiring three or four rounds, balancing privacy and computational cost. However, communication overhead becomes higher. Another recent work toward randomness introduces Funder, a decentralized randomness solution for proof-of-stake blockchains, leveraging a post-quantum threshold Verifiable Random Function (VRF) [23]. It also presents a compiler for transforming classical VRF solutions into post-quantum VRF using symmetric-key primitives, validated and evaluated with quantum-secure zero-knowledge systems ZKBoo and ZKB++. The drawback in this work is the potential performance overhead introduced by using symmetric-key primitives for achieving post-quantum VRF, which may impact the efficiency and scalability of the solution, particularly in large-scale blockchain networks.

From the above discussion, we observe that the random or arbitrary functions are good candidates for being applicable in secret sharing schemes; however, the existing literature has not emphasized this aspect and is more inclined towards developing the schemes for threshold maintenance. Therefore, the missing link of randomness in functions and resilient secret-sharing schemes motivates us for the present work.

## III. SRFG

In our previous development of the Symmetric Random Function Generator (SRFG) mathematical model [24], we show that SRFG produces balanced and symmetric outputs in terms of the number of 1s and number of 0s in the output string with the variable input patterns. In this section, we recall some important mathematical expressions from [24] and [25] for a clear understanding of SRFG behaviour. We show the generalized mathematical expression for SRFG as in Equation 2.

$$f() = \otimes f_i^L, \tag{2}$$

where $i$ is the number of gates used in SRFG (AND, OR, NOT, XOR gates with randomized selection); $L$ represents the expression length. We defined expression length as the number of the combined terms used in $f$ and $\otimes$ symbolizes the random combination. We show the logical model of SRFG in Figure 2. As randomness is the main criterian for a random number generator, we granulate the SRFG generalized equation with $N$ input variables' following a random selection as shown in Equation 3. As per the shown model of SRFG in the literature [24], it uses a finite field $F_2^{0,1}$ and $\otimes$ denotes an operation on the field. SRFG inputs $N$ variables, each of $n$ bit vector $V = v_1, v_2, ..., v_n$. As the values in the vector is either 1 or 0, $V_i$ is a binary vector. . We show the expanded structure and chaining of functions in Figure 3.

$$f(V_1, V_2...., V_N) = \otimes f_i^L \ [rand(V_1, V_2..., V_N)]. \tag{3}$$

SRFG embodies a set of functions as $B_N$ is. All the functions map the elements from $F_2^N$ into $F_2$, where $F_2^N = \{(V_1, V_2, ..., V_N)|V_i \in F_2\}$. The output of SRFG is a vector, which is a combination of $2^N$ with $N$ variables. This vector comprises of all the values $f(y), y \in F_2^N$ as the variables are randomly selected and represented as a polynomial form of Algebraic Normal Form (ANF) [26]. We show the mathematical interpretation as in Equation 4 with the condition in Equation 5 and Equation 6.

$$\begin{aligned} f(V_1, V_2, ..., V_N) = \otimes \ \lambda_u \ \Big( \prod_{i=1}^{N} rand(V_i)^{u_i} \Big)^L, \\ \lambda_u \in F_2, \ u \in F_2^N \ and \ L \in \mathbb{Z}, \end{aligned} \tag{4}$$

with

$$\lambda_u = \otimes \ f(v), \ v \preceq u \ , \ \forall \ V_i = \{v_{i_1}, v_{i_2}, ...., v_{i_n}\}, \tag{5}$$

where

$$(v_{i_1}, v_{i_2}, ..., v_{i_n}) \preceq (u_1, u_2, ..., u_n) \iff \forall i, j, v_{i_j} \leq u_i \tag{6}$$

In coding theory and in cryptography, ANF considers a boolean function as a multivariate polynomial. As the $N$ inputs contain values in $F_2$, we consider modulo $X^2 + X$. Therefore, the multivariate polynomial construction for SRFG has the degree of at most 1 for each input variable such that any monomial of this polynomial is the product of some input variables. For any $u \in F_2^N$, $(V_i)^{u_i}$ defines monomial as: $\Big( \prod_{i=1}^{N} (V_i)^{u_i} \Big)$. Extending this monomial for $L$ terms and random basis we get $\Big( \prod_{i=1}^{N} (V_i)^{u_i} \Big)^L$. The proofs are available in [26]. In SRFG, the outputs and input mapping correspond to a function $g : 0, 1, \ldots, n \rightarrow F_2$ such that $\forall x \in F_2^N, \ f(x) = g(w(x))$. Following this, we reconstruct Equation 4 in the form of Equation 7.

$$\begin{aligned} f(V_1, V_2, ..., V_N) = \\ \otimes \lambda_f(j) \otimes \Big( \prod_{i=1}^{N} rand(V_i)^{u_i} \Big)^L = \otimes \lambda_f(j) X_{j,N}, \end{aligned} \tag{7}$$

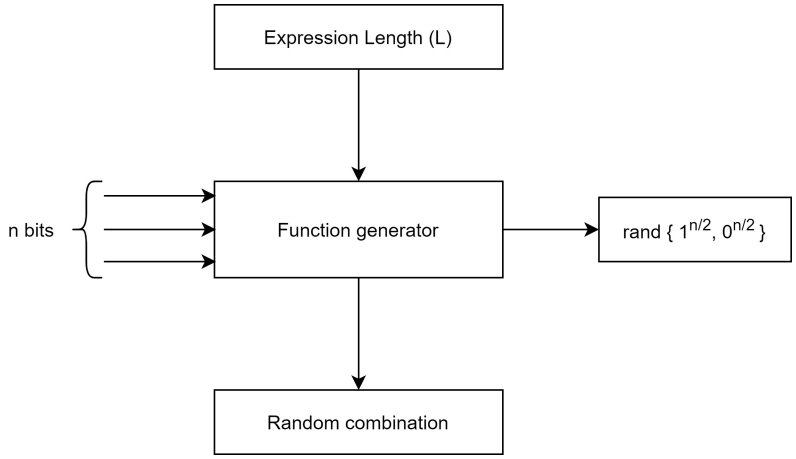


Fig. 2. Logical block structure of SRFG

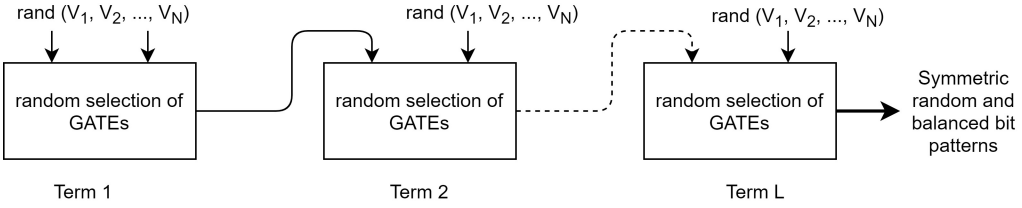


Fig. 3. Chaining of randomization

where $\lambda_f(j), u \in F_2^N$ and $L \in \mathbb{Z},\ j = 1, .., N$ . $X_{j,N}$ is the elementary polynomial of degree $j$ with $N$ variables. $\lambda(f) = \lambda_f(0), \lambda_f(1), ..., \lambda_f(N)$ is called as simplified vector.

For the ease of the understandability of the readers, we have included Table I to summarize the notions used in explaining the technical details of the integration between SRFG and MPC.

TABLE I
SUMMARY OF NOTIONS

| Notion | Description |
|---|---|
| $f()$ | SRFG Function |
| $i$ | Number of GATEs in SRFG function |
| $L$ | Number of terms in SRFG function |
| $V$ | Bit vector |
| $rand$ | Random function |
| $\lambda$ | Random vector output |
| $\prod$ | Selection function |
| $GF$ | Isomorphic field |
| $s$ | A secret |
| $b$ | Size of the secret |
| $\alpha s$ | Encoded secret |
| $n$ | Number of participants |
| $n - variate$ | Polynomial size |
| $FL(.)$ | Linear component |
| $P_j$ | Participants |
| $c$ | Scalar unit belongs to the field |
| $x_i$ | Secret input |

## IV. SRFG ANALYSIS

In this section, we analyze the SRFG properties to check the candidacy of being a primitive in secret sharing use. We divide our discussion into two subsections. Section IV-A shows the applicability properties of SRFG for secret sharing and in Section IV-B, we use the notified properties for MPC.

### A. Secret sharing with SRFG

For the applicability of SRFG, we analyze the properties with random Galois field $GF(2^n)$ [27]. We can reuse the polynomial form of ANF in SRFG as shown in Equation 8.

$$\otimes\ \lambda_u \left( \prod_{i=1}^{N} rand(V_i)^{u_i} \right)^L = rand(V_1)^{u_1} \otimes rand(V_2)^{u_2} \otimes ... \otimes rand(V_N)^{u_N} \tag{8}$$

We consider the secret of sharing to be of $b$ bits. Our goal is to create $n$ parts or shares and require at least $m$ shares to reconstruct. We generate the keys and other initial states such as Initialization Vector (IV) randomly and collectively call them as key $K$. We use traditional Shamir method [28] to split $K$ into $n$ secret shares as: $s_0, s_1, s_2, ..., s_{n-1}$. We use an isomorphic field of $GF(2^b)$, and each $s_i$ is equal to the size of $K$.

We follow the TUS method of secret encryption with a random number for secure secret sharing objectives [29]. Instead of the simple product-sum operation of the shares of random numbers, we use extended product-sum operation of the polynomial terms: $\sum \otimes\ \lambda_u \left( \prod_{i=1}^{N} rand(V_i)^{u_i} \right)^L$. This allows multiple computations to be performed at once instead of only one computation of $\otimes\ \lambda_u \left( \prod_{i=1}^{N} rand(V_i)^{u_i} \right)$ each time. We input the secret $S$ and $timestamp$ to SRFG and an

output random bitstream of size $b$. This $b$ bits output generates the $m-1$ degree random polynomial that follows Equation 9.

$$q(x) = a_0 + a_1 x + ... + a_{m-1}x^{m-1}. \tag{9}$$

In Equation 9, $a_0$ is $S$ to be shared and calculates: $s_1 = q(1),\ s_2 = q(2),\ ...,\ s_n = q(n)$ for $n$ shares. The coefficients $a_0, a_1, ..., a_{m-1}$ in $q(x)$ are random based on SRFG outputs and follows a uniform distribution over the integers in $[0, p]$, and the values $s_1, s_2, ..., s_n$ are computed modulo $p$, where $p$ is a prime number and $S < p$ and $n < p$. We divide the overall process of secret sharing in two phases: distribution and reconstruction. However, we shift some parts of the computation, where the values do not have any dependency on other values, to the pre-processing stage of the computation. Thus, the values can be generated in advance before distribution. This effectively reduces the cost of communication in the online phase of distribution and speeds up the entire process of sharing the secret. In pre-processing, the participants generate the secrets and select the random polynomials through SRFG. In the online phase, the participants distribute the secrets and reconstruct the secrets.

*a) Pre-processing:* As we set the goal of sharing a $b$-bit secret in $n$ shares, each share size is $b/n$ bits. We denote the shares as: $s_i$, where $i = 1, 2, ..., n$. If $b \mod n \neq 0$, we use least significant bits padding. In addition, all the random numbers used in the polynomial are in uniform distribution and do not include the value 0. Moreover, all other values belong to $GF(2^n)$.

*b) Distribution:* Let $s$ be a secret for sharing among $n$ participants and $\overline{s}$ is a share of $s$ for a participant $P_i$. The dealer generates $n$ random numbers $\alpha_0, \alpha_1, ..., \alpha_{n-1}$ and computes $\alpha$ as in Equation 10.

$$\alpha = \prod_{j=0}^{n-1} \alpha_j. \tag{10}$$

The dealer computes $\alpha s$ as a encoded secret and distributes $\alpha s, \alpha_0, ..., \alpha_{n-1}$ to $n$ players by using $(k, n)$ secret sharing schemes. The computation of the dealer's side uses the following equations (Equation 11 to Equation 13) for sharing the secrets.

$$\alpha s = ([\overline{\alpha s}]_0, ..., [\overline{\alpha s}]_{n-1}). \tag{11}$$

$$\alpha_0 = ([\overline{\alpha_0}]_0, ..., [\overline{\alpha_0}]_{n-1}). \tag{12}$$

$$\alpha_{n-1} = ([\overline{\alpha_{n-1}}]_0, ..., [\overline{\alpha_{n-1}}]_{n-1}). \tag{13}$$

*c) Reconstruction:* A user restores a secret $\alpha$ by collecting $\alpha_j$ shares of secret from $j$ parties. The user then reconstructs the secret using the formula shown in Equation 14 to Equation 18.

$$REC([\overline{\alpha s}]_0, ..., \overline{[\alpha s]}_{k-1}) = \alpha s. \tag{14}$$

$$REC([\overline{\alpha_0}]_0, ..., \overline{[\alpha_0]}_{k-1}) = \alpha_0. \tag{15}$$

From the above, we obtain the following.

$$REC([\overline{\alpha_{k-1}}]_0, ..., \overline{[\alpha_{k-1}]}_{k-1}) = \alpha_{k-1}. \tag{16}$$

$$\alpha = \prod_{j=0}^{k-1} \alpha_j. \tag{17}$$

$$\alpha s \times \alpha^{-1} = s. \tag{18}$$

### *B. MPC with SRFG*

We study an MPC problem in an unconditionally secure environment as mentioned in [30]. An *unconditionally secure environment* considers two assumptions: i) in the presence of a passive adversary, no set of size $t < n/2$ of participants learns any additional information other than the information derivable from their own private inputs and the output of the protocol, and ii) in the presence of an active adversary, no set of size $t < n/3$ of participants can learn any additional information or disrupt the protocol. Though there are some developments available for improving the bound of $t < n/3$, the tight bound obtained from the literature is $t < n/2$.

Let $F : (GF(p)^*)^n \rightarrow GF(p)$ denote a $n$-variate polynomial over $GF(p)$ (with inputs restricted to $GF(p)^*$) having $\ell$ non-linear monomials, a form of polynomial with a single term. We represent the function $F$ as $F(x_1, ..., x_n) = F_L(\cdot) + F_{C1}(\cdot) + + F_{C\ell}(\cdot)$, where $FL(\cdot)$ denotes the linear component, and $F_{Ci}(\cdot)(i = 1, ..., \ell)$ denotes monomials. The previously mentioned secret sharing scheme provides the outputs of shares as $s_1, s_2, ..., s_N$ of $N$ participants using the additive-$(n, n)$ threshold scheme. For the generality of the solution, we can assume a multiplicative scheme too as per the application requirements. Let $s_{i,j}$ and $s_{k,j}$ are the shares of a participant $P_j$ from the secret inputs $x_i$ and $x_k$, respectively. We compute every linear function in the following way. To obtain $x_i + x_k$ , each participant $P_j$ computes $s_{i+kj} = s_{i,j} + s_{k,j}$ as an additive $(n, n)$-threshold scheme is $(+, +)$-homomorphic (a homomorphism is a structure-preserving map between two algebraic structures of the same type of groups). For every known scalar $c \in GF(p)$ and each secret input $x_i$, computation of $c \times x_i$ requires that each participant $P_j, j = 1, ..., n)$ calculates $c \times s_{i,j}$ as his share of $c \times x_i$. We can follow two ways for the computation of $c \times x_i$: (i) share the value $c$ amongst all the participants, using the additive $(n, n)$-threshold scheme. Then each participant $P_j$ computes $c_j + s_{i,j}$ as his share of $c + x_i$ , where $c_j$ is the share of $P_j$ from $c$ and (ii) a designated participant, $P_\ell$, chosen by all participants adds $c$ to his share from $x_i$, i.e., computes $c + s_{i,\ell}$. We also observe that computation of an additive inverse is easy and every participant computes the additive inverse of his own share. Thus, every linear function with $n$ inputs can be computed with no interaction. Similarly, we can also compute $c \times x_i$ if we use a multiplicative $(n, n)$-threshold; the only change is that we share the value of each monomial in this scheme. To compute the function value, without revealing any information about the value of each component, we require the conversion of multiplicative $(n, n)$-threshold sharing associated with each monomial $F_{Ci}(\cdot)(i = 1, ..., \ell)$ to a corresponding additive $(n, n)$-threshold sharing. Each participant $P_i$ receives $n - 1$

values $\alpha_{i,j}m_j$ from the participants. Knowing $\alpha_{i,i}$, $m_i$, and the received information, a participant $P_i$ computes $s_i$ as shown in Equation 19. The correctness of the computation is available in [31].

$$s_i = \prod_{j=1}^{n} \alpha_{i,j} m_j. \tag{19}$$

Our assumption restricts the private inputs of all participants to be non-zero. This implies that the value of all monomials in the function $F$ (when evaluated at the private inputs) is in $GF(p)*$, as required. Although this restriction prevents our protocol from being applied non-trivially over $GF(2)$, it is still possible to use our protocol to compute arbitrary functions over $GF(2)$ by encoding them as polynomials over a larger field such as $GF(5)$. Each participant $P_i$ distributes private input $x_i \in GF(p)^*$ amongst all participants, using the additive and multiplicative $(n, n)$-threshold schemes. In order to compute the function $F(x_1, x_2, ..., x_n) = FL(\cdot)+F_{C1}(\cdot)+ ..., F_{C\ell}(\cdot)$, each participant $P_i$ privately computes $F_L(\cdot)$ and all monomials $F_{Cj}(\cdot), j = 1, 2, ..., \ell$.

All the participants collectively have the values of each component of the function. We share the linear component in additive $(n, n)$-threshold form, while each shared monomial is in multiplicative $(n, n)$-threshold form. Let $A_{i,j}$ be the share of participant $P_i$ associated with monomial $F_{Cj}(\cdot)$, in an additive $(n, n)$-threshold format. The participant $P_i$ computes $Y_i = A_{i,0} + A_{i,1} + ..., A_{i,\ell}$, where $Ai, 0$ is the share of $P_i$ associated with the linear component $FL(\cdot)$. Since each participant has a share of the function value associated with an additive $(n, n)$-threshold scheme, they can pool their shares and compute the function value $f_out$, using Equation 20.

$$f_{out} = \sum_{i=1}^{n} Y_i (mod\ p). \tag{20}$$

*a) **Randomness in MPC**:* Reconsidering Equation 5, the Algebraic Normal Form (ANF) of the polynomial representation of SRFG has a degree at most 1 in each input variable following that any monomial of this polynomial is the product of some input variables. In our MPC model with randomness shown above, these linearly connected monomials are the constructions of MPC. As in [25], for any $u \in {F_2}^N, {V_i}^{u_i}$ defines monomial as: $\prod_{i=1}^{N}(V_i)^{u_i}$. Solving these linear monomials implies the fact of SRFG applicability in MPC. Random functions in cryptography ensure unconditional security against adversaries with unlimited resources by introducing unpredictability and complexity. In a key generation, high-entropy random numbers enhance security by generating unpredictable cryptographic keys, making brute-force attacks computationally infeasible. Cryptographic hash functions utilize randomness to produce seemingly random outputs, resisting predictability and collision attacks. Randomized encryption employs Initialization Vectors (IVs) to add variability, preventing patterns in the ciphertext. In Secure MPC, randomization techniques like secret sharing leverage random values to protect sensitive information. The random oracle model, a theoretical construct, aids in analyzing cryptographic schemes' security under certain conditions. Quantum random number generators exploit quantum indeterminacy for true randomness, offering an additional layer of security. Post-quantum cryptography explores randomness in lattice-based schemes, where the difficulty of finding short vectors in random lattices contributes to security against quantum attacks. Overall, the strategic use of randomness across cryptographic applications forms a robust defense against sophisticated adversaries.

*b) **Privacy norms and MPC, SRFG coverage**:* In MPC, n/2-private (semi-honest) assumes up to n/2 parties may act maliciously, while n-private (malicious) allows for adversarial behavior by any subset of up to $n$ parties. The decision depends on the application's sensitivity, practicality, and regulatory demands. N-private norms offer higher security but involve more complex cryptography, potentially leading to increased computational overhead. N/2-private protocols strike a balance between security and efficiency, making them suitable for scenarios where malicious behavior is deemed less likely or where stringent security measures may be impractical. The choice depends on a nuanced evaluation of specific application requirements and threat landscapes. A SRFG can be employed in MPC as a cryptographic primitive when it possesses certain properties essential for secure computation. Through the work of SRFG, we have seen that SRFG possesses secure cryptographic properties as it shows randomness. In MPC, parties jointly compute a function on their private inputs while revealing only the necessary results. The SRFG exhibits pseudorandomness and proceeds toward true randomness behavior with the efficient use of tuning parameters, ensuring that the generated values are computationally indistinguishable from each other. Additionally, the generator is resistant to key recovery attacks, as each party must independently contribute to the computation without revealing sensitive information. Properly incorporating the SRFG within the MPC protocol, often through techniques like secret sharing and cryptographic commitments, guarantees that joint computations maintain privacy and security. The selection of SRFG is crucial for achieving confidentiality and integrity across distributed computations in MPC scenarios.

## V. Security proof

The objective of applying SRFG in secret sharing and MPC is to build a privacy-ensured MPC framework. We also show that randomness in sharing secrets leads to MPC with privacy assurance. In this section, we first define the adversary model and analyze the application of SRFG based on two aspects: information-theoretic security and privacy analysis. Finally, we also provide a discussion on the trust assumptions for the operational context of the proposed MPC framework.

### A. Adversary model and trust assumptions

Let us consider a specific case with a focus on a passive adversary model in the context of secure MPC. The goal is to mathematically describe the capabilities of an adversary attempting to breach privacy in an MPC protocol. The adversary uses the inputs of the number of parties in the distributed network for computation, say $n$. The attacker also attempts to observe the inputs and outputs of the parties, say $X_i$ and $Y_i$, respectively. The attacker is unaware of the $f()$ used for secure computation of secret shares and reconstruction. We

assume that the adversary is passive and can only eavesdrop on the communication channels between parties during the MPC protocol execution. The adversary cannot actively tamper with the messages, inject new messages, or alter the behavior of honest parties. The adversary aims to infer additional information about the private inputs of the parties by analyzing the exchanged messages. The attacker attempts to calculate the indistinguishability of two adjacent inputs for a party with a probability of $Pr[f(X_i) = Y_i] \approx Pr[f(X_i') = Y_i]$.

Adversary models outline potential attacks, while trust assumptions denote parties' reliability. Therefore, it is also necessary to include the trust assumptions under which the adversary model operates and our proposed approach provides robustness. In the context of MPC, trust assumptions regarding the use of SRFG are crucial for the security of cryptographic protocols. Let $G$ be a SRFG $G(.)$ its core algorithm. The trust assumptions typically include the following parameters to be applicable in MPC.

- Unpredictability: The SRFG produces unpredictable and statistically indistinguishable random values. We can formalize it for the SRFG as: $\forall i, j \in$ Output space of $G, Pr[G(.) = i] \approx Pr[G(.) = j]$.
- Seed Unpredictability: The initial seed used to initialize the SRFG is secret to prevent retroactive predictions. Formally, we write this as: $\forall s_1, s_2, Pr[G(s_1) = .] \approx Pr[G(s_2) = .]$.
- Independent outputs: The outputs of the SRFG are independent, ensuring that knowledge of one output does not reveal information about others. We can write this as in Equation 21.

$$Pr[G(.) = i_1, G(.) = i_2, ..., G(.) = i_k] = \prod_{j=1}^{k} Pr[G(.) = i_j]. \tag{21}$$

### B. Information-theoretic analysis of SRFG application

In our information-theoretic analysis of SRFG, we consider two aspects: secrecy and privacy separately.

In information theory, we can represent a digital information content by a message; further we can describe a message as a random variable $Z$, where $Z$ is a discrete set, $Z \in 1, ..., M$. An adversary can observe this message as a random vector $R^n$, where $n$ is the number of symbols in the vector taking value in the set $R$. We call $n$ block length implies the sequences in a vector. In the best case, the joint probability distribution $P_{Z R^n}$ of $Z$ and $R^n$ is known, which implicitly assumes that i) the statistics of the source of information are fully controlled; and ii) the statistical models that describe the processes relating the observation $R^n$ to the message $Z$ are fully characterized. The notion of perfect secrecy implies that $Z$ and $R^n$ be statistically independent, which means that $\forall m \in Z, \forall r^n \in R^n$ we say,

$$P_{Z R^n}(m, r^n) = P_Z(m) P_{R^n}(r^n), i.e., I(W, R^n) = 0. \tag{22}$$

In Equation 22, $I(W, R^n)$ is the mutual information, which measures how much one random variable tells us about another, between the messages and the adversary's interpreted observation. In our SRFG application in MPC, message and shared secrets are synonymous in our discussion. We also need to remember that $Z$ cannot be a function of $R^n$, which implies that some randomization is important to assure information secrecy. Thus, our SRFG application in MPC becomes significant. Now, we analyze how the properties of SRFG can provide $I(W, R^n) = 0$ in MPC and make the MPC information-theoretic secure. We consider two random variables $x \in X$ and $y \in Y$, where $|X|, |Y| < \infty$ and have the fixed joint distribution $P_{XY}$. $x$ is a random secret share output from SRFG and $y$ is the adversary's observation and correlated with $x$. Further, we also consider another variable $q \in Q$ that may have some limited information of $x$. We characterize the quantity as in Equation 23.

$$g_\epsilon(X; Y) := max\ I(Y, Q), I(X; Q) \leq \epsilon. \tag{23}$$

In the extreme point of information leakage, $I(X; Q) = \epsilon$ and assuming that there is a deterministic function $f$, we can get the simplified quantity of Equation 23 as shown in Equation 24.

$$\tilde{g_\epsilon}(X; Y) := sup\ H(f(Y)), f : I(f(Y); X) = \epsilon. \tag{24}$$

In case of perfect secrecy, $\epsilon = 0$ and the notion of Equation 23 becomes as Equation 25.

$$g_0(X; Y) := max\ I(Y, Q), I(X; Q) = 0. \tag{25}$$

Assuming the random $b$ bits variable for $X$ and $Y$, and $l$ bits of $Q$, the probability of information leakage among $X, Y$, and $Z$ becomes as shown in Equation 26.

$$P_{Q|Y} := Q \models X. \tag{26}$$

Since $X, Y$, and $Q$ form a Markov chain $X \to Y \to Q$, we say that Equation 27 and Equation 28 hold.

$$P_{Q|Y}(.|0)P_{Y|X}(0|1) + P_{Q|Y}(.|1)P_{Y|X}(1|1) = P_{Q|X}(.|1) \tag{27}$$

and

$$P_{Q|Y}(.|0)P_{Y|X}(0|0) + P_{Q|Y}(.|1)P_{Y|X}(1|0) = P_{Q|X}(.|0) \tag{28}$$

The condition $Q \models X$ implies that $P_{Q|X}(.|1) = P_{Q|X}(.|0) = P_Q(.)$. Therefore, we can rewrite the above equations as in Equation 29:

$$\begin{aligned} &P_{Q|Y}(.|0)P_{Y|X}(0|1) + P_{Q|Y}(.|1)P_{Y|X}(1|1) = \\ &P_{Q|Y}(.|0)P_{Y|X}(0|0) + P_{Q|Y}(.|1)P_{Y|X}(1|0) = P_Q(.). \end{aligned} \tag{29}$$

From the assumption that $X$ and $Y$ are dependent as $X$ is an input to SRFG and $Y$ is an output from SRFG for MPC, the above system of equations has a unique solution that turns out to satisfy Equation 29. This implies that $I(Y; Q) = 0$. Similarly, we can also follow for $I(X; Q) = 0$. Thus, MPC with SRFG as a primitive ensures information theoretic secrecy.

### C. Privacy analysis of SRFG application

We follow the $t - private$ notion for privacy analysis as shown in [32]. We also use the concepts of strong privacy, and weak privacy as mentioned in the work [32]. Let $V_1, V_2, ..., V_N$ are non-zero binary variables used in SRFG and each variable consists of $\{0,1\}^n$, where $n$ is the number of bits in the variable. The interpretation of the variables is synonymous with the private inputs to MPC, which outputs shared secrets; thus we formalize MPC as a system of randomization. We also consider $\epsilon, \delta \geq 0$ satisfying $\epsilon + \delta < \frac{1}{2}$, and $f(V_1, V_2, ..., V_N) \rightarrow \{0,1\}^n$. Following this construction and the respective Theorem 1 and Lemma 5 of [32], we say that $f$ is weakly $\frac{n}{2}$-private. This means that no coalition of size $\leq \frac{n}{2}$ can get any additional information on the private input variables. We also apply Theorem 3 of [32] to check the strong privacy validity for SRFG in MPC. As we know, SRFG uses a chaining process for the randomization, there exist $L$ subsets depending upon the chaining length or the required terms of chains. Any subset of SRFG $f$ can be written as in Equation 30:

$$\tilde{f}_L(V_1, V_2, ..., V_N) = \bigoplus V_j, j \in L, \tag{30}$$

where $\bigoplus$ can be any boolean function. The probability of any $V_i = V_j$ is minimum as $\frac{1}{n^n.n^n} \rightarrow 0$. Thus, for $n$-bit private inputs SRFG benefits to provide $n$-privacy.

We also analyze the information-theoretic privacy for our SRFG application in MPC. Information-theoretic privacy is another aspect to ensure the privacy of a system. As MPC deals with private inputs, the analysis of information-theoretic privacy becomes important. The main goal of the presented secret sharing and MPC in this paper is to use SRFG as a primitive in the mentioned processes. Therefore, we are more interested to analyze the information-theoretic privacy for SRFG. Let $S$ be a set of secret shares in our presented MPC and $J$ be the set of interpreted or guessed versions of the secrets by an adversary. We can measure the leakage of privacy as shown with a conditional probability formation in Equation 31 [33].

$$\mathcal{L}(S \rightarrow J) \triangleq \frac{P(S|J)}{P(S)}, \tag{31}$$

where $P(S|J) := \sum_{j \in J} P_J(j) max_{s \in S} P_{S|J}(s|j) = \sum_{j \in J} max_{s \in S} P_S(s) P_{J|S}(j|s)$. The value of this conditional probability of leakage becomes 0 as per norms of SRFG [25]. Thus, SRFG ensures the information-theoretic privacy norms.

## VI. Experiments and Results

In this section, we first describe the methods followed to execute the integration of SRFG in MPC in Section VI-A. It also includes the definition of performance metrics used for experiments and evaluation. We discuss the observed results in Section VI-B. Further, we analyze the impact of MPC and SRFG integration on the messaging systems in Section VI-C and provide a comparative analysis with the existing frameworks in Section VI-D.

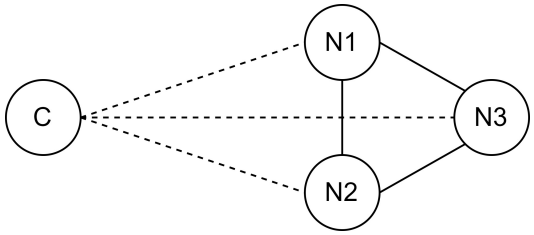


Fig. 4. Logical network model of the proposed MPC

### A. Experimental methodology

We use a three-node network model for the implementation, where the three nodes are three participants under a controller node. The function of a controller node is to keep a watch on the trust and privacy factors of the MPC, whereas individual nodes (N1, N2, and N3) compute secrets based on the use of SRFG. The nodes have standard computing hardware with sufficient processing power and memory, Network interface cards (NICs) for communication, and storage for data and program files. The controller also confirms the task of combining the secret shares. We show a logical network model in Figure 4. Each node is configured with 16GB RAM, 4TB of hard disk space, and 4.6GHZ CPU speed. The controller $C$ is a general computing system only doing synchronization tasks and also has the specifications of other nodes. We use Linux OS and server for storage. For the network, we use Campus Area Network (CAN) with a transmission speed of 100 Mbps. The steps of implementing MPC in a distributed network model are discussed below.

- Step 1: Initialize node configuration with pre-defined secure keys for computation purposes. We use *PyCryptodome*, previously PyCrypto, and *pyOpenSSL*. *PyCryptodome* provides a wide range of cryptographic functions such as encryption, decryption, hashing, and key generation; it offers a high-level API for easy integration into Python applications. *pyOpenSSL* provides a Python wrapper around the OpenSSL library for SSL/TLS functionality; it is useful for secure communication over networks using SSL/TLS protocols and supports certificate management, encryption, decryption, and secure connections.
- Step 2: Ensure secure channels with VPN installed among the three nodes and the controller. We use *openvpn-api* for interacting with *OpenVPN*. These libraries enable Python scripts to initiate and manage VPN connections for secure node-to-node communication.
- Step 3: Initialize a secret pool. We use Python libraries like *secrets* for generating random secrets, *pyOpenSSL* for encryption, and *sqlite3* for storage.
- Step 4: Select a random secret and send it to the nodes from the controller.
- Step 5: All the nodes and the controller are pre-tested with the SRFG function call. SRFG function call is implemented in Python and included in the overall execution using higher-level libraries such as *Socket.IO*.
- Step 6: Split input of secret share and use SRFG function call. We use Python's *MPyC*.

- Step 7: Integrate consistency checks and cryptographic verifications to ensure that messages are not tampered with during transmission and that parties adhere to the protocol.
- Step 8: Execute joint computations.
- Step 9: Reconstruct the final output by aggregating results from different parties securely.
- Step 10: Conclude the protocol securely, accounting for potential network failures or disruptions. Implement mechanisms for graceful termination and cleanup.
- Step 11: Review privacy attainment.

Based on the above methodology, we implement our MPC with SRFG and measure the performance based on the following metrics.

- Latency: We measure the latency by calculating the delay of the message transmission. In this case, we consider the time starting from initializing the secret till the reconstruction of the secret.
- Throughput: Throughput is measured by the number of secrets processed successfully by the proposed system in a single unit of time. We observe the results in minutes.
- Privacy attainment: The number of attempts of privacy breaches on the secrets or the components of the secrets and the resiliency of the proposed system to avoid the breaches. The ratio provides us with a clear indication of the privacy measures of the system.

To compare the proposed MPC using SRFG with other random number generators, we use the Permuted Congruential Generator (PCG) and Xorshift [34]. The criteria to select these two random number generators for comparison is the quality of randomness is statistically proven and both are quite fast in producing randomness.

### B. Obtained results

We execute overall 500 secret-sharing trials with the three nodes and for each trial, we try to execute adversarial methods to obtain a secret component. We compare all the trials for all the random number generators. The comparison of the latency for all the methods is shown in Figure 5. The results show that PCG has more latency than the SRFG, whereas Xorshift is the fastest randomness generator in the comparison and also has less latency than the SRFG. Statistically, the average latency for PCG is $32.4$ seconds, SRFG is $22.5$ seconds, and Xorshift is $20.7$ seconds.

In the next experiment, we measure the throughput of the systems and obtain the results shown in Figure 6. We observe the fact that our proposed MPC framework with SRFG randomness is good in throughput; it is approximately $30\%$ better in throughput as compared to PCG and Xorshift combined. Another interesting fact to notice between Figure 5 and Figure 6 is that despite showing more latency for SRFG-based MPC, the throughput is better for this combination. This is because of the internal construction of the random number generators.

In another experiment, we measure the privacy to ensure the resiliency of the methods. We convert the privacy attainment ratio into percentages and obtain a suitable graph as shown in Figure 7. From this figure, we observe that privacy attainment performance is optimum for our proposed SRFG-based MPC with $100\%$, whereas the other two methods show degrading performance with the increased number of shares. This signi-

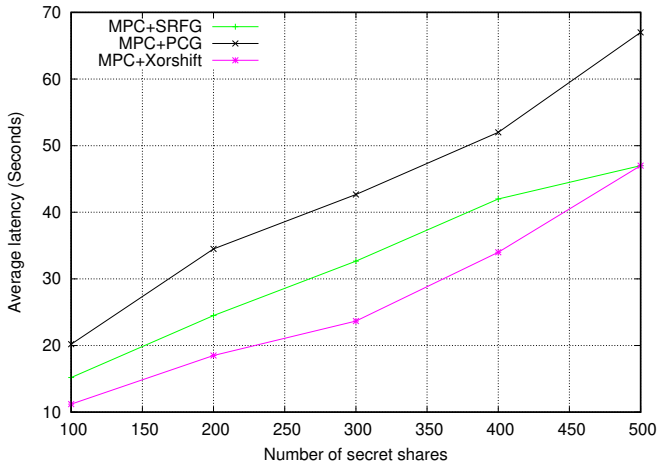


Fig. 5. Comparison of latency

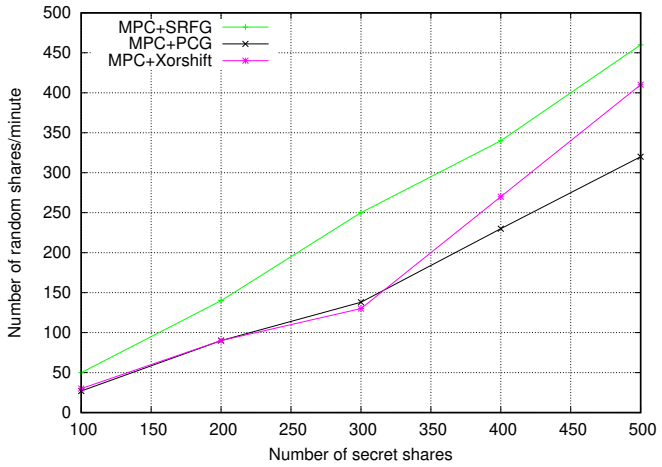


Fig. 6. Comparison of throughput

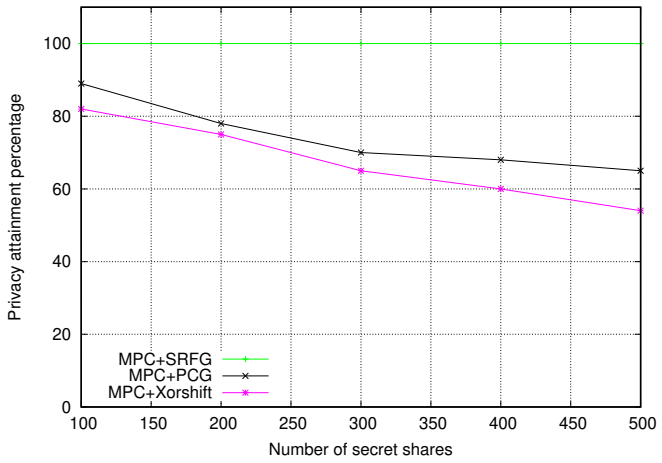


Fig. 7. Comparison of privacy attainment

TABLE II
COMPARATIVE ANALYSIS OF THE STATE-OF-THE-ART MODELS

| Ref. | Randomness | Communication overhead | Computational overhead | Average latency (seconds) | Average throughput | Privacy attainment |
|---|---|---|---|---|---|---|
| L. Harn et al. [8] | No | $O(nmlogn)$ | $O(nmlogn)$ | 40.2 | 46 | 78% |
| R. Xu et al. [11] | No | $O(n)O(lognm)$ | $O(nmlogn)$ | 30.5 | 43 | 80% |
| J. Ding et al. [12] | No | $O(nmlogn)$ | $O(nmlogn)$ | 19.5 | 56 | 65% |
| J. Yang et al. [15] | No | $O(n^2logm)$ | $O(nlogn)$ | 21.7 | 67 | 89% |
| Z. Li et al. [23] | No | $O(nlogm)$ | $O(nlogn)$ | 24.7 | 32 | 92% |
| Proposed SRFG-based MPC | Yes | $O(logn)$ | $O(logn)$ | 22.5 | 70 | 100% |

fies that our proposed method of SRFG-based MPC is resilient and ensures privacy, which is the main objective of developing the proposed method.

### C. Impact on messaging system compliances

In this part, we analyze the impact of SRFG on a general distributed messaging system that allows asynchronous connections, fault tolerance, and scalability.

- Asynchronous connections and scalability: Our proposed MPC framework with SRFG construction for random secret shares ensures asynchronous connections, as we have experimented with the work in a distributed asynchronous messaging system architecture. With the increasing number of shares or secrets, the proposed system does not show any significant overhead; however, its latency is higher. We keep the scope open to optimize the latency in our future works.
- Fault tolerance: In terms of fault tolerance, our proposed system shows high tolerance as we can see from Figure 7 the system is stable as its privacy attainment percentage is 100 percent.

Apart from the above discussion for the messaging system compliance, we also measure the complexity of our system in terms of communication cost and computational cost. For computation cost, the average cost for secret share and secret reconstruction stands for $O(nlogn)$, where $n$ is the number of secret shares; the communication cost becomes $O(logn)$. The use of oblivious transfers and shared random values reduces the need for direct communication between all parties, contributing to the $O(logn)$ communication complexity.

### D. Comparative analysis

We compare our proposed work with some of the significant state-of-the-art models such as the work of L. Harn et al. [8], R. Xu et al. [11], J. Ding et al. [12], J. Yang et al. [15], and Z. Li et al. [23]. We show the comparative analysis in Table II. The table shows that our proposed MPC framework is better than the other existing frameworks in terms of complexity, randomness of shares, throughput, and privacy attainment; however, there is scope for improvement in the latency.

## VII. CONCLUSION

In our presented work, we show the analysis of random function applicability in secret sharing and MPC. We show the use of a symmetric random function generator in secret sharing and MPC. Our application is the first one in the direction of MPC to use a random function generator for secret sharing. The analysis of the properties of the used random function generator shows that our application of MPC can achieve a negligible probability of information leakage; thus, it confirms provides information-theoretic security and privacy. Besides, for the first time in the domain of randomness, we show that randomness in MPC can obtain optimum privacy with $n$-private notions for a Boolean function-based random function generator for binary dependent variables, such as the secret shares. This observation also implies that the properties of a symmetric random function generator are well-suited for information security and privacy. The presented work lacks the efficient computation of the decryption model ensuring trust and privacy. Besides, the presented work faces the limitation of communication overhead arising from the random secret exchanges. In the future, we would like to design a fair decryption model with the same notions of privacy and use the random function generator to be applicable in a trustworthy and uncoordinated environment.

## REFERENCES

[1] Al-Otaibi, Y.D., Distributed multi-party security computation framework for heterogeneous internet of things (IoT) devices. Soft Computing, Volume 25, 2021 Pages 12131–12144.
[2] D. W. Archer, D. Bogdanov, L. Kamm, Y. Lindell, K. Nielsen, J. I. Pagter, N. P. Smart, R. N. Wright, From Keys to Databases – Real-World Applications of Secure Multi-Party Computation, available at: https://eprint.iacr.org/2018/450.pdf, accessed on: May 5, 2022.
[3] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C. Z. Gao, H. Li, Y. Tan, Secure Multi-Party Computation: Theory, practice and applications, Information Sciences, Volume 476, 2019,Pages 357-372.
[4] Y. Wu, X. Wang, W. Susilo, G. Yang, Z. L. Jiang, S. M. Yiu, H. Wang, Generic server-aided secure multi-party computation in cloud computing, Computer Standards & Interfaces, Volume 79, 2022.
[5] W. Diffie, M. Hellman, New directions in cryptography, IEEE Transactions on Information Theory, Volume 22, No. 6, 1976, Pages 644-654.
[6] B. Sanders, Quantum Cryptography for Information-Theoretic Security. In: Vaseashta, A., Braman, E., Susmann, P. (eds) Technological Innovations in Sensing and Detection of Chemical, Biological, Radiological, Nuclear Threats and Ecological Terrorism. NATO Science for Peace and Security Series A: Chemistry and Biology, Springer, 2012.
[7] Z. Wang, S. S. Cheung, Y. Luo, Information-Theoretic Secure Multi-Party Computation With Collusion Deterrence, IEEE Transactions on Information Forensics and Security, Volume 12, No. 4, 2017, Pages 980-995.
[8] L. Harn, Z. Xia, C. Hsu, Y. Liu, Secret sharing with secure secret reconstruction, Information Sciences, Volume 519, 2020, Pages 1-8.
[9] R. Cramer, I. B. Damgrd, J. B. Nielsen, Secure Multiparty Computation and Secret Sharing (1st. ed.), Cambridge University Press, USA,. 2015.
[10] F. Miao, Y. Fan, X. Wang, Y. Xiong, M. Badawy, A (t, m, n)-Group Oriented Secret Sharing Scheme, Chinese J. Electron., Volume 25, 2016, Pages 174-178.

[11] R. Xu, Xu Wang, K. Morozov, C. Cheng, J. Ding, Revisiting group oriented secret sharing schemes, Information Sciences, Volume 589, 2022, Pages 751-769.

[12] J. Ding, P. Ke, C. Lin, H. Wang, Bivariate polynomial-based secret sharing schemes with secure secret reconstruction, Information Sciences, Volume 593, 2022, Pages 398-414.

[13] R. Kaboli, S. Khazaei, M. Parviz, On group-characterizability of homomorphic secret sharing schemes, Theoretical Computer Science, Volume 891, 2021, Pages 116-130.

[14] A. Chandramouli, A. Choudhury, A. Patra, A Survey on Perfectly-Secure Verifiable Secret-Sharing, ACM Computing Survey, Accepted, 2022.

[15] J. Yang, F. W. Fu, New (k,l,m)-verifiable multi-secret sharing schemes based on XTR public key system, Theoretical Computer Science, Volume 910, 2022, Pages 54-67.

[16] A. Kanso, M. Ghebleh, A trapdoor one-way function for verifiable secret sharing, High-Confidence Computing, Volume 2, Issue 2, 2022.

[17] E. Hoogerwerf, D. van Tetering, A. Bay, Z. Erkin, Efficient Joint Random Number Generation for Secure Multi-party Computation, In Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021), 2021, Pages 436-443.

[18] L. F. de Souza, A. Tonkikh, S. T. Piergiovanni, R. Sirdey, O. Stan, N. Quero, P. Kuznetsov, RandSolomon: Optimally Resilient Random Number Generator with Deterministic Termination, OPODIS 2021, Volume 23, Pages 1-16.

[19] E. Vedadi, Y. Keshtkarjahromi, H. Seferoglu, Efficient Coded Multi-Party Computation at Edge Networks, IEEE Transactions on Information Forensics and Security, Volume 19, 2024, Pages 807-820.

[20] K. Iwamura, A. A. A. M. Kamal, "Communication-Efficient Secure Computation of Encrypted Inputs Using (k, n) Threshold Secret Sharing," in IEEE Access, Volume 11, 2023, Pages 51166-51184.

[21] B. Yin, H. Zhang, J. Lin, F. Kong, L. Yu, PVFL: Verifiable federated learning and prediction with privacy-preserving, Computers & Security, Volume 139, 2024, Pages 1-11.

[22] R. B. Christensen, P. Popovski, Private Randomness Agreement and its Application in Quantum Key Distribution Networks, IEEE Communications Letters, Volume 27, No. 2, 2023, Pages 477-481.

[34] K. Bhattacharjee, S. Das, A search for good pseudo-random number generators: Survey and empirical studies, Computer Science Review, Volume 45, 2022, Pages 1-27.

[23] Z. Li, M. Wang, T. G. Tan, J. Zhou, Funder: Future-Proof Unbiased Decentralized Randomness, IEEE Internet of Things Journal, Volume 11, No. 6, 2024, Pages 10352-10367.

[24] R. Saha, G. Geetha, Symmetric random function generator (SRFG): A novel cryptographic primitive for designing fast and robust algorithms, Chaos, Solitons & Fractals, Volume 104, 2017, Pages 371-377.

[25] R. Saha, G. Geetha, G. Kumar, W. J. Buchanan, T. H. Kim, A Secure Random Number Generator with Immunity and Propagation Characteristics for Cryptography Functions. Applied Sciences, Volume 11, 2021, Page 1-12.

[26] A. Canteaut, Lecture Notes on Cryptographic Boolean Functions, 2016, available at: https://www.rocq.inria.fr/secret/Anne.Canteaut/, accessed on: 01 May 2022.

[27] A. Lubotzky, L. Rosenzweig, The Galois group of random elements of linear groups, American Journal of Mathematics, October 2014, Vol. 136, No. 5, 2014, Pages 1347-1383.

[28] A. Shamir, How to share a secret, Communications of the ACM, Volume 22, no.11, 1979, Pages 612–613.

[29] T. Shingu, K. Iwaumura, K. Kaneda, Secrecy Computation without Changing Polynomial Degree in Shamir's (K, N) Secret Sharing Scheme, In Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016), Volume 1, DCNET, 2016, Pages 89-94.

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

[31] H. Ghodosi, J. Pieprzyk, R. Steinfeld, Multi-party computation with conversion of secret sharing. Designs, Codes and Cryptography, Volume 62, 2012, Pages 259–272.

[32] B. Chor, E. Kushilevitz, A zero-one law for Boolean privacy. In Proceedings of the twenty-first annual ACM symposium on Theory of computing (STOC '89), Association for Computing Machinery, New York, NY, USA, 1989, Pages 62–72.

[33] M. Bloch, O. Günlü, A. Yener, F. Oggier, H. V. Poor, L. Sankar, R. F. Schaefer, An Overview of Information-Theoretic Security and Privacy: Metrics, Limits and Applications, IEEE Journal on Selected Areas in Information Theory, Volume 2, no. 1, 2021, Pages 5-22.