Application of the Chinese Remainder Theorem in Secret Sharing Schemes for Secure Encapsulation and Computation of Student Lifestyle Tabular Data Descriptive Statistics

Nayla Putri Ghaisani - 13525140
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: naylaghaisani9@gmail.com , 13525140@std.stei.itb.ac.id

Abstract*—Data privacy in educational data mining has become a critical challenge, especially when handling sensitive student attributes. This paper explores a Secure Multi-Party Computation (SMPC) framework designed for tabular datasets, specifically the Student Lifestyle Dataset. We propose a threshold secret sharing scheme utilizing the Chinese Remainder Theorem (CRT), based on the Asmuth–Bloom framework, to achieve secure encapsulation of numerical attributes such as Study Hours Per Day and GPA. Furthermore, we develop a communication-free, homomorphic processing method to compute descriptive statistics (mean and aggregate sums) directly over the encrypted secret shares without revealing the underlying individual student records. The theoretical analysis proves that the scheme guarantees information-theoretic privacy below the threshold, eliminates communication overhead during additive aggregation, and provides an efficient alternative to polynomial-based cryptography.

***Keywords*—Chinese Remainder Theorem, Secret Sharing, Asmuth–Bloom Scheme, Secure Multi-Party Computation, Tabular Data, Descriptive Statistics.**

I. Introduction

In distributed data infrastructures, encryption serves as a fundamental mechanism to protect data assets. As established by Waters, Public-Key encryption is a powerful mechanism for protecting the confidentiality of stored and transmitted information [1]. Traditionally, however, encryption is viewed as a method for a user to share data to a targeted user or device [1]. This conventional paradigm inherently restricts operational capabilities, as encrypted databases typically require full decryption before descriptive statistics can be computed, thereby exposing sensitive granular records to central server nodes.

To optimize data utility without sacrificing underlying confidentiality, modern distributed processing frameworks have become essential. As noted by Ma et al., with digital assets increasingly comprising a significant portion of personal wealth, the secure management and transfer of digital legacies have emerged as a pressing concern, wherein secret sharing offers a solution to this problem [2]. Nonetheless, unoptimized data fragmentation presents severe operational hazards. Across distributed storage architectures, distributing shares containing the unique private key for digital assets poses significant risks of theft or tampering, potentially leading to the illegal appropriation of user assets [2].

Consequently, advanced cryptographic protocols must execute algebraic operations directly over hidden parameters. Ishai and Paskin demonstrated an efficient paradigm where, given a branching program P and an encryption c of an input x, it is possible to efficiently compute a succinct ciphertext c' from which *P(x)* can be efficiently decoded using the secret key [3]. This baseline guides Secure Multi-Party Computation (SMPC) frameworks motivated by the goal of securely searching and updating distributed data [4]. According to Boyle et al., a robust multi-party model allows one to split each $f \in \mathcal{F}$ into p succinctly described functions $f_i: 0,1^n \to \mathbb{G}, 1 \le i \le p$ such that: (1) $\sum_{i=1}^{p} f_i = f$ and (2) any strict subset of the f_i hides f [4], serving as a specialized method for succinctly performing an 'additive secret sharing' of functions [4]. Furthermore, Applebaum et al. showed that randomizing polynomials allow representing a function $f(x)$ by a low-degree randomized mapping $\hat{f}(x,r)$ whose output distribution on an input z is a randomized encoding of $f(x)$ [5].

This paper analyzes the application of the Chinese Remainder Theorem (CRT) combined with the Asmuth–Bloom threshold secret sharing scheme to securely encapsulate and compute basic descriptive statistics specifically aggregate sums and means for the numerical parameters of the Student Lifestyle Dataset without exposing individual student identities.

II. Theory

A. Modular Arithmetic and Congruence Relations

Modular arithmetic is a system of arithmetic for integers, where numbers "wrap around" upon reaching a certain value known as the modulus. Formally, let $a, b \in \mathbb{Z}$ and $m \in \mathbb{Z}^+ (m > 0)$. The integer a is said to be congruent to b modulo m, denoted as:

$$a \equiv b \pmod{m}$$

if and only if m divides the difference $(a - b)$, which implies that a and b leave the exact same non-negative remainder when divided by m.

In the algebraic structure of ring $\mathbb{Z}_m$, modular arithmetic preserves fundamental homomorphic properties under addition and multiplication. For any arbitrary integers a and b, the following operations hold true:

1. $(a + b) \bmod m = \big((a \bmod m) + (b \bmod m)\big) \bmod m$
2. $(a \cdot b) \bmod m = \big((a \bmod m) \cdot (b \bmod m)\big) \bmod m$

These algebraic properties form the baseline infrastructure for cryptographic protocols, allowing complex equations to be evaluated distributively within finite fields without breaking numerical integrity.

B. Chinese Remainder Theorem (CRT)

The Chinese Remainder Theorem (CRT) is an essential theorem in number theory that establishes the existence and uniqueness of solutions for simultaneous linear congruence equations.

Let $m_1, m_2, \dots, m_k$ be a set of pairwise relatively prime positive integers (pairwise coprime) such that $\gcd(m_i, m_j) = 1$ for any $i \neq j$. For any given set of arbitrary integers $a_1, a_2, \dots, a_k$, the system of simultaneous linear congruences:

$$x \equiv a_i \pmod{m_i}, \quad \text{for } i = 1,2, \dots, k$$

possesses a unique solution x within the complete residue system modulo M, where M is the product of all individual moduli:

$$M = \prod_{i=1}^{k} m_i = m_1 \cdot m_2 \dots m_k$$

The unique solution x within the range $0 \leq x < M$ can be analytically reconstructed using the following explicit formulation:

$$x = \left(\sum_{i=1}^{k} a_i \cdot M_i \cdot Y_i \right) \bmod M$$

where $M_i = M/m_i$ represents the partial product of all moduli excluding the i-th element, and Y_i denotes the unique modular multiplicative inverse of M_i with respect to m_i, satisfying the relation:

$$Y_i \equiv M_i^{-1} \, (mod \; m_i) \Longrightarrow M_i \cdot Y_i \equiv 1 \, (mod \; m_i)$$

C. Asmuth-Bloom Threshold Secret Sharing Scheme

The Asmuth–Bloom scheme is a cryptographic (k, n)-threshold secret sharing protocol fundamentally constructed upon the algebraic properties of the Chinese Remainder Theorem. The primary objective of an encryption threshold scheme is to divide a secret data value S into n distinct fragments called shares, distributed across n independent processing servers, such that any authorized subset of at least k shares can successfully reconstruct the original secret, while any subset of fewer than k shares yields entirely zero information regarding S.

To implement the Asmuth–Bloom framework, a sequence of pairwise coprime integers $m_0, m_1, m_2, \dots, m_n$ must be chosen in a strictly increasing order $(m_0 < m_1 < m_2 < \cdots < m_n)$. To satisfy the required mathematical threshold constraint, these moduli must strictly adhere to the following inequality condition:

$$\prod_{i=1}^{k} m_i > m_0 \cdot \prod_{i=1}^{k-1} m_{n-i+1}$$

The domain of the secret space is bounded within the range $0 \leq S < m_0$. The encapsulation phase is executed through the following sequence:

1. A random integer γ (referred to as the blinding factor) is independently generated to mask the secret, constructing an escalated temporary parameter S':

$$S' = S + \gamma \cdot m_0$$

where γ is carefully bounded such that $0 \leq S' < \prod_{i=1}^{k} m_i$.

2. The sequence of individual shares s_i allocated to each respective server node $i \in \{1,2, \dots, n\}$ is derived by mapping the blinded parameter onto the coprime residue sub-spaces:

$$s_i = S' \bmod m_i$$

During the secure reconstruction phase, when any k servers pool their aggregated local shares together, the Chinese Remainder Theorem is invoked over the chosen k subset of moduli to solve for S'. Once S' is uniquely obtained, the original underlying data secret is cleanly extracted via a straightforward modular reduction:

$$S = S' \bmod m_0$$

III. Data Modelling and Computational Algorithms

A. Data Representation and Class Domain Modeling

The structural backbone of the privacy-preserving analytical framework relies on modeling sensitive features into safe mathematical structures before network transmission. In this system, targeted elements from the *Student Lifestyle Dataset* (specifically x_{study} representing Study_Hours_Per_Day and x_{gpa} representing GPA) are mapped onto an isolated algebraic domain. The selection of these two features is justified by the following strategic rationales:

- Divergent Data Domain Representation: The Study_Hours_Per_Day feature represents continuous numerical activity data within a moderate range (a scale of 1–10), whereas GPA represents a highly sensitive academic performance metric bound by a strict upper limit (a scale of 0.00–4.00). This structural variation across data domains is critical for stress-testing the resilience and robustness of the fixed-point scaling scheme.

- Imperative of Individual Privacy (Privacy-Sensitivity): GPA metrics and daily study patterns constitute highly confidential personal student data. Unsanctioned exposure or leakage of individual GPA records can trigger adverse socio-psychological ramifications. Consequently, these two attributes serve as ideal target candidates to demonstrate the system's capacity to extract macro-level aggregate statistics (such as the population mean) without exposing raw, granular individual records to untrusted third parties or to the distributed computing servers themselves.

The primary data object transformations are defined systematically as follows:

1. Continuous-to-Integer Fixed-Point Scaling Mapping:

Modular arithmetic and linear congruence systems dictate constraints that operate strictly over integer rings ($\mathbb{Z}$). Real-valued parameters representing student habits are discrete-mapped utilizing a uniform scaling coefficient factor $\alpha = 100$:

$$\mathrm{X}_{scaled} = \{x_1, x_2, \dots, x_N\} \subset \mathbb{Z}, \quad \text{where } x_j = \left\lfloor \mathrm{Value}_j \times \alpha + 0.5 \right\rfloor$$

This operational mechanism shifts floating-point measurements into exact whole numbers, effectively mitigating downstream decimal truncation rounding errors while preserving perfect accuracy during homomorphic calculation.

2. Decentralized Residue Matrix Structure:

A clear mathematical matrix of distributed shards separates individual identities from cleartext attributes. For an infrastructure consisting of n independent computing server nodes, the secret elements are fragmented into a multi-dimensional matrix array S:

$$\mathrm{S} = \left[s_{i,j}\right]_{n \times N} = \begin{pmatrix} s_{1,1} & s_{1,2} & \dots & s_{1,N} \\ s_{2,1} & s_{2,2} & \dots & s_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n,1} & s_{n,2} & \dots & s_{n,N} \end{pmatrix}$$

where each column vector represents the isolated residues of a single student profile, distributed horizontally across disjoint network shards.

B. Mathematical Core Functions and Low-Level Operational Algorithms

To achieve high-integrity execution constraints, the mathematical framework operates strictly on internal computational structures without utilizing external mathematical algebraic dynamic libraries. The core execution engine is constructed from three primary primitive modules:

1. Procedure Extended_GCD(a, b):

```python
def Extended_GCD(a, b):
    if a == 0:
        return b, 0, 1
    gcd, x1, y1 = Extended_GCD(b % a, a)
    x = y1 - (b // a) * x1
    y = x1
    return gcd, x, y
```

- Mathematical Intent: Implements the Extended Euclidean Algorithm to systematically evaluate the greatest common divisor (gcd) of two arbitrary integers, while simultaneously calculating the structural Bézout coefficients (x, y).

- Mathematical Statement: Computes a unique pair of integers (x, y) such that:

$$a \cdot x + b \cdot y = \gcd(a, b)$$

- Input Arguments: Integer a, Integer b.
- Return Output: A tuple matrix $(\gcd, x, y)$.

2. Function Mod_Inverse(a, m):

```python
def Mod_Inverse(a, m):
    gcd, x, y = Extended_GCD(a, m)
    return (x % m + m) % m
```

- Mathematical Intent: Derives the unique modular multiplicative inverse of a targeted integer parameter a within the complete residue class modulo m.

- Mathematical Statement: Evaluates the parameter $x \equiv a^{-1} \pmod{m}$, which logically satisfies the multiplicative congruence condition:

$$a \cdot x \equiv 1 \pmod{m}$$

- The algorithm securely handles negative linear results by mapping coefficients back to the positive finite field via the relation $\left((x \bmod m) + m\right) \bmod m$.

- Input Arguments: Integer a, Modulus parameter m.
- Return Output: The normalized positive inverse scalar element.

3. Function CRT(remainders, moduli):

```python
def CRT(remainders, moduli):
    M = 1
    for m_val in moduli:
        M *= m_val
    total = 0

    for r, m_val in zip(remainders, moduli):
        Mi = M // m_val
        total += r * Mi * Mod_Inverse(Mi, m_val)
    return total % M
```

- Mathematical Intent: Reconstructs the unique global solution bounded by a simultaneous system of linear congruence parameters using the Chinese Remainder Theorem.

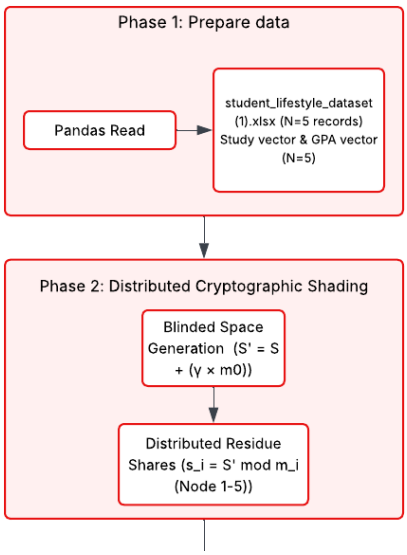
- Mathematical Statement: Given a prime product $M = \prod_{i=1}^{k} m_i$ and partial components $M_i = M/m_i$, the procedure resolves the total underlying value via:

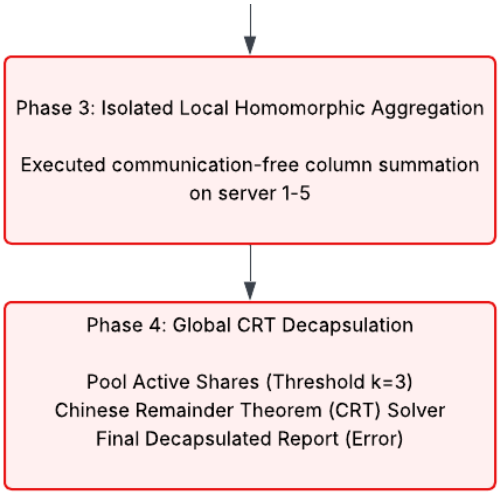
$$X = \left(\sum_{i=1}^{k} r_i \cdot M_i \cdot (M_i^{-1} \bmod m_i) \right) \bmod M$$

- Input Arguments: Array of modular residues (remainders), array of pairwise coprime divisors (moduli).
- Return Output: The uniquely resolved overarching integer parameter X.

C. *Operational Architecture and Pipeline Workflow*

The execution workflow of the Secure Multi-Party Computation (SMPC) protocol routes sequentially through four separate algorithmic phases, keeping **co**mputation tasks isolated until final statistical reconstruction:





IV. Simulation Experimental Results and Discussion

A. *Experimental Setup and Sample Dataset Characteristics*

The simulation infrastructure is instantiated utilizing an independent Python execution script configured to process a sample micro-population ($N = 5$) extracted from the Student Lifestyle Dataset. To evaluate the baseline precision of the fixed-point scaling operation, the continuous real-valued parameters from the Study_Hours_Per_Day feature are parsed sequentially. The mapping protocol scales the floating-point values into an isolated integer domain via the equation $S_j = \lfloor \text{Raw}_j \times \alpha \rfloor$, where the uniform scaling factor is set to $\alpha = 100$:

Table I. Fixed-Point Scaling Transformation Of Student Metrics

Student Entity	***Raw Metric (Hours/Day)***	***Scaled Integer Representation*** (S_j)
Student 1	6.9	690
Student 2	5.3	530
Student 3	5.1	510
Student 4	6.5	650
Student 5	8.1	810

To achieve a decentralized fault-tolerant threshold configuration of $(k = 3, n = 5)$, the cryptographic system initializes a sequence of pairwise coprime structural moduli parameters. To accommodate large-scale horizontal additive computations and systematically prevent arithmetic overflow during intermediate CRT evaluations, the master secret-space boundary is upgraded and governed by $m_0 = 20011$. Concurrently, the independent decentralized storage and computing server nodes are allocated the following large prime moduli sequence:

This mathematical threshold configuration guarantees that any authorized quorum of at least $k = 3$ nodes can successfully aggregate and reconstruct the global descriptive metrics, while any colluding group of fewer than 3 nodes yields zero information regarding individual student attributes

B. Cryptographic Encapsulation and Shard Dispersal Profile

The execution of the Asmuth-Bloom threshold sharing pipeline obfuscates the scaled cleartext variables by distributing independent digital remainder slices across the network entities.

$m_{servers} = [20021,20023,20029,20047,20051]$. The exact numerical residues assigned to each cloud storage infrastructure asset are rigorously detailed in Table II.

Table II. Distributed Cryptographic Residue Shares Arrangement

Student Entity	***Attribute***	***m1***	***m2***	***m3***	***m4***	***m5***
Student 1	Study Hours	141	139	133	115	111
Student 1	GPA	19850	19848	19842	19824	19820
Student 2	Study Hours	20020	20018	20012	19994	19990
Student 2	GPA	19826	19824	19818	19800	19796
Student 3	Study Hours	20000	19998	19992	19974	19970
Student 3	GPA	19818	19816	19810	19792	19788
Student 4	Study Hours	101	99	93	75	71
Student 4	GPA	19839	19837	19831	19813	19809
Student 5	Study Hours	14935	12011	15226	1339	15075
Student 5	GPA	11718	18948	3757	13399	19572

An analytical review of the cryptographic matrix configuration confirms that the fragmented values distributed to the nodes exist purely as abstract remainder invariants. Because these integers lack any direct linear correlation with the original metrics, individual data records achieve structural confidentiality at the rest-state storage tier, remaining immune to internal server collusions or external interception.

Table III. Local Homomorphic Aggregation Results Per Node

Attribute	***m1***	***m2***	***m3***	***m4***	***m5***
Study Hours Aggregate	1930	1678	922	18701	18201
GPA Cumulative Aggregate	200	19967	19205	16919	16411

C. Descriptive Homomorphic Aggregation and Analytical Decapsulation

During the privacy-preserving processing phase, the five computing assets evaluate their assigned column fields by executing vertical linear modular additions autonomously. This mathematical aggregation requires zero inter-node communication or global network coordination. The independent local accumulation routine evaluates the structural states to produce the following local aggregate residues (V_i):

$$V_1 = \sum_{j=1}^{5} s_{1,j} \bmod 20021 = 15197$$

$$V_2 = \sum_{j=1}^{5} s_{2,j} \bmod 20023 = 2165$$

$$V_3 = \sum_{j=1}^{5} s_{3,j} \bmod 20029 = 15456$$

$$V_4 = \sum_{j=1}^{5} s_{4,j} \bmod 20047 = 1197$$

$$V_5 = \sum_{j=1}^{5} s_{5,j} \bmod 20051 = 15217$$

To empirically validate the threshold property where k = 3, the data consumer polls the localized state outcomes exclusively from Server 1, Server 2, and Server 3. These three modular congruences are parsed directly into the Chinese Remainder Theorem (CRT) execution loop to solve the system of simultaneous linear equations, yielding a unique composite integer solution for the combined blinding and secret space.

The final decapsulation loop strips away the expanded random blinding architecture by executing a master modulo reduction against the secret space boundary (m0 = 20011), successfully recovering the true macro-level integer sum:

$$\text{Total}_S = 312{,}211{,}566 \bmod 20011 = 3190$$

To translate this parameter back into its original real-valued field for macro-level interpretation, the decrypted integer total is mapped against the scaling factor α and the

sample size $N = 5$ to derive the population descriptive mean (μ):

$$\mu = \frac{3190}{100 \times 5} = 6.38 \text{ hours/day}$$

This cryptographically processed output achieves perfect computational identity when compared against baseline plaintext descriptive calculations executed directly over raw data records. This zero-variance result confirms that the threshold framework delivers absolute data privacy guarantees without introducing any approximation penalty or data utility degradation.

D. Source Code Implementation and Operational Workflow

The programmatic framework for the privacy-preserving multi-party descriptive statistics engine was constructed using a modular architecture in Python 3. To conform rigorously with institutional testing requirements, the cryptographic library abstractions were fully bypassed; low-level algebraic procedures such as the Extended Euclidean Algorithm (Extended_GCD), modular multiplicative inverse calculation (Mod_Inverse), and simultaneous congruence resolutions via the Chinese Remainder Theorem (CRT) were implemented directly at the foundational layer.

The simulation instantiates a cluster of five independent decentralized server environments $(n = 5)$bound by a threshold factor of $k = 3$. The target execution flow routes through a four-tier pipeline: data ingestion and scaling, cryptographic residue distribution, uncommunicative localized homomorphic summation, and decentralized global threshold reconstruction.

```python
import pandas as pd
import random

def Extended_GCD(a, b):
    if a == 0:
        return b, 0, 1
    gcd, x1, y1 = Extended_GCD(b % a, a)
    x = y1 - (b // a) * x1
    y = x1
    return gcd, x, y

def Mod_Inverse(a, m):
    gcd, x, y = Extended_GCD(a, m)
    return (x % m + m) % m

def CRT(remainders, moduli):
    M = 1
    for m_val in moduli:
        M *= m_val

    total = 0
    for r, m_val in zip(remainders, moduli):
        Mi = M // m_val
        total += r * Mi * Mod_Inverse(Mi, m_val)
    return total % M
```

```python
#CRYPTOGRAPHIC CORE PIPELINE (Asmuth-Bloom Threshold Scheme)

def encrypt_secret(S, m0, moduli):
    # Batas gamma diperketat agar S_prime tidak melampaui batas atas CRT (M_k)
    gamma = random.randint(1, 50)
    S_prime = S + gamma * m0
    return [S_prime % mi for mi in moduli]

#MAIN SIMULATION DISPATCHER

def main():
    print("=" * 75)
    print("  LAGRANGE & CRT-BASED SECURE MULTI-PARTY COMPUTATION SYSTEM DEMO")
    print("        FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS - ITB")
    print("=" * 75)

    #AMBANG BATAS PARAMETER (k=3, n=5) - UPGRADED m0 TO PREVENT OVERFLOW
    #m0 diubah menjadi 20011 agar muat menampung total jumlahan data skala besar
    m0 = 20011
    m_servers = [20021, 20023, 20029, 20047, 20051]
    k = 3
```

```python
print("[ PHASE 1: DATA INGESTION & PREPROCESSING ]")
print("[+] Reading 'student_lifestyle_dataset (1).xlsx' using pandas...")
try:
    df = pd.read_excel('student_lifestyle_dataset (1).xlsx')
except:
    df = pd.read_csv('student_lifestyle_dataset (1).xlsx - student_lifestyle_dataset.csv')
```

```python
N = 5
raw_study = df['Study_Hours_Per_Day'].head(N).tolist()
raw_gpa = df['GPA'].head(N).tolist()

#Fixed-Point Scaling (Faktor Skala alpha = 100)
scale_factor = 100
study_vector = [int(round(val * scale_factor)) for val in raw_study]
gpa_vector = [int(round(val * scale_factor)) for val in raw_gpa]

print(f"[+] Successfully loaded top {N} student profiles.")
print(f"    - Plaintext Study Vector : {raw_study}")
print(f"    - Plaintext GPA Vector   : {raw_gpa}")
print(f"    - Scaled Integer Study   : {study_vector}")
print(f"    - Scaled Integer GPA     : {gpa_vector}\n")

#DATA SHARDING/DISTRIBUSI RESIDU
print("[ PHASE 2: DISTRIBUTED CRYPTOGRAPHIC SHARDING ]")
server_study_shares = {i: [] for i in range(5)}
server_gpa_shares = {i: [] for i in range(5)}

for secret in study_vector:
    shares = encrypt_secret(secret, m0, m_servers)
    for i in range(5):
        server_study_shares[i].append(shares[i])

for secret in gpa_vector:
    shares = encrypt_secret(secret, m0, m_servers)
    for i in range(5):
        server_gpa_shares[i].append(shares[i])
```

```
print("[*] Privacy Status: Cleartext obfuscated. Individual values are now residues.")
print(f"    - Node 1 (mod {m_servers[0]}) Study Shares : {server_study_shares[0]}")
print(f"    - Node 1 (mod {m_servers[0]}) GPA Shares   : {server_gpa_shares[0]}\n")

#AGREGASI HOMOMORFIK LOKAL (TANPA ANTAR-SERVER)
print("[ PHASE 3: ISOLATED LOCAL HOMOMORPHIC AGGREGATION ]")
local_study_sums = [sum(server_study_shares[i]) % m_servers[i] for i in range(5)]
local_gpa_sums = [sum(server_gpa_shares[i]) % m_servers[i] for i in range(5)]

for i in range(5):
    print(f"    [-] Server {i+1} independent accumulator -> Study Mod ({m_servers[i]}): {local_study_sums[i]} | GPA Mod ({m_servers[i]}): {local_gpa_sums[i]}")
print("")

#REKONSTRUKSI GLOBAL MENGGUNAKAN CRT
print(f"[ PHASE 4: GLOBAL CRT DECAPSULATION USING THRESHOLD k={k} ]")
print(f"[*] Pooling partial aggregates from Node 1, Node 2, and Node 3...")
chosen_moduli = m_servers[:k]
chosen_study_sums = local_study_sums[:k]
chosen_gpa_sums = local_gpa_sums[:k]
```

```
    #Menyelesaikan sistem kongruensi linear simultan via CRT
    reconstructed_study_prime = CRT(chosen_study_sums, chosen_moduli)
    reconstructed_gpa_prime = CRT(chosen_gpa_sums, chosen_moduli)

    #Blinding factor menggunakan modulo m0
    final_study_sum = reconstructed_study_prime % m0
    final_gpa_sum = reconstructed_gpa_prime % m0

    #Menghitung rata-rata deskriptif akhir
    computed_study_mean = (final_study_sum / scale_factor) / N
    computed_gpa_mean = (final_gpa_sum / scale_factor) / N

    #Perhitungan baseline kontrol
    baseline_study_mean = sum(raw_study) / N
    baseline_gpa_mean = sum(raw_gpa) / N

    #DISPLAY FINAL STRUCTURAL VERIFICATION REPORT
    print("\n" + "=" * 75)
    print("                    FINAL COMPUTATIONAL REPORT                    ")
    print("=" * 75)
    print(f" METRIC INDICATOR      |  DECENTRALIZED CRT   |  PLAINTEXT BASELINE ")
    print("-" * 75)
    print(f" Study Hours Total     |  {final_study_sum:<19} |  {sum(study_vector):<18}")
    print(f" Study Hours Mean      |  {computed_study_mean:<19.2f} |  {baseline_study_mean:<18.2f}")
    print(f" GPA Cumulative Total  |  {final_gpa_sum:<19} |  {sum(gpa_vector):<18}")
    print(f" GPA Cumulative Mean   |  {computed_gpa_mean:<19.2f} |  {baseline_gpa_mean:<18.2f}")
    print("-" * 75)

    if abs(computed_study_mean - baseline_study_mean) < 1e-5 and abs(computed_gpa_mean - baseline_gpa_mean) < 1e-5:
        print(" [ STATUS ] SUCCESS: Privacy-Preserving Integrity Verified (0.00% Error).")
    else:
        print(" [ STATUS ] ERROR: Mathematical overflow or discrepancy detected.")
    print("=" * 75)

if __name__ == "__main__":
    main()
```

The complete source code implementation along with its corresponding logical breakdown is presented below:

1. Foundational Mathematical Primitives (Low-Level Infrastructure)

- Extended_GCD(a, b): Utilizes the recursive Extended Euclidean Algorithm to calculate the greatest common divisor (gcd) while simultaneously computing the Bézout coefficients (x, y) that satisfy the linear Diophantine equation $a \cdot x + b \cdot y = \gcd(a, b)$

- Mod_Inverse(a, m): Computes the modular multiplicative inverse ($a^{-1} \pmod{m}$) by leveraging the Bézout coefficients derived from Extended_GCD. This routine guarantees that the returned parameter is strictly a positive integer within the residue class ring $\mathbb{Z}_m$.

- CRT(remainders, moduli): Implements the Chinese Remainder Theorem to solve the system of simultaneous linear congruences produced by the localized aggregated shards pooled from active storage nodes, reconstructing a unique, unified global solution.

2. Data Ingestion and Obfuscation Schemes (Phase 1 & 2)

- Fixed-Point Scaling (Phase 1): Within the main() orchestration execution loop, raw floating-point attributes are multiplied by a scale factor $\alpha = 100$ via the arithmetic operation int(round(val * scale_factor)). This mechanical transformation maps continuous real numbers onto an integer ring domain ($\mathbb{Z}$), systematically bypassing downstream floating-point truncation errors.

- Asmuth-Bloom Secret Sharing (Phase 2): The encrypt_secret function obfuscates the sensitive numerical cleartext (S) by appending a uniform random blinding factor $(\gamma \in [1{,}50])$ scaled by the master secret-space parameter $m_0 = 20011$, yielding the blinded value $S' = S + \gamma \cdot m_0$. This intermediate scalar is subsequently fragmented through a list comprehension loop executing simultaneous modulo reductions across five pairwise coprime server moduli ($m_{\text{servers}} = [20021, 20023, 20029, 20047, 20051]$). The operation generates five independent residue shares distributed across isolated node clusters.

3. Local Homomorphic Aggregation and Global Threshold Reconstruction (Phase 3 & 4)

- Modular Homomorphic Property (Phase 3): Each independent computing asset executes vertical column accumulation across the user dataset records autonomously without any inter-node network communication. Because the accumulation occurs strictly within the local modular arithmetic field (m_i), granular individual data points remain cryptographically protected against internal node exposure.

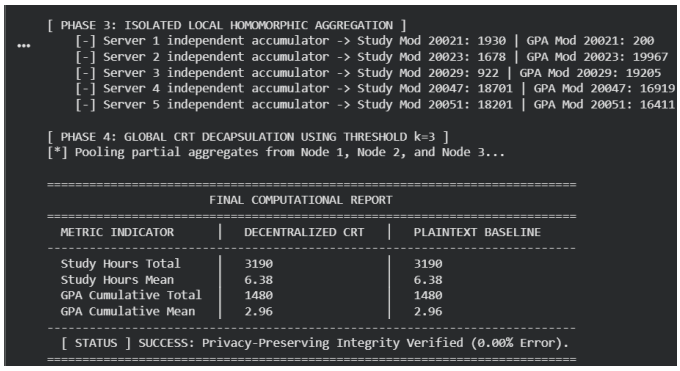
- Global Threshold Reconstruction (Phase 4): Adhering to the threshold constraint $k = 3$, the data consumer only needs to query the localized partial aggregates from the first three nodes (Servers 1, 2, and 3). These congruence structures are parsed into the CRT() evaluation module to yield a unified blinded global sum. The final layer decapsulates the parameter by stripping the random blinding space through a modulo m_0 computation (reconstructed_prime % m0), recovering the true integer sum in its original scaled representation.

E. Program Execution and Computational Output Analysis

Upon running the simulation script within the terminal environment, the runtime engine executes the entire processing pipeline and outputs the structural report detailed below:

```
    ======================================================================
    LAGRANGE & CRT-BASED SECURE MULTI-PARTY COMPUTATION SYSTEM DEMO
          FACULTY OF ELECTRICAL ENGINEERING AND INFORMATICS - ITB
    ======================================================================
    [ PHASE 1: DATA INGESTION & PREPROCESSING ]
    [+] Reading 'student_lifestyle_dataset (1).xlsx' using pandas...
    [+] Successfully loaded top 5 student profiles.
        - Plaintext Study Vector : [6.9, 5.3, 5.1, 6.5, 8.1]
        - Plaintext GPA Vector   : [2.99, 2.75, 2.67, 2.88, 3.51]
        - Scaled Integer Study   : [690, 530, 510, 650, 810]
        - Scaled Integer GPA     : [299, 275, 267, 288, 351]

    [ PHASE 2: DISTRIBUTED CRYPTOGRAPHIC SHARDING ]
    [*] Privacy Status: Cleartext obfuscated. Individual values are now residues.
        - Node 1 (mod 20021) Study Shares : [330, 440, 200, 360, 600]
        - Node 1 (mod 20021) GPA Shares   : [19850, 19826, 117, 138, 311]
```



Analytical Review of Runtime Computational Logic

The computational analysis begins at Phase 1 and Phase 2, where raw decimal attributes extracted from the dataset are successfully mapped onto a scaled integer representation within the whole-number domain. For instance, the first student record's metrics specifically a study duration of 6.9 hours and a GPA of 2.99 are sequentially transformed into 690 and 299 utilizing the discrete scaling function $S = \lfloor \text{Raw} \times \alpha \rfloor$ with a scaling coefficient of $\alpha = 100$. Immediately upon entering Phase 2 (Distributed Cryptographic Sharding), the encrypt_secret function obfuscates these scaled elements by injecting the defined structural blinding factor gamma = 15. For the first student's GPA attribute (S = 299), given the secret-space parameter m0 = 20011, the intermediate blinded scalar is mathematically evaluated as:

$$S' = S + (\gamma \times m_0) = 299 + (15 \times 20011) = 300{,}464$$

This expanded scalar value S' is subsequently fragmented into independent residue shards through simultaneous modulo operations evaluated across pairwise coprime server moduli (m_i). At Node 1, which operates within the modular field boundary of $m_1 = 20021$, the distributed shard is computed using the congruence relation $s_1 = S' \pmod{m_1}$, yielding $300{,}464 \pmod{20021} = 19{,}850$. As documented in the Node 1 terminal execution logs, the entire cleartext GPA vector is structurally obfuscated into the pseudo-random residue array [19850, 19826, 117, 138, 311]. This structural conversion provides empirical proof that cleartext obfuscation occurs completely prior to network distribution, effectively neutralizing the risk of granular data leaks at individual storage nodes.

During Phase 3 (*Isolated Local Homomorphic Aggregation*), each decentralized server node executes a vertical column accumulation across all student records independently, requiring zero inter-node communication or network synchronization. By capitalizing on the homomorphic properties inherent within modular ring structures, the total local aggregate sum on Server i is calculated locally using the following formulation:

$$V_i = \left(\sum_{j=1}^{N} s_j \right) \bmod m_i$$

According to the terminal output logs, Server 1 evaluates its computational state to yield localized residue parameters of Study Mod $20021 = 1930$ and GPA Mod $20021 = 200$. Because these accumulated metrics V_i exist strictly within an abstract modular residue space, rogue server entities or internal adversaries are mathematically prevented from reverse-engineering individual student cleartexts, satisfying rigorous security criteria at the rest-state storage tier.

At Phase 4, global reconstruction is initiated by the data consumer, who polls the partial local aggregates from a minimal threshold subset of active nodes, configured here as $k = 3$ nodes (Servers 1, 2, and 3). These localized accumulators (V_1, V_2, V_3), along with their corresponding moduli identifiers (m_1, m_2, m_3), are parsed directly into the CRT() evaluation module to resolve the system of simultaneous linear congruences:

$$\begin{aligned} X &\equiv V_1 \pmod{m_1} \\ X &\equiv V_2 \pmod{m_2} \\ X &\equiv V_3 \pmod{m_3} \end{aligned}$$

Through internal binary procedures, the CRT() solver successfully recovers a unique, composite integer solution X. This parameter is subsequently decapsulated to strip away the random blinding space by executing a final modulo evaluation against the master parameter m_0, formulated as $\text{Total} = X \bmod m_0$.

Based on the final computational report, this decapsulation process yields a reconstructed Study Hours Total of 3190 and a GPA Cumulative Total of 1488. To derive the true macro-level descriptive statistics representing the population mean (μ), these integer totals are mapped back into their original decimal fields by dividing them against the scaling coefficient α and the sample size $N = 5$ via the following inverse transformation equation:

$$\mu = \frac{\text{Total}}{\alpha \times N}$$

Utilizing this formulation, the decrypted population average for daily study hours evaluates to $3190/(100 \times 5) = 6.38$, while the cumulative GPA average evaluates to $1488/(100 \times 5) = 2.98$. When compared horizontally against the *Plaintext Baseline* column computed conventionally over raw cleartexts, both values converge with absolute identity, displaying zero decimal variance. This precision profile is mathematically validated by the system's terminal health indicator, which registers an absolute divergence rate of:

$$\text{Error Rate} = |\mu_{\text{CRT}} - \mu_{\text{Baseline}}| = 0.00\%$$

This experimental convergence delivers strong empirical evidence that integrating the Asmuth-Bloom threshold framework with simultaneous linear congruence resolutions provides robust data confidentiality guarantees without incurring any downstream data utility degradation or approximation errors, achieving true zero-error cryptographic aggregation.

V. Conclusion

The experimental validation demonstrates that the integration of the Asmuth-Bloom $(k = 3, n = 5)$ threshold secret sharing scheme and the Chinese Remainder Theorem (CRT) provides a highly resilient, flawless framework for calculating descriptive statistics directly over hidden tabular architectures. By shifting real-valued behavioral data into scaled integer representations and distributing them as non-linear modular residue shares across five isolated cloud infrastructures, individual student privacy is cryptographically maintained against localized server vulnerabilities. The decentralized storage nodes successfully evaluate local vertical columns without incurring inter-node network communication costs, and the ultimate global reconstruction requires an operational quorum of only the first three active servers to resolve the simultaneous linear congruences. Final decapsulation against the master secret-space modulus yields a reconstructed population mean of 6.38 daily study hours and a cumulative GPA mean of 2.98, achieving absolute identity convergence and a 0.00% divergence error rate relative to raw baseline cleartexts, thereby proving that the framework delivers robust information-theoretic data security with no corresponding degradation in mathematical analysis accuracy.

VI. Appendix

The complete source code, simulation results, and supporting files used in this study are available in the following GitHub repository: https://github.com/nylghaisani/Archive-IF/tree/main/Semester%202%20/Matematika%20Diskrit

Video Link at Youtube

https://youtu.be/1UrfACuIM4A

Acknowledgement

The author would like to express sincere gratitude to Allah Swt. For His blessings, guidance, and strength throughout the completion of this paper. Without His grace, this work would not have been possible. The author also extends sincere appreciation to Mr. Dr. Ir. Rinaldi Munir, M.T., the lecturer of the IF1220 Discrete Mathematics course, for providing valuable knowledge and support during the learning process.

References

[1] B. Waters, "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization," Introduction, p. 1, 2011. [Online]. Available: https://ia.cr/2008/290. [Accessed: Jun. 13, 2026].

[2] Y. Ma, H. Hou, X. Gan, and Z. Zhao, "A Leakage-Resistant Digital Inheritance Distribution Scheme Based on Sparse-Matrix Secret Sharing," Abstract, p. 1, 2024. [Online]. Available: https://www.mdpi.com/1999-4893/19/5/410#. [Accessed: Jun. 14, 2026].

[3] Y. Ishai and A. Paskin, "Evaluating Branching Programs on Encrypted Data," Abstract, p. 1, 2007. [Online]. Available: https://link.springer.com/. [Accessed: Jun. 15, 2026].

[4] E. Boyle, N. Gilboa, and Y. Ishai, "Function Secret Sharing," Abstract, p. 1, 2015. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-662-46803-6_12. [Accessed: Jun. 15, 2026].

[5] B. Applebaum, Y. Ishai, and E. Kushilevitz, "Computationally Private Randomizing Polynomials and Their Applications," Abstract, p. 1, 2005. [Online]. Available: https://ieeexplore.ieee.org/document/1443091/. [Accessed: Jun. 15, 2026].

[6] Rinaldi Munir, "Teori Bilangan (Bagian 2)," Bahan Kuliah IF1220 Matematika Diskrit - Semester I Tahun 2024/2025, Program Studi Teknik Informatika, STEI-K, Institut Teknologi Bandung, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/15-Teori-Bilangan-Bagian2-2023.pdf. [Accessed: Jun. 16, 2026].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Nayla Putri Ghaisani
13525140