Seiyun University, Yemen

Subject Name

SPECIAL TOPICS IN

INFORMATION SECURITY

Prepared by :

walaa Abdaruhman

Registration number

2202041073

information security

Lab Assignment No. 04

Secure Multi-Party Computation (SMPC)

Q1 What is the main purpose of Secure Multi-Party Computation?

A1. To compute a function jointly using private without revealing those inputs to each other.

Q2. How does Secret sharing ensure privacy?
A2. it splits a secret into shares. A single share reveals no information; only a sufficient number (k) can reconstruct the secret.

Q3 What is the difference between SMPC and Homomorphic Encryption?

- SMPC:- requires multiple parties to collaborate
- Homomorphic Encryption allows computation on encrypted data by a single party (e.g a cloud server)

Q4: Explain one real-word application of SMPC?
Ans. A Sealed-bid auction, where the winner is determined whithout revealing any individual bid.

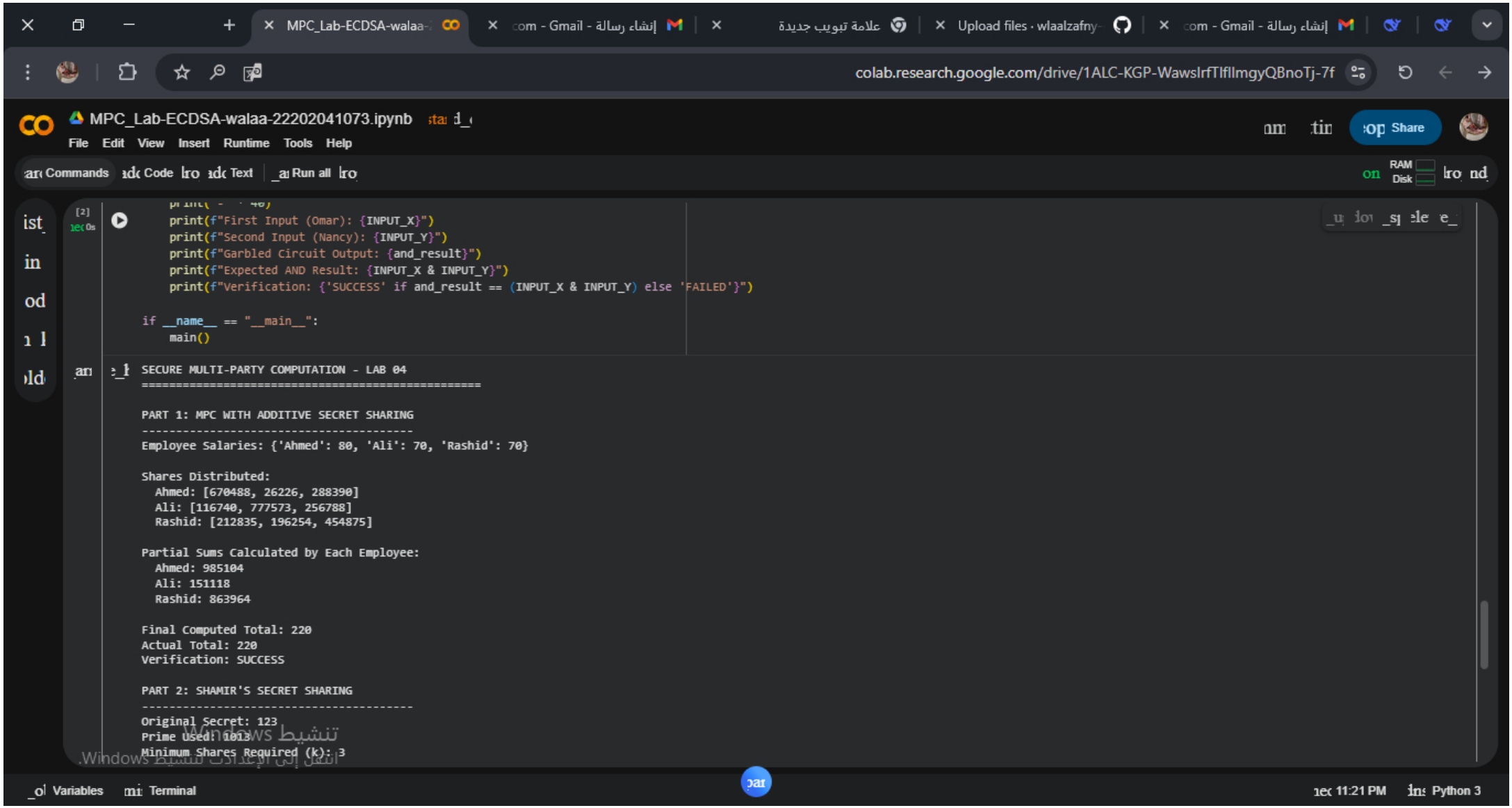
Q5:- What happens if fewer than k shares are available in sharing?
A5: it is impossible to reconstruct the secret or gain any information about it.

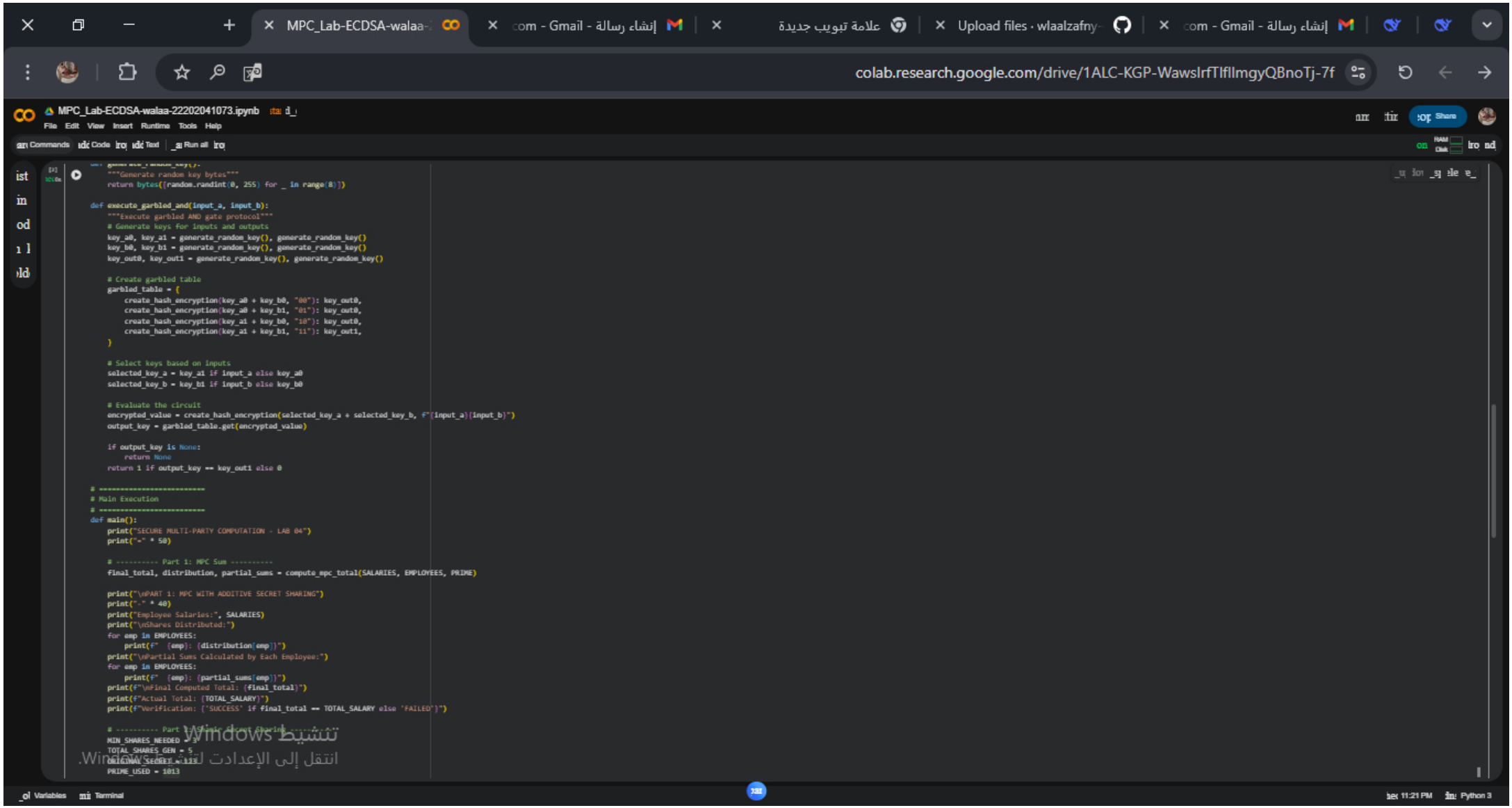
Q7: How does MPC enhance privacy in blockchain sy.?

A7: It enables private execution of smart contracts and calculations on the blockchains onwithout making the underlying data public.

Q8: Describe your experiment and observations from the MPC sum code?

A8: We calculated the total sum of private salaries. Each person split their salary into random shares. After exchanging and summing shares, we got the total, but no individual's salary.


MPC_Lab-ECDSA-walaa-22202041073.ipynb
File Edit View Insert Runtime Tools Help
Share
colab.research.google.com/drive/1ALC-KGP-WawslrfTIfIlmgyQBnoTj-7f
print(f"First Input (Omar): {INPUT_X}")
print(f"Second Input (Nancy): {INPUT_Y}")
print(f"Garbled Circuit Output: {and_result}")
print(f"Expected AND Result: {INPUT_X & INPUT_Y}")
print(f"Verification: {'SUCCESS' if and_result == (INPUT_X & INPUT_Y) else 'FAILED'}")
if __name__ == "__main__":
main()
SECURE MULTI-PARTY COMPUTATION - LAB 04
PART 1: MPC WITH ADDITIVE SECRET SHARING
Employee Salaries: {'Ahmed': 80, 'Ali': 70, 'Rashid': 70}
Shares Distributed:
Ahmed: [670488, 26226, 288390]
Ali: [116740, 777573, 256788]
Rashid: [212835, 196254, 454875]
Partial Sums Calculated by Each Employee:
Ahmed: 985104
Ali: 151118
Rashid: 863964
Final Computed Total: 220
Actual Total: 220
Verification: SUCCESS
PART 2: SHAMIR'S SECRET SHARING
Original Secret: 123
Prime Used: 1013
Minimum Shares Required (k): 3
Variables
Terminal
Python 3


MPC_Lab-ECDSA-walaa-22202041073.ipynb
File Edit View Insert Runtime Tools Help
colab.research.google.com/drive/1ALC-KGP-WawslrfTIfIlmgyQBnoTj-7f
"""Generate random key bytes"""
return bytes([random.randint(0, 255) for _ in range(8)])
def execute_garbled_and(input_a, input_b):
"""Execute garbled AND gate protocol"""
Generate keys for inputs and outputs
key_a0, key_a1 = generate_random_key(), generate_random_key()
key_b0, key_b1 = generate_random_key(), generate_random_key()
key_out0, key_out1 = generate_random_key(), generate_random_key()
# Create garbled table
garbled_table = {
create_hash_encryption(key_a0 + key_b0, "00"): key_out0,
create_hash_encryption(key_a0 + key_b1, "01"): key_out0,
create_hash_encryption(key_a1 + key_b0, "10"): key_out0,
create_hash_encryption(key_a1 + key_b1, "11"): key_out1,
}
# Select keys based on inputs
selected_key_a = key_a1 if input_a else key_a0
selected_key_b = key_b1 if input_b else key_b0
# Evaluate the circuit
encrypted_value = create_hash_encryption(selected_key_a + selected_key_b, f"{input_a}{input_b}")
output_key = garbled_table.get(encrypted_value)
if output_key is None:
return None
return 1 if output_key == key_out1 else 0
# ==========
# Main Execution
# ==========
def main():
print("SECURE MULTI-PARTY COMPUTATION - LAB 04")
print("=" * 50)
# ---------- Part 1: MPC Sum ----------
final_total, distribution, partial_sums = compute_mpc_total(SALARIES, EMPLOYEES, PRIME)
print("\nPART 1: MPC WITH ADDITIVE SECRET SHARING")
print("-" * 40)
print("Employee Salaries:", SALARIES)
print("\nShares Distributed:")
for emp in EMPLOYEES:
print(f" {emp}: {distribution[emp]}")
print("\nPartial Sums Calculated by Each Employee:")
for emp in EMPLOYEES:
print(f" {emp}: {partial_sums[emp]}")
print(f"\nFinal Computed Total: {final_total}")
print(f"Actual Total: {TOTAL_SALARY}")
print(f"Verification: {'SUCCESS' if final_total == TOTAL_SALARY else 'FAILED'}")
MIN_SHARES_NEEDED = 3
TOTAL_SHARES_GEN = 5
PRIME_USED = 1013
Variables
Terminal
11:21 PM
Python 3