# CS489/689 Privacy, Cryptography, Network and Data Security

## MPC, and PSI

Winter 2023, Tuesday/Thursday 8:30-9:50am

# Construct generic multi-party computations?

# Composing Protocols with Additive Shares

- Let there be values u and v
  - Alice has $u_A$ and $v_A$, Bob has $u_B$ and $v_B$
  - $u = u_A + u_B$, $v = v_A + v_B$
- Compute $s = s_A + s_B = u + v$
  - Alice computes $s_A = u_A + v_A$
  - Bob computes $s_B = u_B + v_B$
- Compute $t = t_A + t_B = u*v$
  - See exercise

# Composing Protocols with Additive Shares

- Let there be values u and v
  - Alice has $u_A$ and $v_A$, Bob has u
  - $u = u_A + u_B$, $v = v_A +$
- Compute $s = s_A + s$
  - Ali
  - Bob com
- Co $t_B$ v
  - See ex rcise

**Break it down**

# Composing Protocols with Additive Shares

**Goal: Compute the sum of two values**

# Composing Protocols with Additive Shares

**Goal: Compute the sum of two values**

**Catch:** neither value can be shared

# Composing Protocols with Additive Shares

**Goal: Compute the sum of two values**

**Catch:** neither value can be shared

**Let** the values be **u** and **v**.

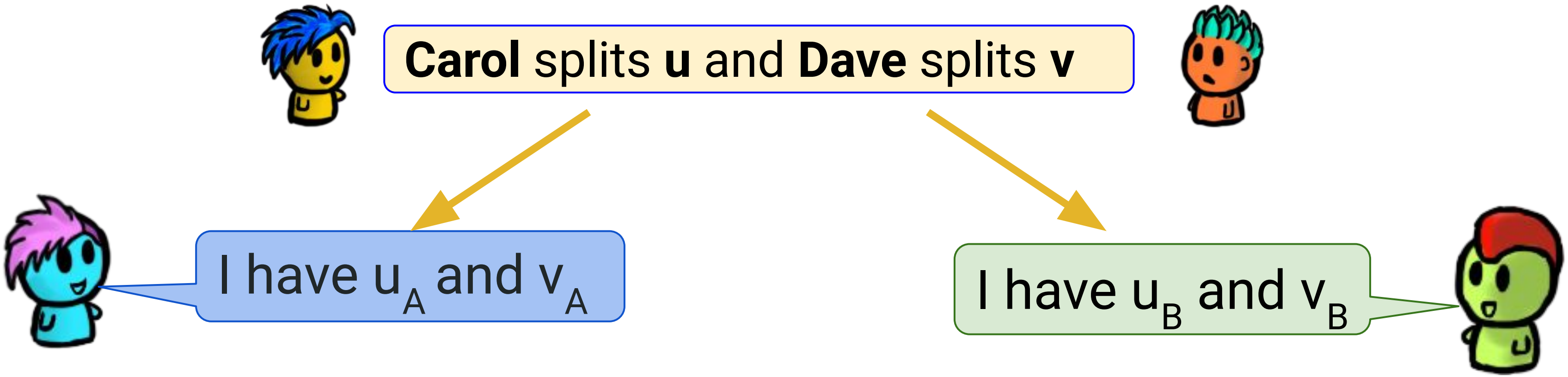
**Carol** splits **u** and **Dave** splits **v**

# Composing Protocols with Additive Shares

**Goal: Compute the sum of two values**

**Catch:** neither value can be shared

**Let** the values be **u** and **v**.



# Composing Protocols with Additive Shares

**Goal: Compute the su**

**Catch:** shared

**What next?**

ues be **u** and **v**.

arol splits **u** and **Dave** splits **v**

I have $u_A$ and $v_A$

I have $u_B$ and $v_B$

# Computing the Sum "Secretly"

Compute $S_A = u_A + v_A$

# Computing the Sum of U and V "Secretly"

Compute $S_A = u_A + v_A$

Compute $S_B = u_B + v_B$

**Since:** $u = u_A + u_B$ and $v = v_A + v_B$ **then**

$$S_A + S_B = U + V$$

# Computing the Sum of U and V “Secretly”

Compute $S_A = u_A + v_A$

Compute $S_B = u_B + v_B$

**Since:** $u = u_A + u_B$ and $v = v_A + v_B$ **then**

$$S_A + S_B = U + V$$

**Thus** we learn the sum of **u** and **v** without revealing either individual value

# Computing the Sum of U and V “Secretly”

Compute $S_A = u + v$

...te $S_B = u_B +$ ...

B

**Q: what is the trust model?**

**Si...e:** $u = u_A + u_B$ and $v = v_A + v_B$ **then**

$$S_A + S_B = U + V$$

**Thus** we learn the sum of **u** and **v** without revealing either individual value

# Compose, but no Carol or Dave

- Alice has x, Bob has y
  - Alice create $x_A + x_B$, Bob $y_A + y_B$
- They execute a number of addition and multiplication protocols
  - All intermediate outputs are uniformly random to the respective party
  - All intermediate outputs allow to continue performing additions and multiplications

# Compose, but no Carol or Dave

Have x

Have y

# Compose, but no Carol or Dave

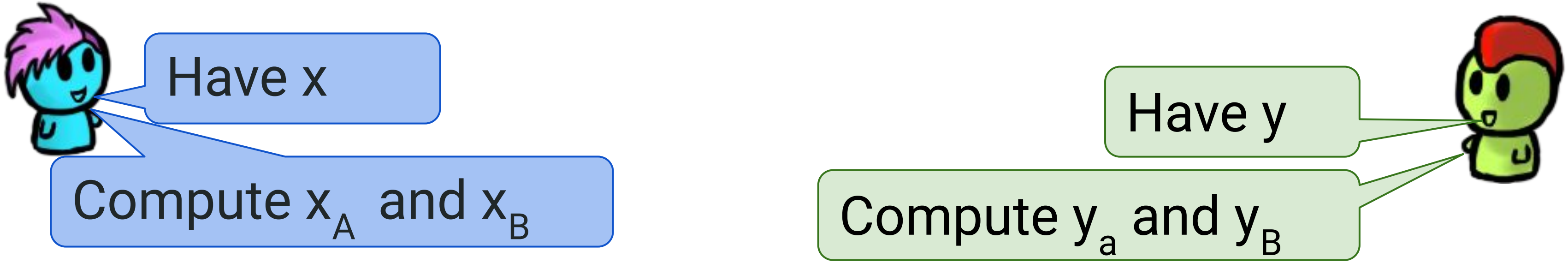
Have x

Compute $x_A$ and $x_B$

Have y

Compute $y_a$ and $y_B$

# Compose, but no Carol or Dave




- They can execute a number of addition and multiplication protocols
  - All intermediate outputs are uniformly random to the respective party
  - All intermediate outputs allow to continue performing additions and multiplications

# Compose, but no Carol or Dave

Have x

Compute $x_A$ and $x_B$

Have y

Compute $y_a$ and $y_B$

**Reconstruction:**

- There is a result $r = r_A + r_B$
- Alice sends $r_A$ to Bob
- Bob sends $r_B$ to Alice (if they agreed on this)

# Exercise: Design a protocol to compute $z_A$ (for Alice) and $z_B$ (for Bob)

- Alice has $x_A$, $y_A$; Bob has $x_B$, $y_B$
  - $x = x_A + x_B$; $y = y_A + y_B$
- The goal is to compute $z_A + z_B = x*y$
- Alice has the private key to an (additive) homomorphic enc. scheme $E_A()$ (e.g. Paillier's encryption)
  - Alice can perform $D_A(E_A(x)) = x$
  - $D_A(E_A(x) * E_A(y)) = x+y$
- Bob has the public key to Alice's private key
  - Bob can perform $c = E_A(x)$ (but not DA(c))

# Towards Proving Passive Security

- Let $VIEW_A$ be Alice's **view** during a multi-party computation
  - All messages received by Alice
- Let $SIM_A$ be a randomized algorithm (simulator) that outputs (a "guess" of) $VIEW_A$
- Give Alice's input x and output z (of the multi-party computation) as input to the simulator $SIM_A(x, z)$
- If $SIM_A(x, z)$ = (indistinguishable) $VIEW_A(x, y)$, then Alice cannot learn anything beyond x, z (about y)
  - What does indistinguishable mean?

# Indistinguishability

- Let D and E be two distributions
- Information-theoretic indistinguishability
  - D = E
    - Example: One-time pad as before

# Indistinguishability

- Let D and E be two distributions
- Information-theoretic indistinguishability
  - D = E
    - Example: One-time pad as b

**Sounds great…but not always an option**

# Indistinguishability Con't

- Computational indistinguishability
  - Let A be any polynomial-time algorithm
  - $Pr[A(x \leftarrow D) = 1] - Pr[A(x \leftarrow E) = 1]$ is negligible in the security parameter (smaller than any polynomial as long as the parameter is large enough)
    - Example: Let r be a random number. $E_A(x)$ is computationally indistinguishable from $E_A(r)$ (recall semantic security)

# Overview

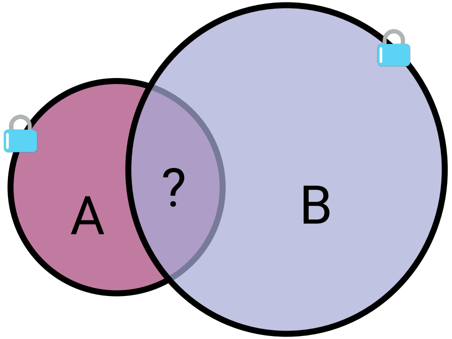
- **Two-party** computation requires (public-key) computational assumptions
- **Multi-party** computation can be implemented using only information-theoretic assumptions
- Protocols using information-theoretic assumptions are often faster than ones using computational assumptions
- However, the more parties, the slower the protocol

# PSI

# Private Set Intersection (PSI)

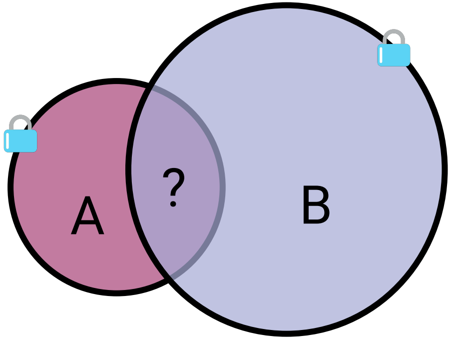
- Alice has set $X = \{x_1, x_2, x_3, \ldots, x_n\}$
- Bob has set $Y = \{y_1, y_2, y_3, \ldots, y_m\}$
- They want to compute $Z = X \cap Y$ (but reveal nothing else)
- This is an instance of a two-party computation of a specific function

# Private Set Intersections



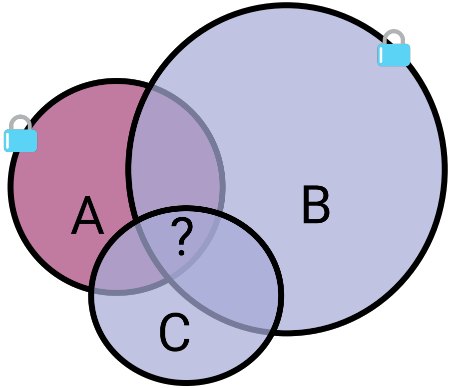

2-Party, One-Way PSI

A ⟶ B



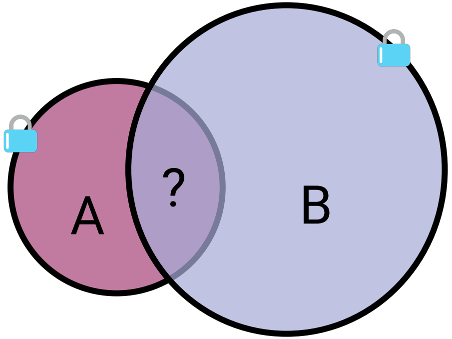

2-Party, Two-Way PSI

A ⟷ B



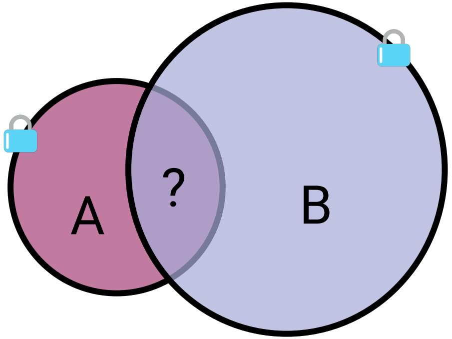

n-Party PSI

# Private Set Intersections



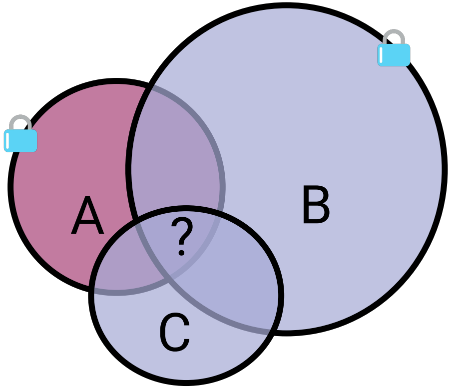

2-Party, One-Way PSI

A → B




2-Party, Two-Way PSI

A ↔ B




n-Party PSI

**Directionality** | **Reducing Information** | **Multi-party** | **Varying Guarantees**

# Strawman Protocol

- Alice permutes her set X, Bob permutes his set Y
- For each $x \in X$
  - For each $y \in Y$
    - Compute x =? y
- Protocol for comparison x =? y
  - Alice → Bob: $E_A(x)$
  - Bob: Choose r. $c = (E_A(x) * E_A(-y))$^r
  - Bob → Alice: c
  - Alice: Output $x = y$, if $D_A(c) = 0$, else $x \neq y$

# Exercise

Act.

- Prove the security of the comparison protocol against passive adversaries

# Improved Protocol

- The complexity of the previous protocol is O(nm)
- Wlog $m \leq n$
- If you hash n elements into a table with n bins with high probability the maximum number of elements in a bin is O(log n) (Balls-to-bins problem)

# Improved Protocol Con’t

- Improved protocol
  - Alice hashes elements to table with n bins
  - Alice pads each bin to O(log n) elements with dummy elements
  - Bob hashes elements to table with n bins
  - Bob pads each bin to O(log m) elements with dummy elements
  - Alice and Bob use comparison protocol for each pair in their respective bins
  - Complexity O(n log n log m)

# Precomputation

- The cryptographic operations of the comparison protocol can be precomputed
- Alice: Choose r
- Alice → Bob: $E_A(r)$
- Bob: Choose b and s
- Bob → Alice: $c = (E_A(r)^s) * E_A(-b)$
- Alice: Set $a = D_A(c)$
- It holds

  $a+b = r*s$

# Comparison Protocol After Precomputation

- Alice has a, r.  Bob has b, s: $a+b = r*s$
- Alice $\longrightarrow$ Bob: $c = a + x$
- Bob: $d = (c - y)/r$
- Bob $\longrightarrow$ Alice: d
- Alice: If $d = s$, then output $x = y$, else output $x \neq y$

# OPRF Comparison Protocol

- Let H be a cryptographic hash function
- Alice has x
- Bob has y and a key k
- Alice: Choose r
- Alice → Bob: H(x)^r
- Bob → Alice: c = H(x)^(r k), d = y^k
- Alice: If d = c^($r^{-1}$), output x=y, else x≠y

# Exercise

- Prove the OPRF comparison protocol secure against passive adversaries
- Hint
  - Recall the Decisional Diffie-Hellman assumption for $g^a$, $g^b$, $g^c$
  - Let $H(x)$ output $g^a$ (programmable random oracle assumption)

# PSI using OPRF comparison

- Alice has $X = \{ x_1, x_2, x_3, \ldots, x_n \}$
- Bob has $Y = \{ y_1, y_2, y_3, \ldots, y_m \}$ and a key k
- Alice and Bob compute OPRF protocol for each $x \in X$
  - Alice obtains $H(x_i)^k$
- Bob sends $H(y_j)^k$
- Alice compares each $H(x_i)^k$ and $H(y_j)^k$
  - Use a hash table
- This protocol has complexity $O(n + m)$

# Exercises related to Homomorphic Encryption

# Exercise I

- Write a small program that tries to break the discrete logarithm in a prime group with 2048 bits
  - Choose a prime p
  - Set a + b = 2, 4, 8, 16, …
  - Compute $c = g^{a+b} \pmod{p}$
  - Try i = 1, 2, 3, …
    - If $g^i = c$, stop and output i

# Exercise II

- Study the baby-step, giant-step algorithm
    - https://en.wikipedia.org/wiki/Baby-step_giant-step
- What is the complexity of the algorithm?
- When computing over number $a, b \in \{0, D\}$, $D \ll p$
    - Does the baby-step, giant-step algorithm help?

# Exercises related MPC

# Exercise

- Let $x \in \{0,1\}^n$ be a binary string of length n
- Let $y \in \{0,1\}^m$ be a binary string of length m
- Let f: $x \rightarrow y$ be "any" function
- Prove that f can be construction from only AND and XOR gates
- Hint
    - Show it for the case n=1 and m=1
    - Show that n' = n+1 can be constructed from solution for n
    - Show that m' = m+1 can be constructed from solution for m

# Exercises related to PSI

# Exercise I

- Alice has $X = \{ x_1, x_2, x_3, \ldots, x_n \}$
- Bob has $Y = \{ y_1, y_2, y_3, \ldots, y_m \}$ and a key k
- They want to compute $|X \cap Y|$, i.e., the size of the intersection (only)
- Design a protocol for this
- Hint: Alice cannot distinguish $H(x_i)^k$ from $H(x_j)^k$

# Exercise II

- Alice has $X = \{ x_1, x_2, x_3, \ldots, x_n \}$
- Bob has pairs (elements with payload) $Y = \{ (y_1, p_1), (y_2, p_2), (y_3, p_3), \ldots, (y_m, p_m) \}$ and (at least) a key k
- They want to compute $\Sigma p_j$ over $\{p_j \mid y_j \in X \}$
- Design a protocol for this
- Hint: Recall Paillier's encryption

# Security proof of generic protocol

- Addition
  - Trivial no messages are exchanged, simulator outputs empty view
- Multiplication
  - Alice → Bob: $E_A(x_A)$, $E_A(y_A)$
  - Bob: Choose r. Set $c = E_A(x_A)^\wedge y_B * E_A(y_A)^\wedge x_B * E_A(x_B y_B - r)$. Set $z_B = r$
  - Bob → Alice: c
  - Alice: $z_A = D_A(c) + x_A * y_A$

# Security proof of generic protocol con't

- Simulator for Bob
  - Choose s, t.  Output EA(s), EA(t)
    - Recall computational indistinguishability (semantic security) – Bob does not have private key
- Simulator for Alice
  - Choose s.  Output s
    - Recall one-time pad – Bob chooses s