De Cifris Trends
in
Cryptographic Protocols

University of Trento and *De Componendis Cifris*

October 2023

Lecture 7

Carlo Blundo Università degli Studi di Salerno

Private Set Intersection

Carlo Blundo

Università degli Studi di Salerno

Problem Defintion

Private Set Intersection

- Two participant Alice and Bob have a set of values S_A and S_B taken from a universe U
- Alice wants to compute the values in common with Bob, i.e., $S_A \cap S_B$
- Both want to preserve the *privacy* of the values in their sets

Is PSI needed?

Is PSI needed?

Agenzia Entrate

IRS (Internal Revenue Service)

Foreign Bank

Is PSI needed?

Learn if suspected tax evaders have bank accounts

Agenzia Entrate

IRS (Internal Revenue Service)

Foreign Bank

Is PSI needed?

Learn if suspected tax evaders have bank accounts

IRS (Internal Revenue Service)

Foreign Bank

Central Intelligence Agency

Italian Secret Service

Is PSI needed?

Learn if suspected tax evaders have bank accounts

IRS (Internal Revenue Service)

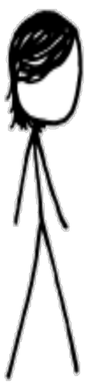
Foreign Bank

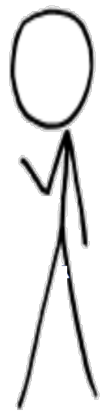
Compare databases of terrorist suspects

Central Intelligence Agency

Italian Secret Service

Secure Two-Party Computation Model





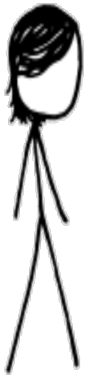
Alice: S_A

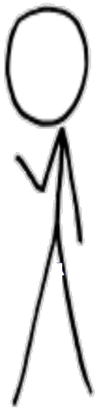
Bob: S_B

Ideal World

Secure Two-Party Computation Model

Trusted Third Party



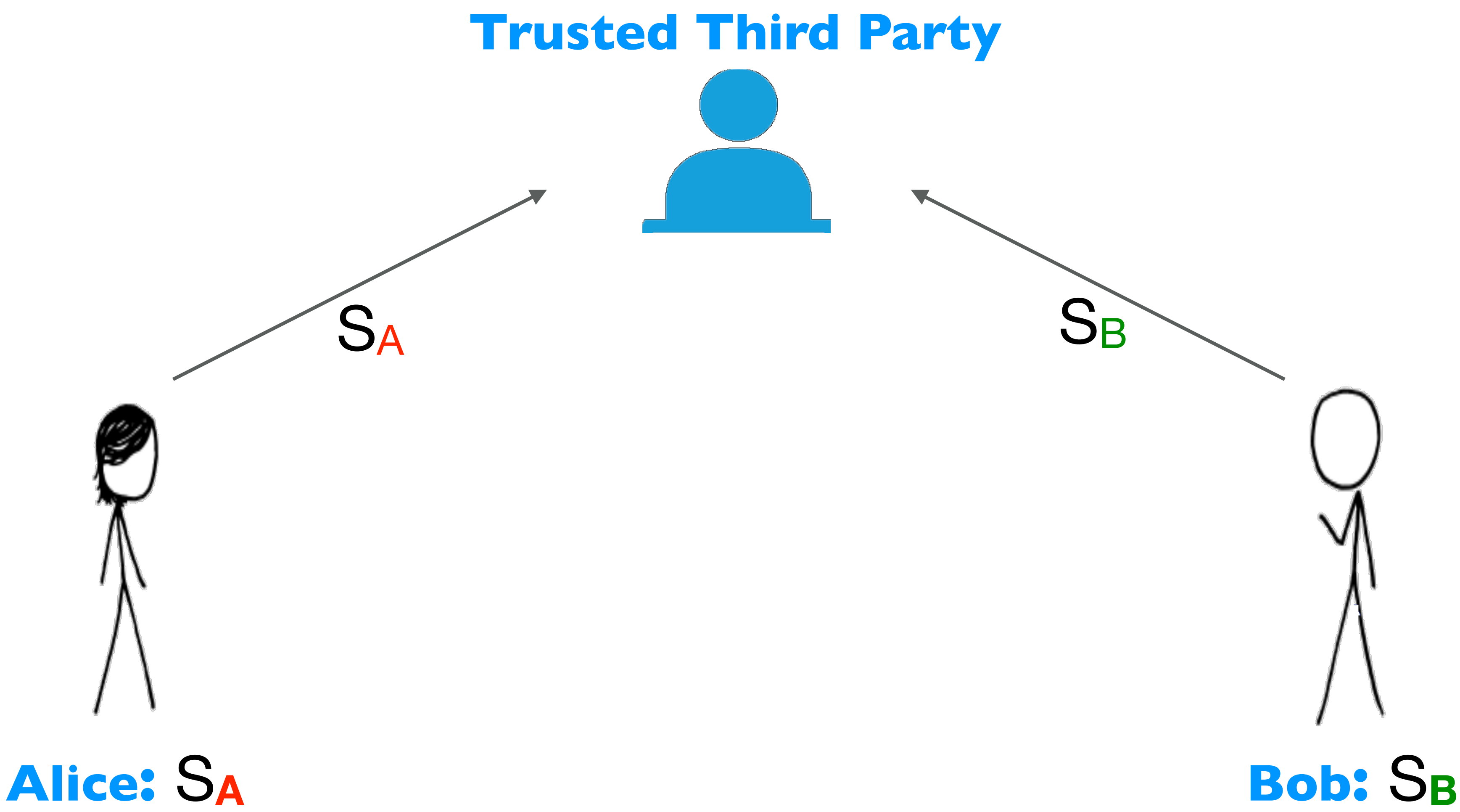


Alice: S_A

Bob: S_B

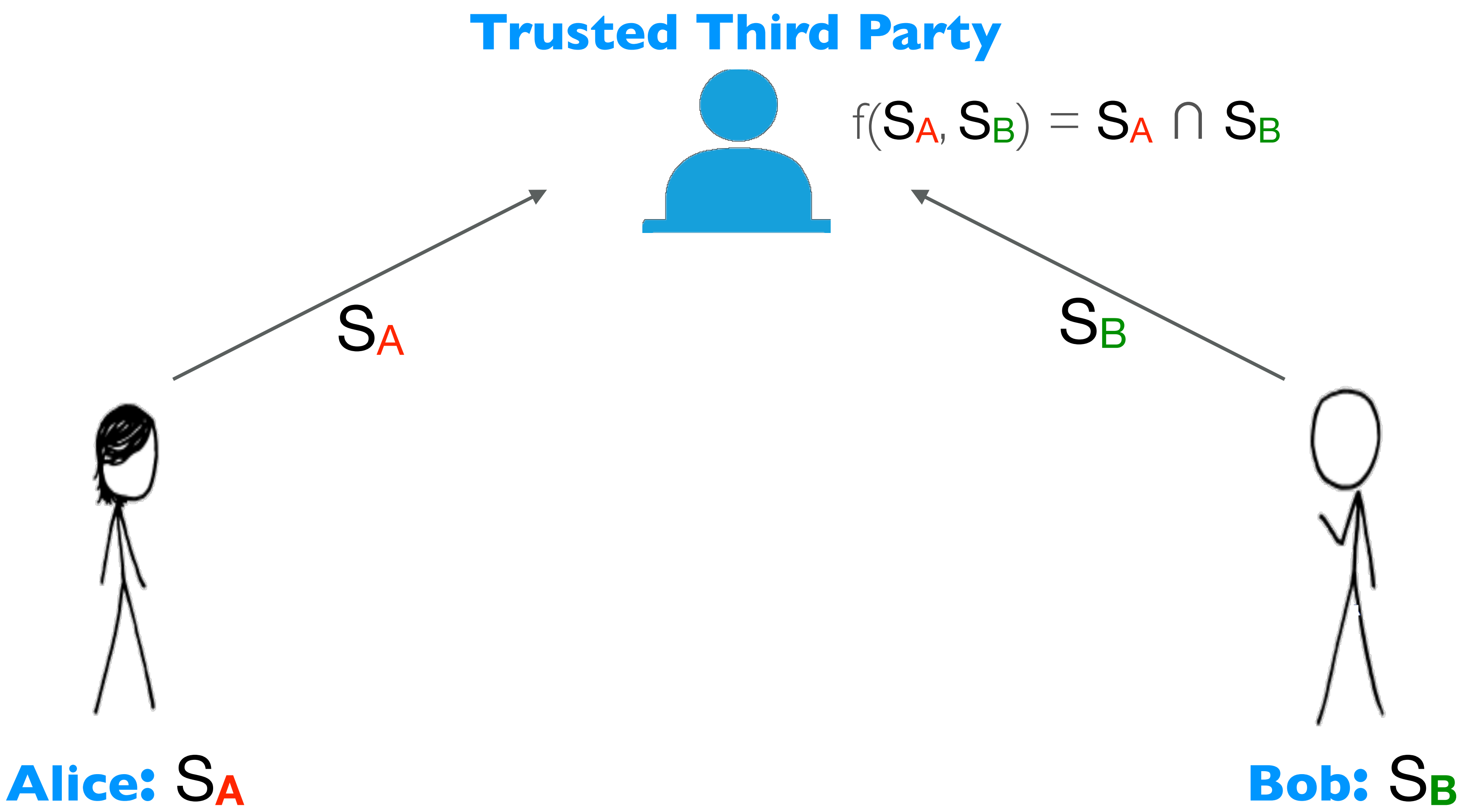
Ideal World

Secure Two-Party Computation Model



Ideal World

Secure Two-Party Computation Model

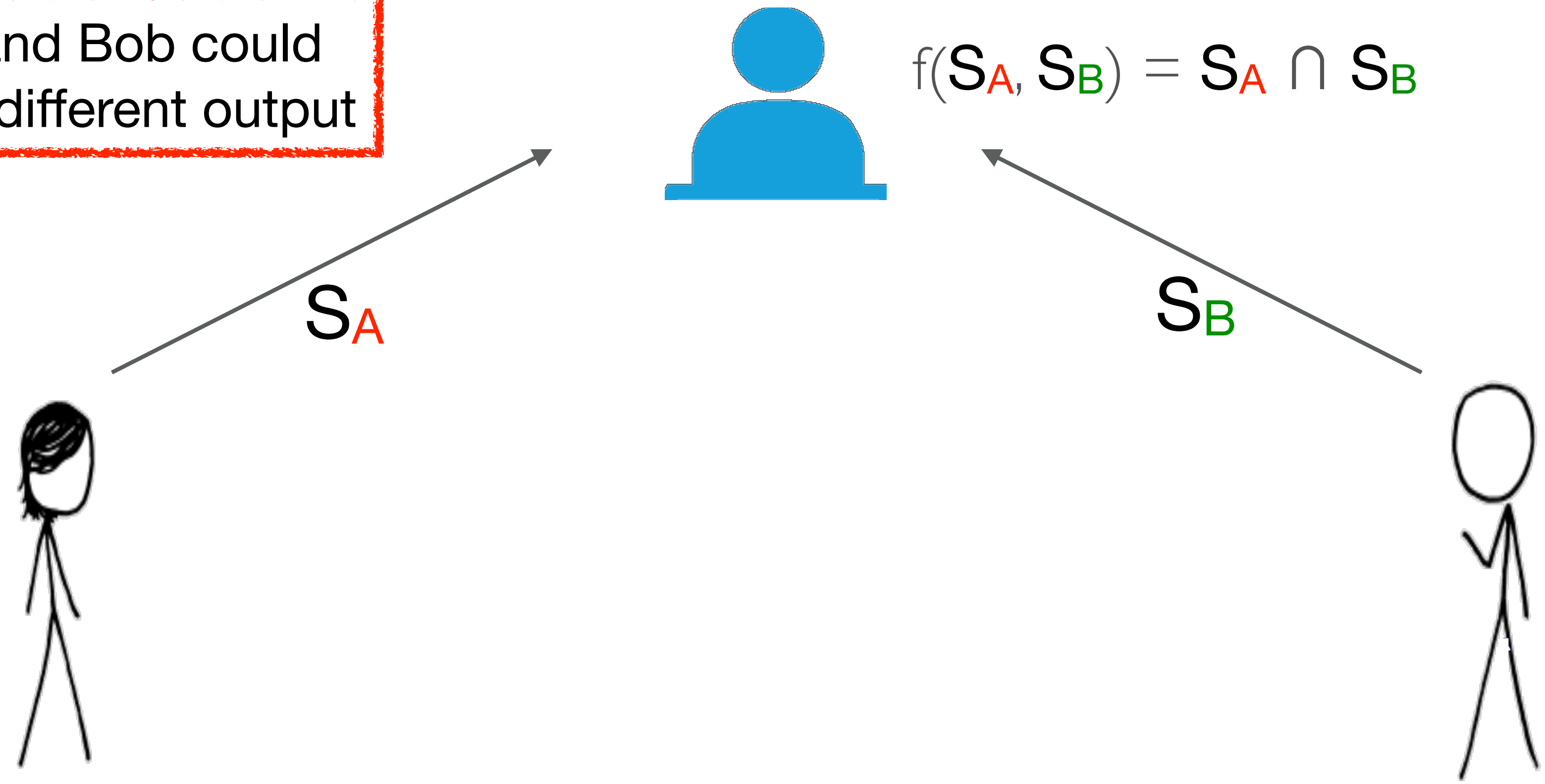


Ideal World

Secure Two-Party Computation Model

Trusted Third Party

Alice and Bob could receive different output



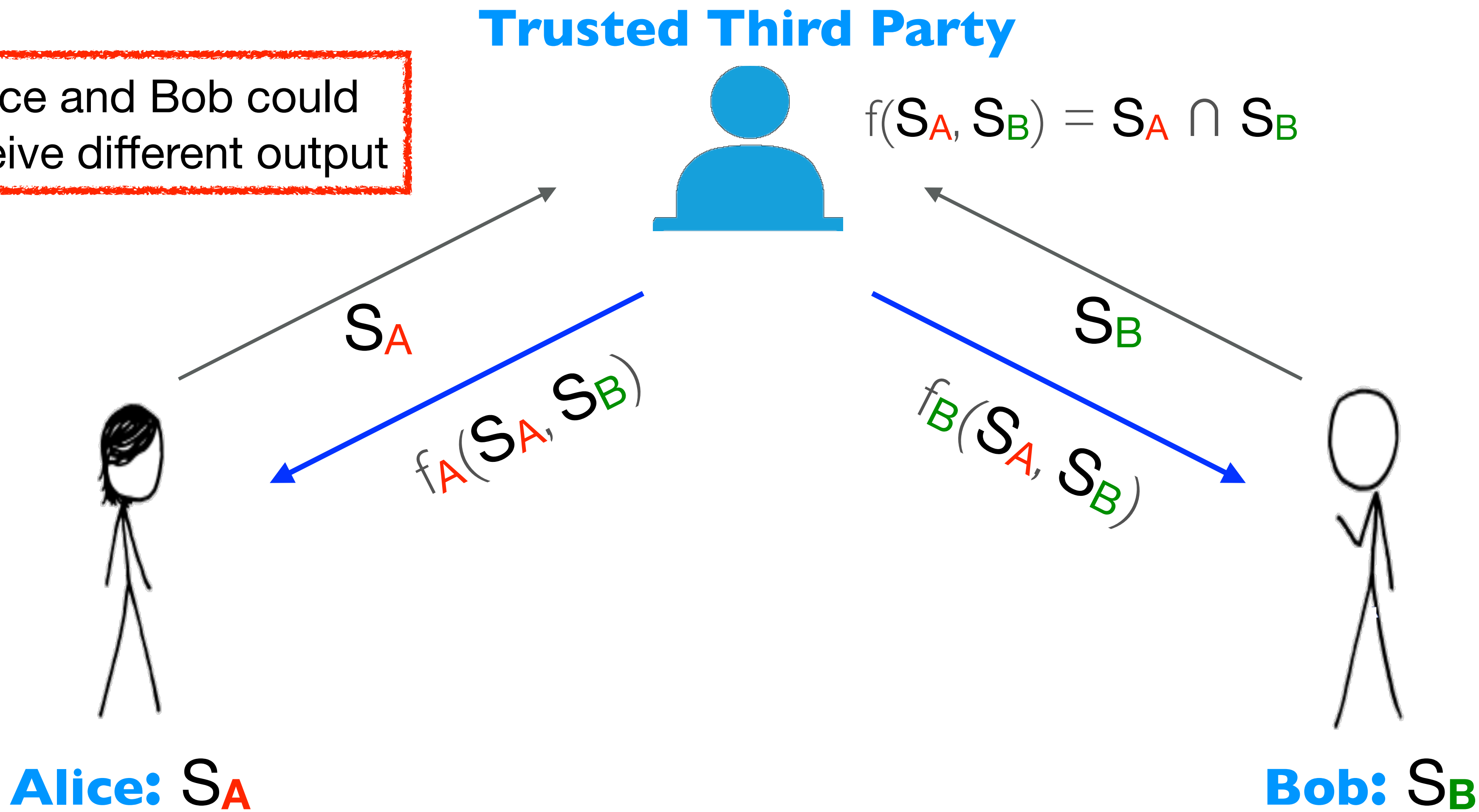
Alice: S_A

Bob: S_B

Ideal World

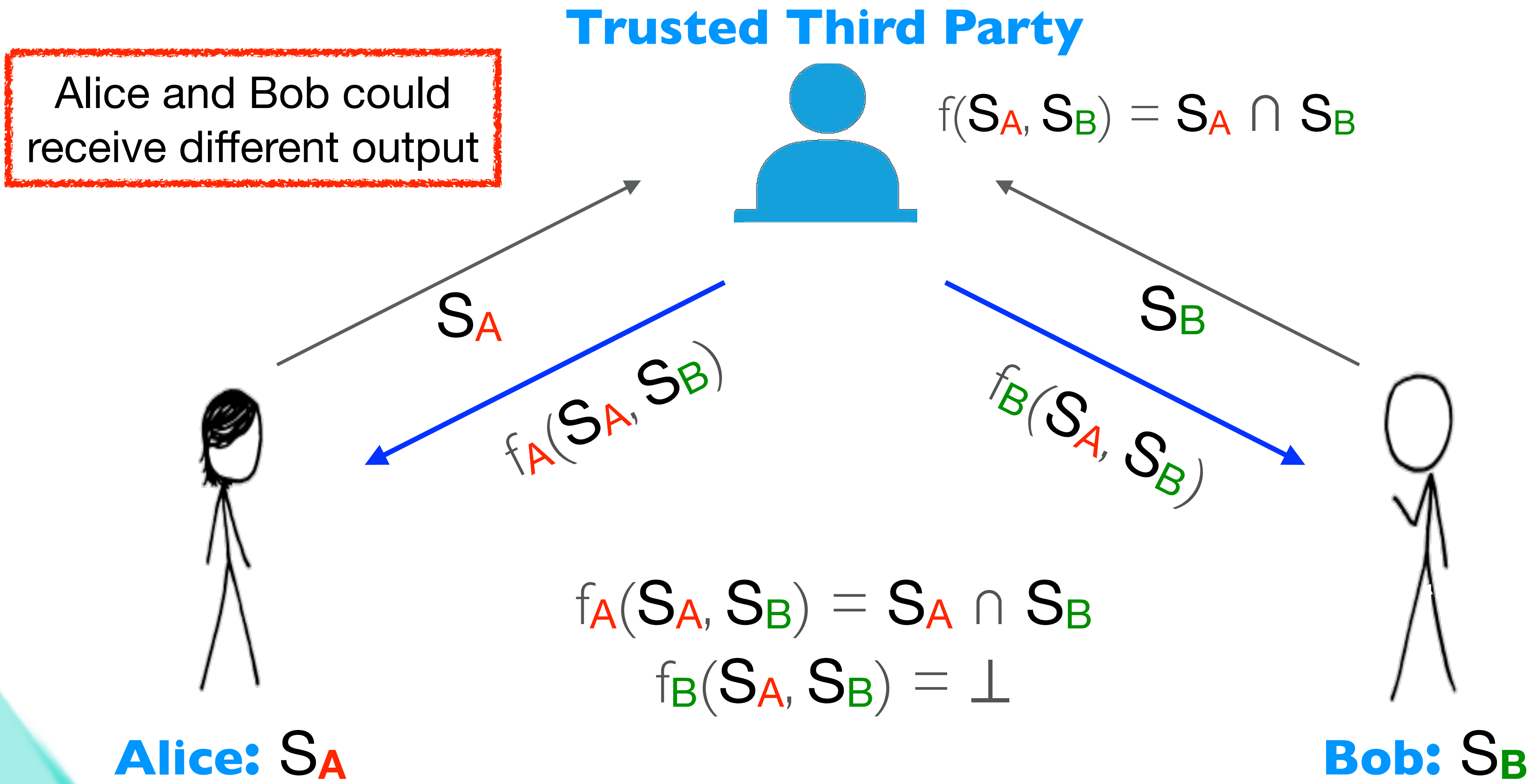
Secure Two-Party Computation Model

Alice and Bob could receive different output



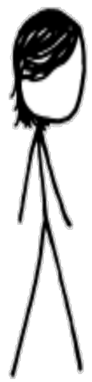
Ideal World

Secure Two-Party Computation Model

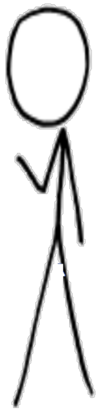


Ideal World

Secure Two-Party Computation Model



Alice: S_A



Bob: S_B

Real World

Secure Two-Party Computation Model

Alice: S_A **Bob:** S_B

Real World

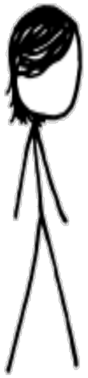
Secure Two-Party Computation Model

Real World

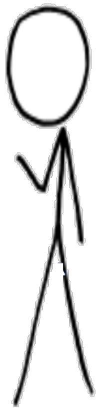
Real World ≈ Ideal World

- **Correctness**
  - Protocol's outputs are identical to the ones obtained in the Ideal World
- **Privacy**
  - Alice's Privacy: Bob learns nothing about Alice input (except its size)
  - Bob's Privacy: Alice learns nothing about Bob's items (except their number) not in the intersection

Private Set Intersection

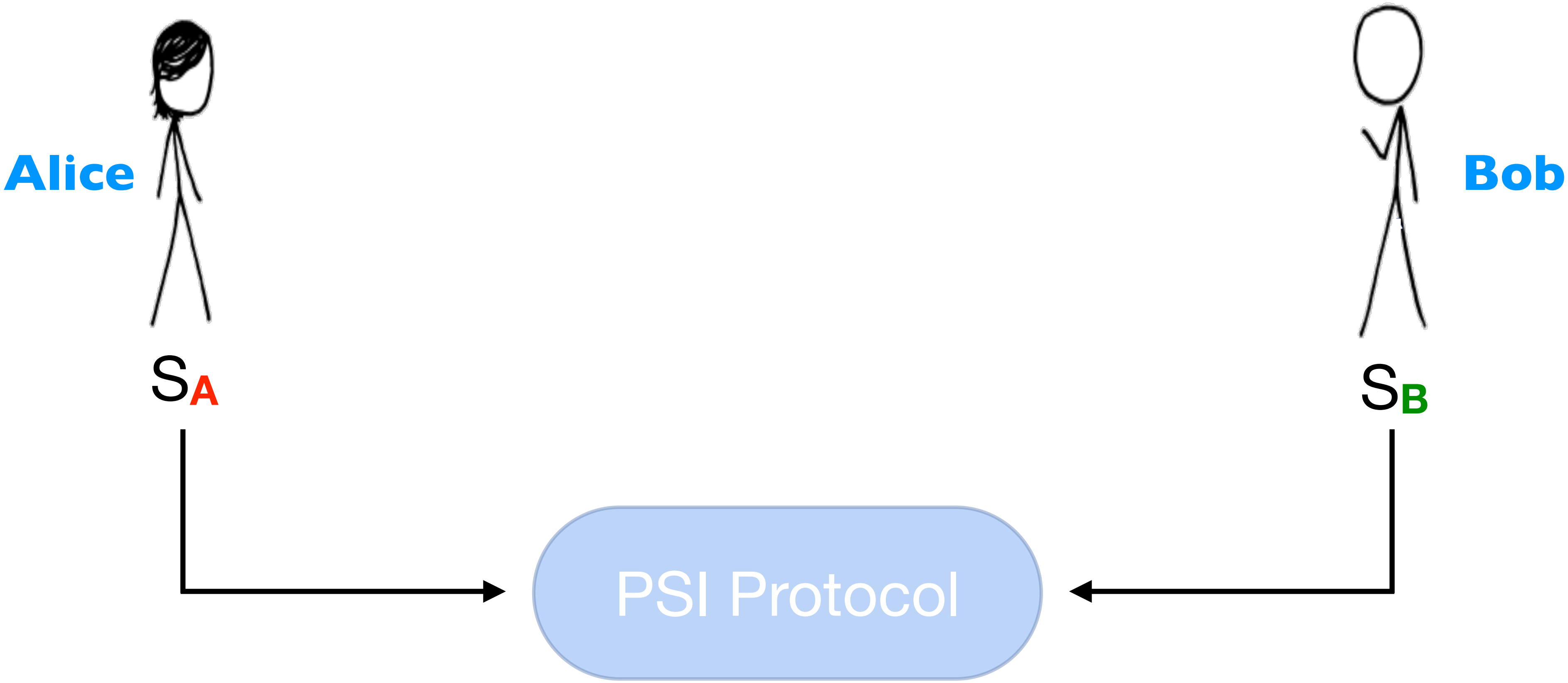
Alice 

S_A

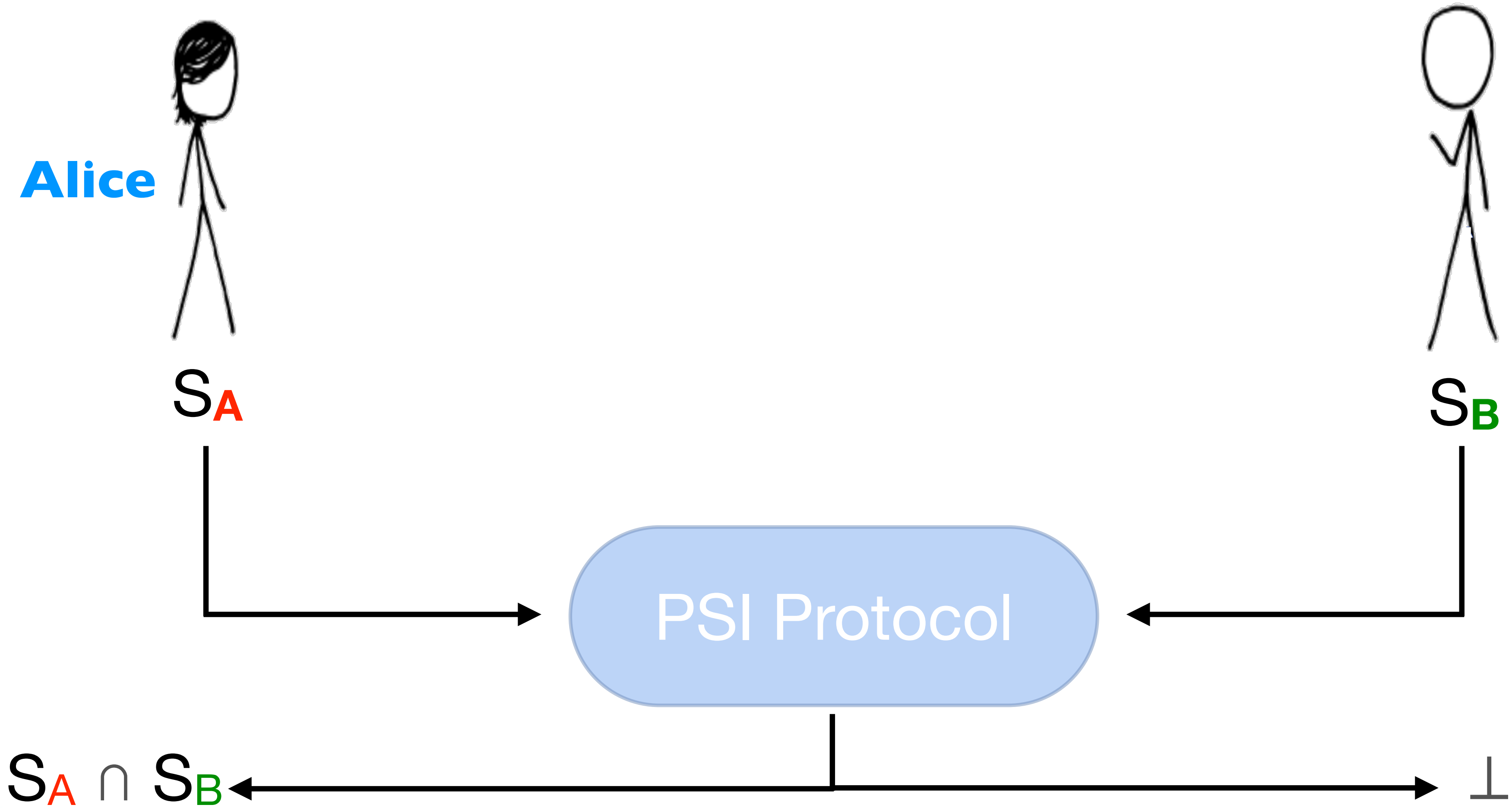
 **Bob**

S_B

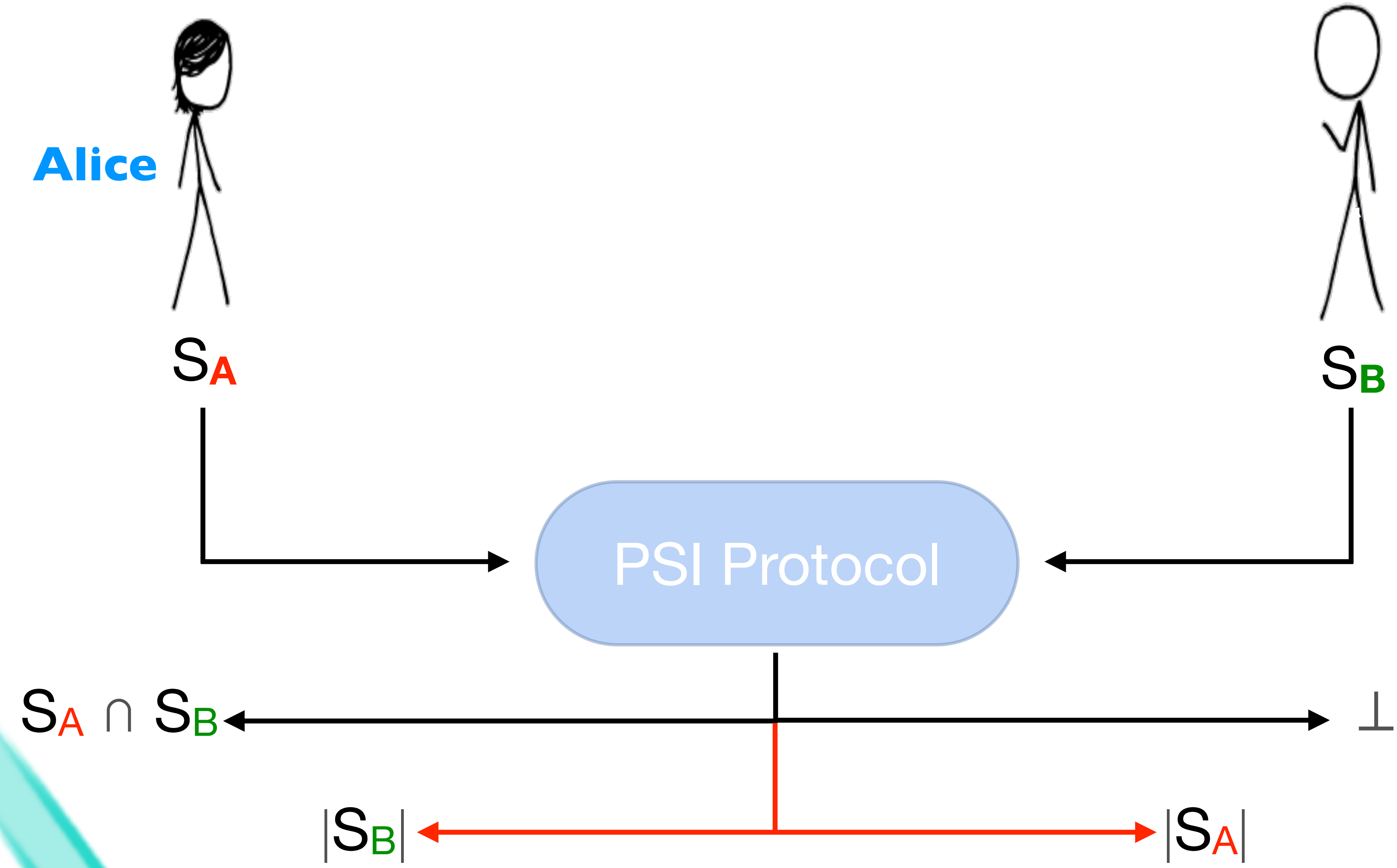
Private Set Intersection



Private Set Intersection



Private Set Intersection

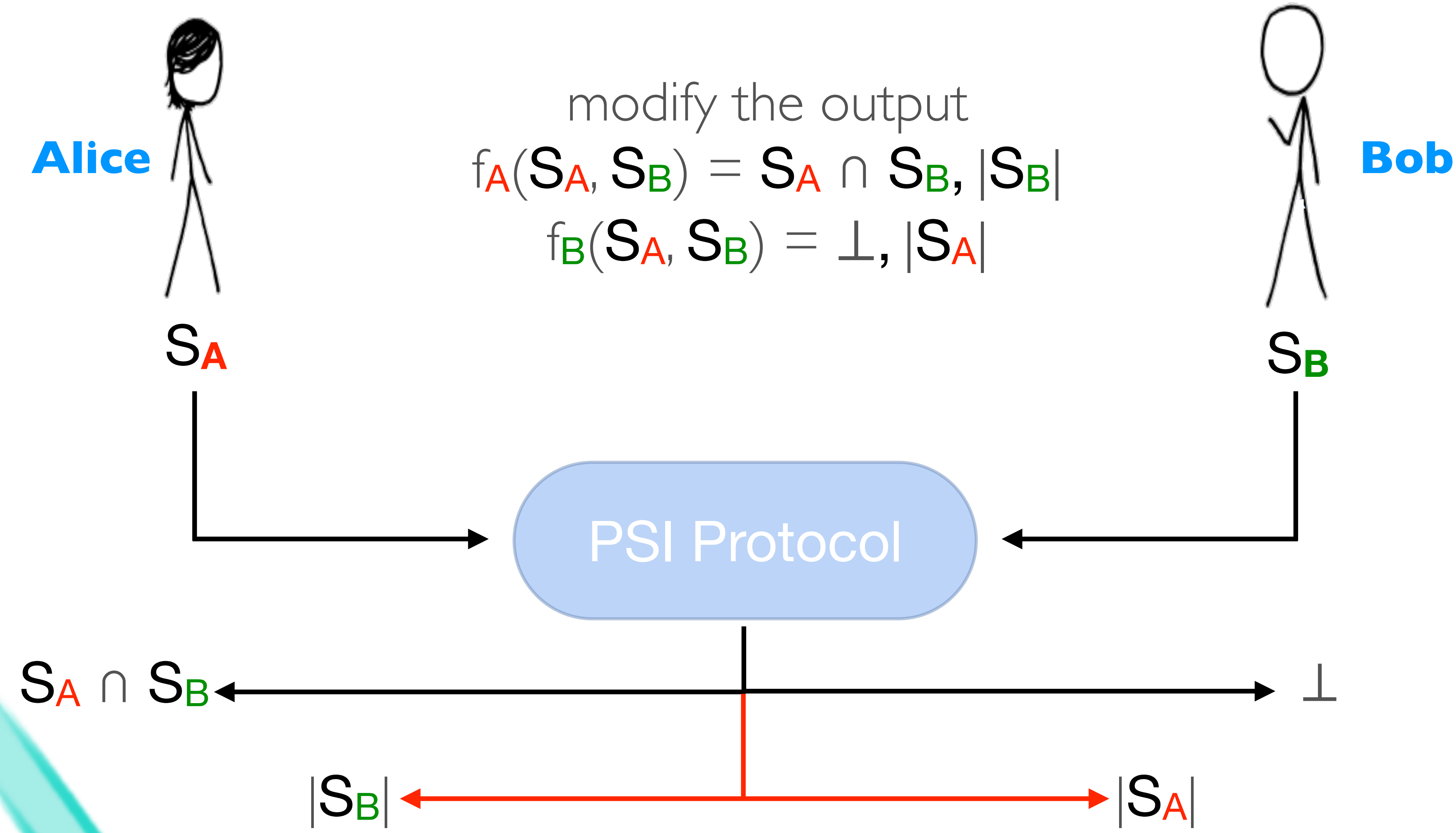


Private Set Intersection

modify the output

$f_A(S_A, S_B) = S_A \cap S_B, |S_B|$

$f_B(S_A, S_B) = \perp, |S_A|$



Type of Adversary

- Honest-but-Curious (a.k.a., semi-honest)
  - faithfully follows protocol specifications
  - does not modify messages/input
  - during or after protocol execution, attempts to infer additional information about the other party's input
- Malicious
  - may deviate from protocol specifications
  - may modify messages/input
  - during or after protocol execution, attempts to infer additional information about the other party's input

A simple insecure protocol

- Alice and Bob must *encode* their values
- Alice and Bob have access to a hash function h
- Bob computes $X_B = \{h(y) : y \in S_B\}$
  - Bob sends X_B to Alice
  - Alice computes the set $\{x \in S_A : h(x) \in X_B\}$
  - This set is equal to $S_A \cap S_B$

A privacy problem

Alice's privacy is preserved

- Alice can easily verify whether a value z is among Bob’s values

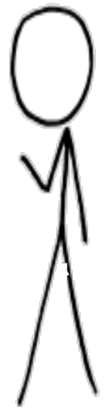
- Alice just check if $h(z) \in X_B$
    - If so, with high probability, $z \in S_A \cap S_B$

A simple secure protocol

- The basic idea is that Alice and Bob jointly encode their values
- The hash function is used to map their values to a set (*multiplicative group*) where some of operations are possible (exponentiation) while others are difficult to compute (discrete logarithm)

A simple secure protocol

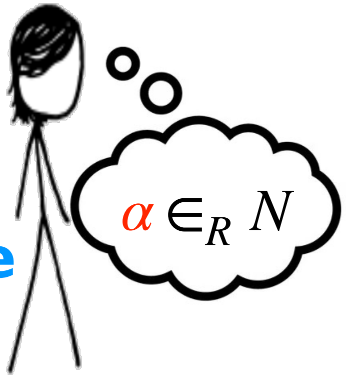
Alice

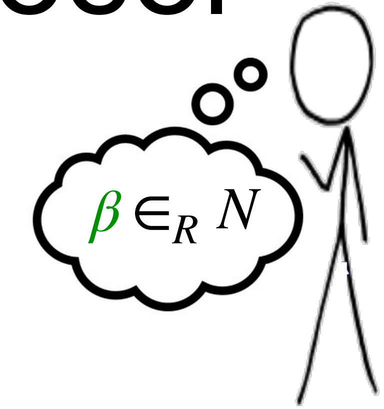


Bob

A simple secure protocol

Alice

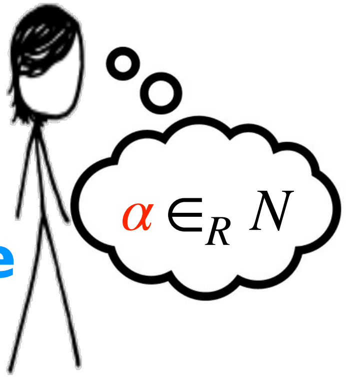




Bob

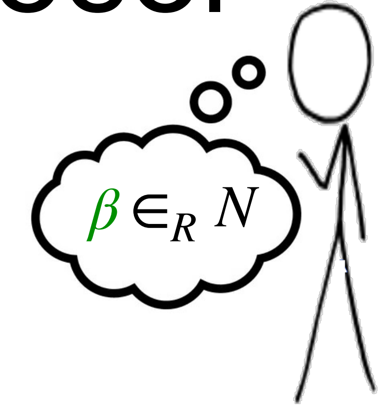
A simple secure protocol

Alice



For $v \in S_A$ and $w \in S_B$,

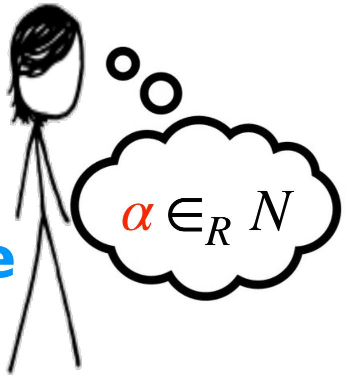
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$



Bob

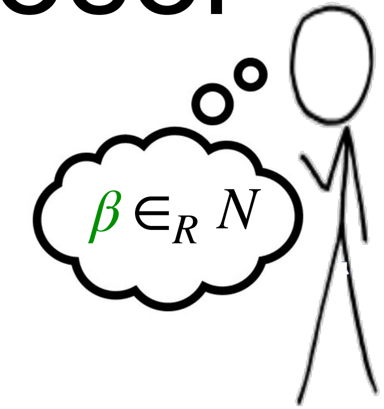
A simple secure protocol

Alice



For $v \in S_A$ and $w \in S_B$,

if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$

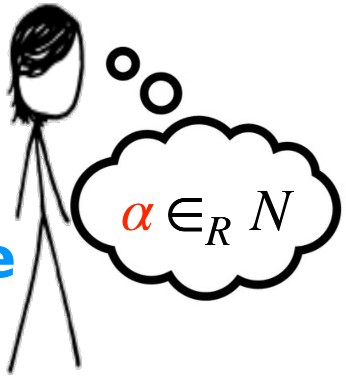


Bob

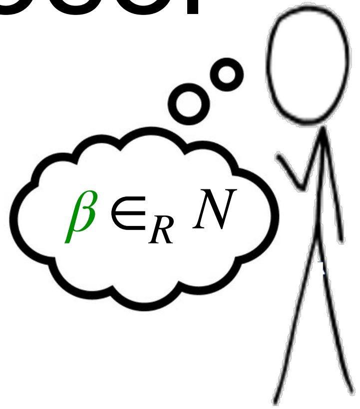
$$E_A = \{(v, h(v)^{\alpha}) : v \in S_A\}$$

A simple secure protocol

Alice



For $v \in S_A$ and $w \in S_B$,
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$

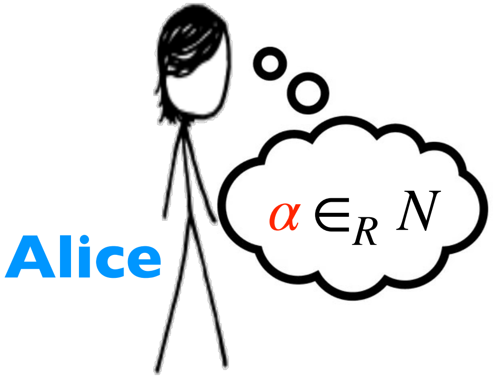


Bob

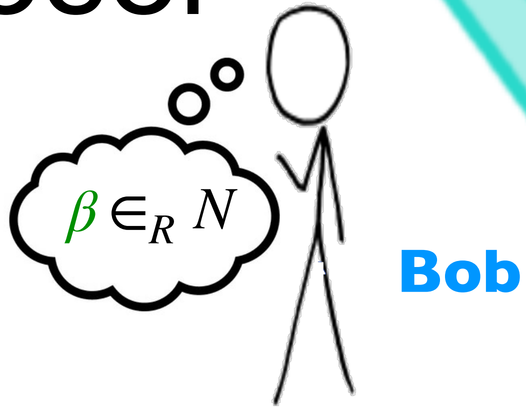
$E_A = \{(v, h(v)^\alpha) : v \in S_A\}$

$$\xrightarrow{S_A^\alpha = \{h(v)^\alpha : v \in S_A\}}$$

A simple secure protocol



For $v \in S_A$ and $w \in S_B$,
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$



$$E_A = \{(v, h(v)^{\alpha}) : v \in S_A\}$$

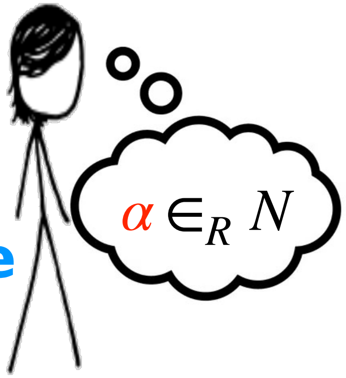
$$\xrightarrow{S_A^{\alpha} = \{h(v)^{\alpha} : v \in S_A\}}$$

$$E_{AB} = \{(h(v)^{\alpha}, (h(v)^{\alpha})^{\beta}) : h(v)^{\alpha} \in S_A^{\alpha}\}$$

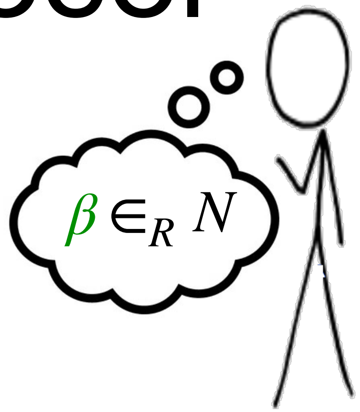
$$S_B^{\beta} = \{h(w)^{\beta} : w \in S_B\}$$

A simple secure protocol

Alice



For $v \in S_A$ and $w \in S_B$,
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$



Bob

$$E_A = \{(v, h(v)^{\alpha}) : v \in S_A\}$$

$$\xrightarrow{S_A^{\alpha} = \{h(v)^{\alpha} : v \in S_A\}}$$

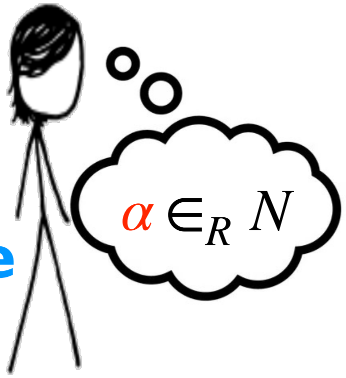
$$E_{AB} = \{(h(v)^{\alpha}, (h(v)^{\alpha})^{\beta}) : h(v)^{\alpha} \in S_A^{\alpha}\}$$

$$S_B^{\beta} = \{h(w)^{\beta} : w \in S_B\}$$

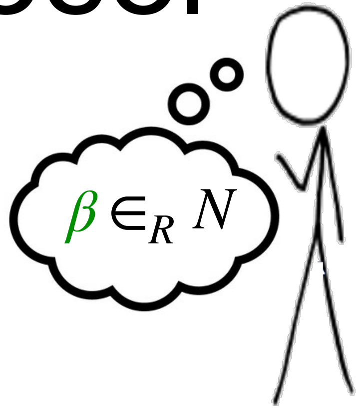
$$\xleftarrow{E_{AB},\ S_B^{\beta}}$$

A simple secure protocol

Alice



For $v \in S_A$ and $w \in S_B$,
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$



Bob

$$E_A = \{(v, h(v)^\alpha) : v \in S_A\}$$

$$\xrightarrow{S_A^\alpha = \{h(v)^\alpha : v \in S_A\}}$$

$$E_{AB} = \{(h(v)^\alpha, (h(v)^\alpha)^\beta) : h(v)^\alpha \in S_A^\alpha\}$$

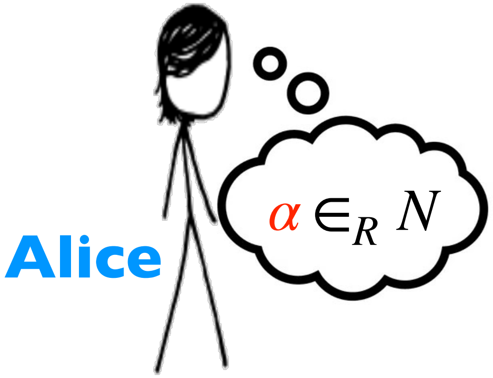
$$S_B^\beta = \{h(w)^\beta : w \in S_B\}$$

$$\xleftarrow{E_{AB},\ S_B^\beta}$$

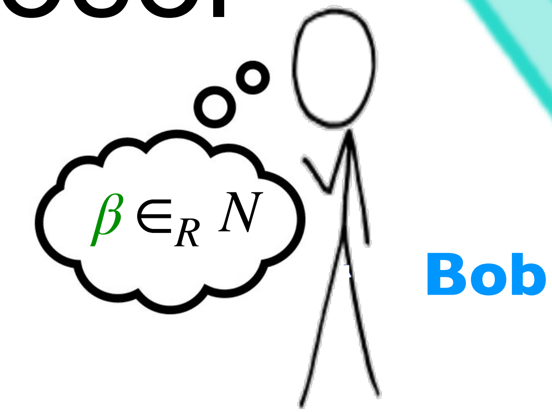
E_A, E_{AB}

$$E_A^\beta = \{(v, h(v)^{\alpha\beta}) : v \in S_A\}$$

A simple secure protocol



For $v \in S_A$ and $w \in S_B$,
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$



$E_A = \{(v, h(v)^{\alpha}) : v \in S_A\}$

$$S_A^{\alpha} = \{h(v)^{\alpha} : v \in S_A\} \longrightarrow$$

$E_{AB} = \{(h(v)^{\alpha}, (h(v)^{\alpha})^{\beta}) : h(v)^{\alpha} \in S_A^{\alpha}\}$

$S_B^{\beta} = \{h(w)^{\beta} : w \in S_B\}$

$$\longleftarrow E_{AB},\ S_B^{\beta}$$

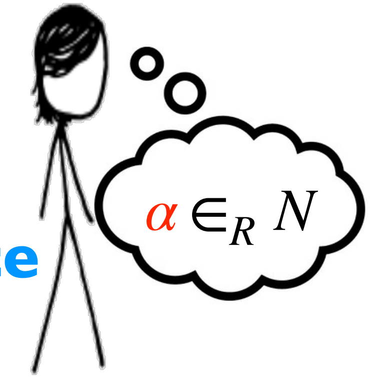
$E_A,\ E_{AB}$

$E_A^{\beta} = \{(v, h(v)^{\alpha\beta}) : v \in S_A\}$

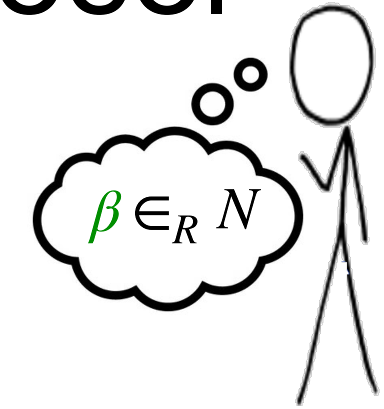
$S_B^{\beta\alpha} = \{h(w)^{\beta\alpha} : h(w)^{\beta} \in S_B^{\beta}\}$

A simple secure protocol

Alice



For $v \in S_A$ and $w \in S_B$,
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$



Bob

$$E_A = \{(v, h(v)^{\alpha}) : v \in S_A\}$$

$$\xrightarrow{S_A^{\alpha} = \{h(v)^{\alpha} : v \in S_A\}}$$

$$E_{AB} = \{(h(v)^{\alpha}, (h(v)^{\alpha})^{\beta}) : h(v)^{\alpha} \in S_A^{\alpha}\}$$

$$S_B^{\beta} = \{h(w)^{\beta} : w \in S_B\}$$

$$\xleftarrow{E_{AB},\ S_B^{\beta}}$$

$E_A,\ E_{AB}$

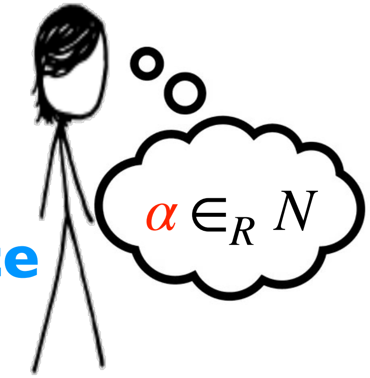
$$E_A^{\beta} = \{(v, h(v)^{\alpha\beta}) : v \in S_A\}$$

$$S_B^{\beta\alpha} = \{h(w)^{\beta\alpha} : h(w)^{\beta} \in S_B^{\beta}\}$$

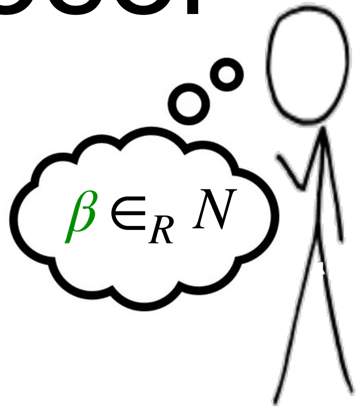
$$S_A \cap S_B = \{v : (v, h(v)^{\alpha\beta}) \in E_A^{\beta} \quad \wedge \quad h(v)^{\alpha\beta} \in S_B^{\beta\alpha}\}$$

A simple secure protocol

Alice



For $v \in S_A$ and $w \in S_B$,
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$



Bob

$$E_A = \{(v, h(v)^{\alpha}) : v \in S_A\}$$

$$\xrightarrow{S_A^{\alpha} = \{h(v)^{\alpha} : v \in S_A\}}$$

$$E_{AB} = \{(h(v)^{\alpha}, (h(v)^{\alpha})^{\beta}) : h(v)^{\alpha} \in S_A^{\alpha}\}$$

$$S_B^{\beta} = \{h(w)^{\beta} : w \in S_B\}$$

$$\xleftarrow{E_{AB},\ S_B^{\beta}}$$

$$E_A,\ E_{AB}$$

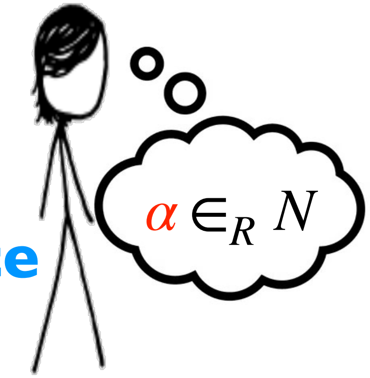
$$E_A^{\beta} = \{(v, h(v)^{\alpha\beta}) : v \in S_A\}$$

$$S_B^{\beta\alpha} = \{h(w)^{\beta\alpha} : h(w)^{\beta} \in S_B^{\beta}\}$$

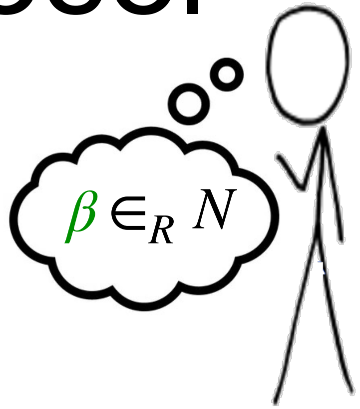
$$S_A \cap S_B = \{v : \underline{(v, h(v)^{\alpha\beta}) \in E_A^{\beta}} \quad \wedge \quad \underline{h(v)^{\alpha\beta} \in S_B^{\beta\alpha}}\}$$

A simple secure protocol

Alice



For $v \in S_A$ and $w \in S_B$,
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$



Bob

$E_A = \{(v, h(v)^{\alpha}) : v \in S_A\}$

$$\xrightarrow{S_A^{\alpha} = \{h(v)^{\alpha} : v \in S_A\}}$$

$E_{AB} = \{(h(v)^{\alpha}, (h(v)^{\alpha})^{\beta}) : h(v)^{\alpha} \in S_A^{\alpha}\}$

$S_B^{\beta} = \{h(w)^{\beta} : w \in S_B\}$

$$\xleftarrow{E_{AB},\ S_B^{\beta}}$$

E_A, E_{AB}

$E_A^{\beta} = \{(v, h(v)^{\alpha\beta}) : v \in S_A\}$

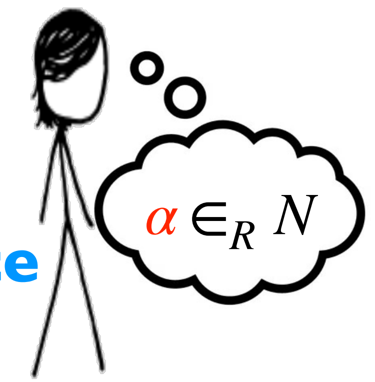
$S_B^{\beta\alpha} = \{h(w)^{\beta\alpha} : h(w)^{\beta} \in S_B^{\beta}\}$

$$S_A \cap S_B = \{v : \underline{(v, h(v)^{\alpha\beta}) \in E_A^{\beta}} \quad \wedge \quad \underline{h(v)^{\alpha\beta} \in S_B^{\beta\alpha}}\}$$

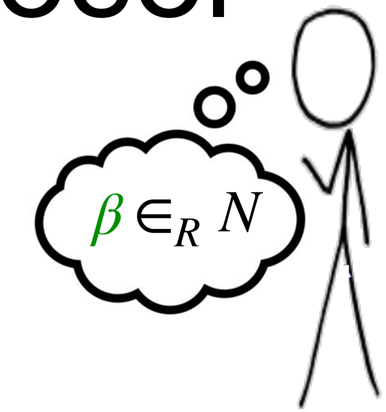
$h(v)^{\alpha\beta} \in S_B^{\beta\alpha} \Rightarrow \exists w \in S_B : h(w)^{\beta\alpha} = h(v)^{\alpha\beta}$

A simple secure protocol

Alice



For $v \in S_A$ and $w \in S_B$,
if $h(v)^{\alpha\beta} = h(w)^{\beta\alpha}$, then $v = w$



Bob

$E_A = \{(v, h(v)^\alpha) : v \in S_A\}$

$$\xrightarrow{S_A^\alpha = \{h(v)^\alpha : v \in S_A\}}$$

$E_{AB} = \{(h(v)^\alpha, (h(v)^\alpha)^\beta) : h(v)^\alpha \in S_A^\alpha\}$

$S_B^\beta = \{h(w)^\beta : w \in S_B\}$

$$\xleftarrow{E_{AB},\ S_B^\beta}$$

$E_A,\ E_{AB}$

$E_A^\beta = \{(v, h(v)^{\alpha\beta}) : v \in S_A\}$

$S_B^{\beta\alpha} = \{h(w)^{\beta\alpha} : h(w)^\beta \in S_B^\beta\}$

For privacy reason
S_A^α and S_B^β are shuffled

$$S_A \cap S_B = \{v : \underline{(v, h(v)^{\alpha\beta}) \in E_A^\beta} \quad \wedge \quad \underline{h(v)^{\alpha\beta} \in S_B^{\beta\alpha}}\}$$

$h(v)^{\alpha\beta} \in S_B^{\beta\alpha} \Rightarrow \exists w \in S_B : h(w)^{\beta\alpha} = h(v)^{\alpha\beta}$

A note on the implementation

- The hash function h maps set elements to group elements $h : U \rightarrow G$ $\quad S_A, S_B \subseteq U$
- The group G could be prime256v1, a NIST elliptic curve group with 256-bit group elements
- Use the *Hashing to Elliptic Curves* algorithms for hashing an arbitrary string to a point on the elliptic curve

draft-irtf-cfrg-hash-to-curve-16 of the Crypto Forum Research Group from the Internet Research Task Force

A polynomial based protocol

- We represent sets by polynomial and use a *partially homomorphic encryption scheme*
  - Public key encryption scheme allowing computation on encrypted data
  - Without knowing sk , given Enc[pk, x], Enc[pk, y], and a constant c, one can compute
    - Enc[pk, x+y]
      - For instance as, Enc[pk, x] · Enc[pk, y]
    - Enc[pk, cx]
      - For instance as, Enc[pk, x]^c

Paillier

ElGamal

Damgard&Jurik

How to represent a set

- To represent a set $S = \{s_1, s_2, \ldots, s_k\}$, we use a degree k polynomial P whose roots are the values in S

How to represent a set

- To represent a set S = {s_1, s_2, …, s_k}, we use a degree k polynomial P whose roots are the values in S

$$P(x) = \prod_{i=1}^{k} (x - s_i) = a_0 + a_1 x + a_2 x + \cdots + a_k x^k$$

How to represent a set

- To represent a set S = {s_1, s_2, …, s_k}, we use a degree k polynomial P whose roots are the values in S

$$P(x) = \prod_{i=1}^{k} (x - s_i) = a_0 + a_1 x + a_2 x + \cdots + a_k x^k$$

$$P(y) \begin{cases} = 0 & \text{if } y \in S \\ \neq 0 & \text{if } y \notin S \end{cases}$$

How to represent a set

- To represent a set S = {s_1, s_2, …, s_k}, we use a degree k polynomial P whose roots are the values in S

$$P(x) = \prod_{i=1}^{k}(x - s_i) = a_0 + a_1 x + a_2 x + \cdots + a_k x^k$$

$$P(y) \begin{cases} = 0 & \text{if } y \in S \\ \neq 0 & \text{if } y \notin S \end{cases}$$

- Given the encryptions of P's coefficients, we can compute, for any value y, the encryption of P(y)

Computing Enc[P(y)]

We omit the public key to simplify the notation

The set S

Encoding of S

Computing Enc[P(y)]

We omit the public key to simplify the notation

The set S $$P(x) \quad = \quad a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$$

Encoding of S

Computing Enc[P(y)]

We omit the public key to simplify the notation

The set S $$P(x) \quad = \quad a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$$

Encoding of S $$Enc[a_0],\ Enc[a_1],\ Enc[a_2],\ \cdots,\ Enc[a_k]$$

Computing Enc[P(y)]

We omit the public key to simplify the notation

The set S $$P(x) \quad = \quad a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$$

Encoding of S $$Enc[a_0],\ Enc[a_1],\ Enc[a_2],\ \cdots,\ Enc[a_k]$$

Knowing y, compute Enc[P(y)]

Computing Enc[P(y)]

We omit the public key to simplify the notation

The set S $$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k$$

Encoding of S $$Enc[a_0],\ Enc[a_1],\ Enc[a_2],\ \cdots,\ Enc[a_k]$$

Knowing y, compute Enc[P(y)]

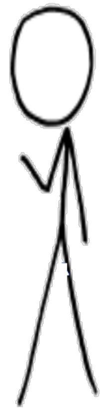
$$\begin{aligned} Enc[P(y)] &= Enc[a_0 + a_1 y + a_2 y^2 + \cdots + a_k y^k] \\ &= Enc[a_0] \cdot Enc[a_1 y] \cdot Enc[a_2 y^2] \cdots Enc[a_k y^k] \\ &= Enc[a_0] \cdot Enc[a_1]^y \cdot Enc[a_2]^{y^2} \cdots Enc[a_k]^{y^k} \end{aligned}$$

A new protocol

Alice

pk: Alice's
public key

Encryptions are under
Alice's public key



Bob

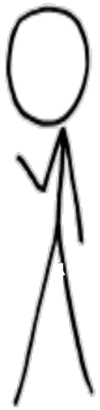
A new protocol

Alice

pk: Alice's
public key

$$P(x) = \prod_{v \in S_A} (x - v) = \sum_{i=0}^{k} a_i x^i$$

Encryptions are under
Alice's public key



Bob

A new protocol

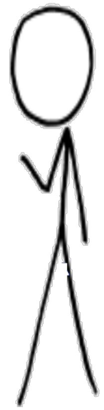
Alice

pk: Alice's public key

$$P(x) = \prod_{v \in S_A} (x - v) = \sum_{i=0}^{k} a_i x^i$$

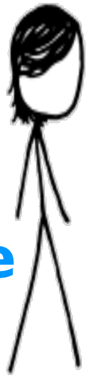
Encryptions are under Alice's public key

$$\xrightarrow{Enc[a_0], \ldots, Enc[a_k]}$$



Bob

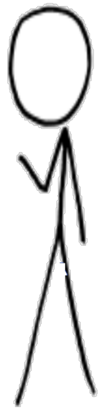
A new protocol



Alice

pk: Alice's public key

Encryptions are under Alice's public key



Bob

$$P(x) = \prod_{v \in S_A} (x - v)$$
$$= \sum_{i=0}^{k} a_i x^i$$

$$\xrightarrow{Enc[a_0], \ldots, Enc[a_k]}$$

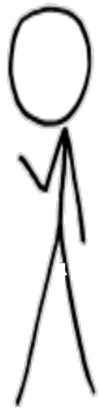
$$S_B^{pk} = \{Enc[rP(w) + w] : w \in S_B\}$$

A new protocol

Alice

pk: Alice's public key

Encryptions are under Alice's public key



Bob

$$P(x) = \prod_{v \in S_A} (x - v)$$

$$= \sum_{i=0}^{k} a_i x^i$$

$$\xrightarrow{Enc[a_0], \ldots, Enc[a_k]}$$

$$S_B^{pk} = \{Enc[rP(w) + w] : w \in S_B\}$$

$$Enc[rP(w) + w] =$$

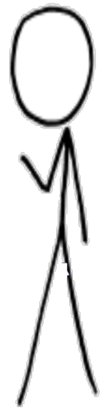
$$Enc[P(w)]^r \cdot Enc[w]$$

A new protocol

Alice

pk: Alice's
public key

Encryptions are under
Alice's public key



Bob

$$P(x) = \prod_{v \in S_A} (x - v) = \sum_{i=0}^{k} a_i x^i$$

$$Enc[a_0], \ldots, Enc[a_k] \longrightarrow$$

$$S_B^{pk} = \{Enc[rP(w) + w] : w \in S_B\}$$

$$\longleftarrow \text{permuted } S_B^{pk}$$

$$Enc[rP(w) + w] = Enc[P(w)]^r \cdot Enc[w]$$

A new protocol

Alice

pk: Alice's
public key

Encryptions are under
Alice's public key

Bob

$$P(x) = \prod_{v \in S_A} (x - v) = \sum_{i=0}^{k} a_i x^i$$

$$Enc[a_0], \ldots, Enc[a_k] \longrightarrow$$

$$S_B^{pk} = \{Enc[rP(w) + w] : w \in S_B\}$$

$$Enc[rP(w) + w] = Enc[P(w)]^r \cdot Enc[w]$$

permuted S_B^{pk} $\longleftarrow$

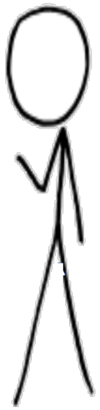
$$S_A \cap S_B = S_A \cap Dec[S_B^{pk}]$$

A new protocol

pk: Alice's public key

Alice

Encryptions are under Alice's public key



Bob

$$P(x) = \prod_{v \in S_A} (x - v)$$

$$= \sum_{i=0}^{k} a_i x^i$$

$$Enc[a_0], \ldots, Enc[a_k] \longrightarrow$$

$$S_B^{pk} = \{Enc[rP(w) + w] : w \in S_B\}$$

$$Enc[rP(w) + w] =$$

$$Enc[P(w)]^r \cdot Enc[w]$$

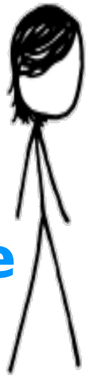
permuted S_B^{pk} $\longleftarrow$

$$S_A \cap S_B = S_A \cap Dec[S_B^{pk}]$$

recall

$$P(y) \begin{cases} = 0 & \text{if } y \in S \\ \neq 0 & \text{if } y \notin S \end{cases}$$

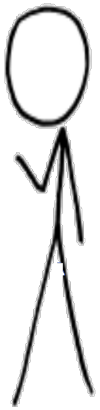
A new protocol



Alice

pk: Alice's public key

Encryptions are under Alice's public key



Bob

$$P(x) = \prod_{v \in S_A} (x - v) = \sum_{i=0}^{k} a_i x^i$$

$$\xrightarrow{Enc[a_0], \ldots, Enc[a_k]}$$

$$S_B^{pk} = \{Enc[rP(w) + w] : w \in S_B\}$$

$$Enc[rP(w) + w] = Enc[P(w)]^r \cdot Enc[w]$$

$$\xleftarrow{\text{permuted } S_B^{pk}}$$

$$S_A \cap S_B = S_A \cap Dec[S_B^{pk}]$$

recall

$$P(y) \begin{cases} = 0 & \text{if } y \in S \\ \neq 0 & \text{if } y \notin S \end{cases}$$

$$Enc[rP(w) + w] = \begin{cases} Enc(w) & \text{if } w \in S_A \cap S_B \\ Enc(r') & \text{if } w \notin S_A \cap S_B \end{cases}$$

Why polynomials?

- We can compute more complex functions
- Add a payload to the elements in the intersection
- Bob, instead of computing $Enc(rP(w) + w)$ computes $Enc(rP(w) + w || \texttt{payload(w)})$
  - `payload(w)` represents *some information* associated to w

PSI Variants

- PET (Private Equality Test)
  - Alice will get **True** iff $S_A = S_B$, where $|S_A| = |S_B| = 1$
- PSI-CA (PSI CArdinality)
  - Alice will compute $|S_A \cap S_B|$
- PSI-CA-T (PSI-CA Threshold)
  - Alice will compute **True** if $|S_A \cap S_B| \geqq t$
- Private Intersection Sum with Cardinality (PSI-Sum)
  - Alice will compute $|S_A \cap S_B|$ and $\sum_{w \in S_A \cap S_B} t_w$

$S_A = \{v\}$ users' ids seeing brand B's ads on A's site

$S_B = \{(w, t_w)\}$ (users' ids, € spent at B's store)

PSI Enhancement

- Malicious PSI
  - Players can cheat and follow arbitrary path in the protocol
- Hiding the size of the set(s)
  - Alice (Bob) does not know the size of Bob’s (Alice’s) set
- Authorized PSI
  - Elements the sets in must be signed by a third party (think of them as patients records certified by a CA)

Application: Paternity test

in silico

- Alice and Bob compute set elements based on their DNA strand (a string over the alphabet {A, T, C, G})
- Tools:
  - Enzymes $\{e_1, \ldots., e_k\}$ breaks down in fragments the DNA strand
  - Markers $\{m_1, \ldots, m_p\}$ select p fragments of the DNA fragments
- $S_A = \{(\,|\,frag_i^A\,|, m_i) : 1 \le i \le p\}$
- $S_B = \{(\,|\,frag_i^B\,|, m_i) : 1 \le i \le p\}$

If $|\,S_A \cap S_B\,| \ge \tau$,
Alice and Bob are relative

Application: Plagiarism Detection

CTRL+C CTRL+V

- Jaccard(S_A, S_B) = $|S_A \cap S_B| / |S_A \cup S_B|$

Set Similarity $= |S_A \cap S_B| / (|S_A| + |S_B| - |S_A \cap S_B|)$

- Text Similarity: represent text by trigrams
  compute Jaccard Index

the quick brown fox jumps over the lazy dog

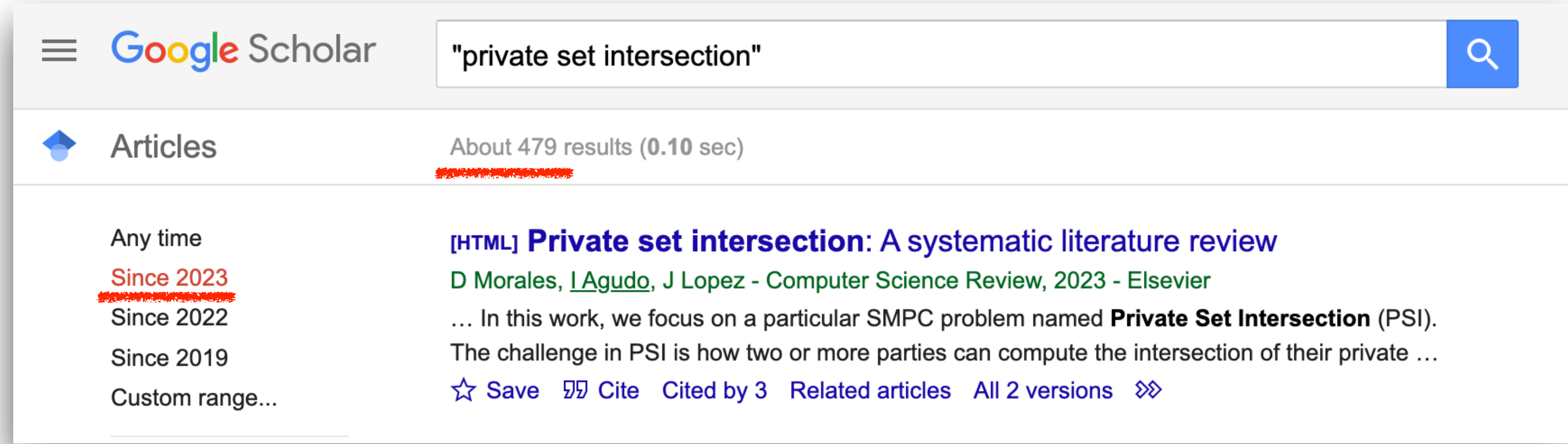
azy, bro, ckb, dog, ela, equ, ert, fox, hel, heq, ick, jum,
kbr, laz, mps, nfo, ove, own, oxj, pso, qui, row, rth, sov,
the, uic, ump, ver, wnf, xju, ydo, zyd

Other applications

- Other Genetic Tests
- Personalized Advertisement
- Biometric Authentication
- Find matching in Social Networks

$\left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\}$ privacy enhanced

Google Scholar Search



De Componendis Cifris

https://www.decifris.it