RETHINKING ALGORITHMS FOR SECURE COMPUTATION *A Greedy Approach*

Muthu Venkitasubramaniam

(joint work with abhi shelat)

Secure Computation

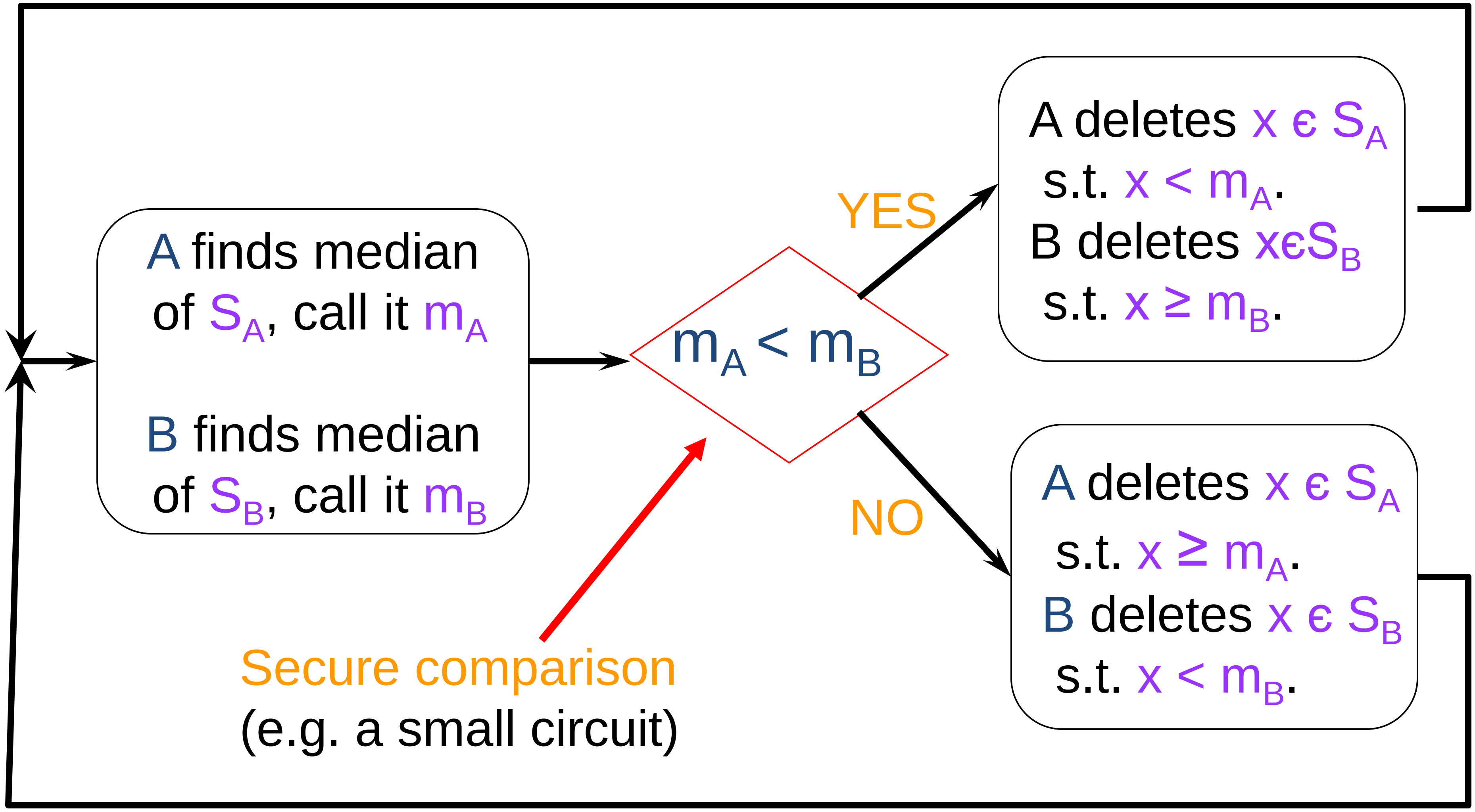
- [STEP 1] Compile f to
  - Boolean Circuits, Arithmetic Circuits, ORAM
- [STEP 2] Generic Approaches
  - Yao based, GMW based, Information Theoretic based
- How to determine which approach
  - Depends on size, latency, bandwidth, etc
- Sometimes specific approaches are better
  - PSI (great primitive, several applications)

THIS TALK

A New Algorithmic Approach for Designing Secure Computation Protocols

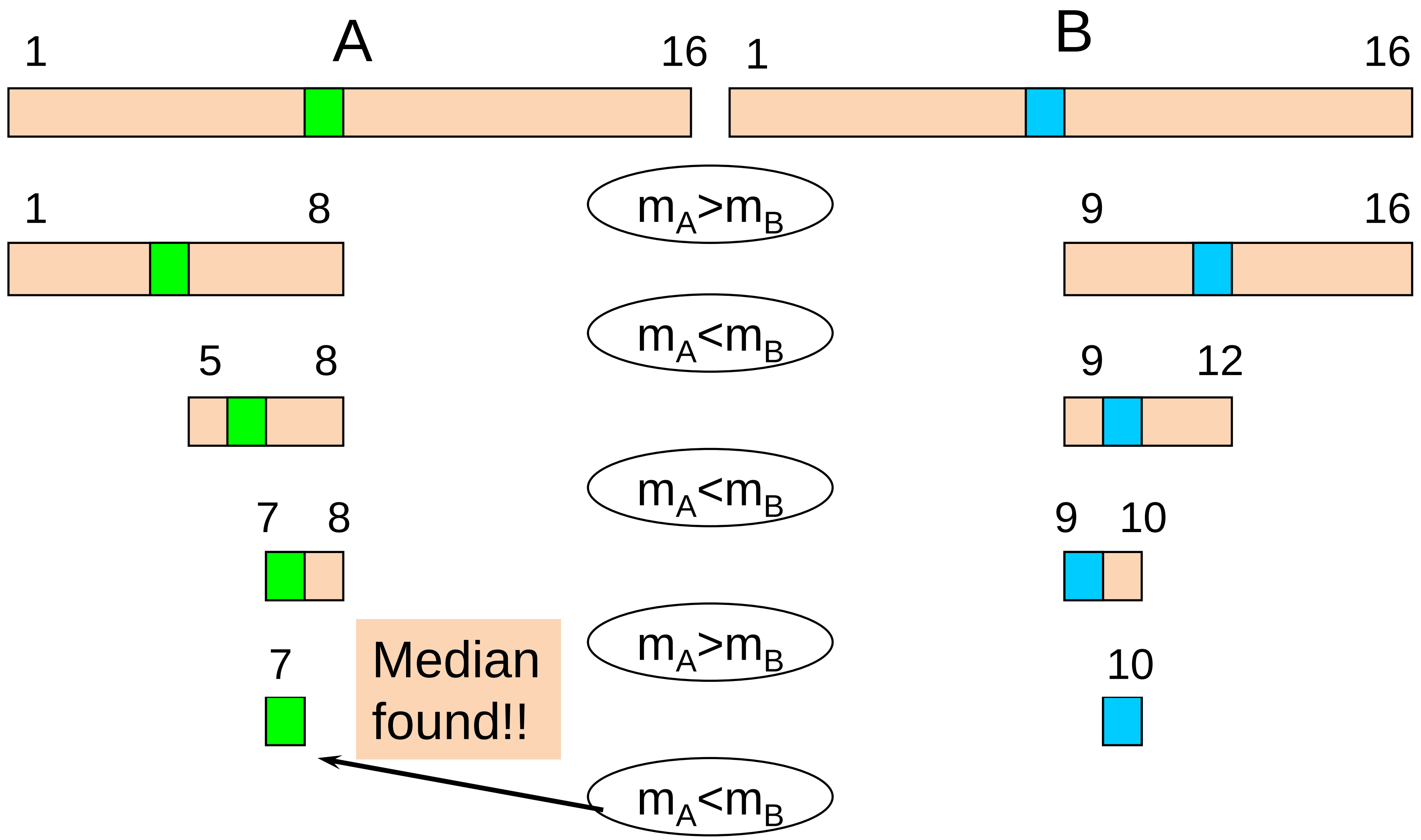
SECURE COMPUTATION OF MEDIAN

Aggarwal, Mishra, Pinkas (Eurocrypt `04, JOC `10)


A finds median of S_A, call it m_A
B finds median of S_B, call it m_B
$m_A < m_B$
YES
NO
A deletes $x \in S_A$ s.t. $x < m_A$.
B deletes $x \in S_B$ s.t. $x \geq m_B$.
A deletes $x \in S_A$ s.t. $x \geq m_A$.
B deletes $x \in S_B$ s.t. $x < m_B$.
Secure comparison
(e.g. a small circuit)

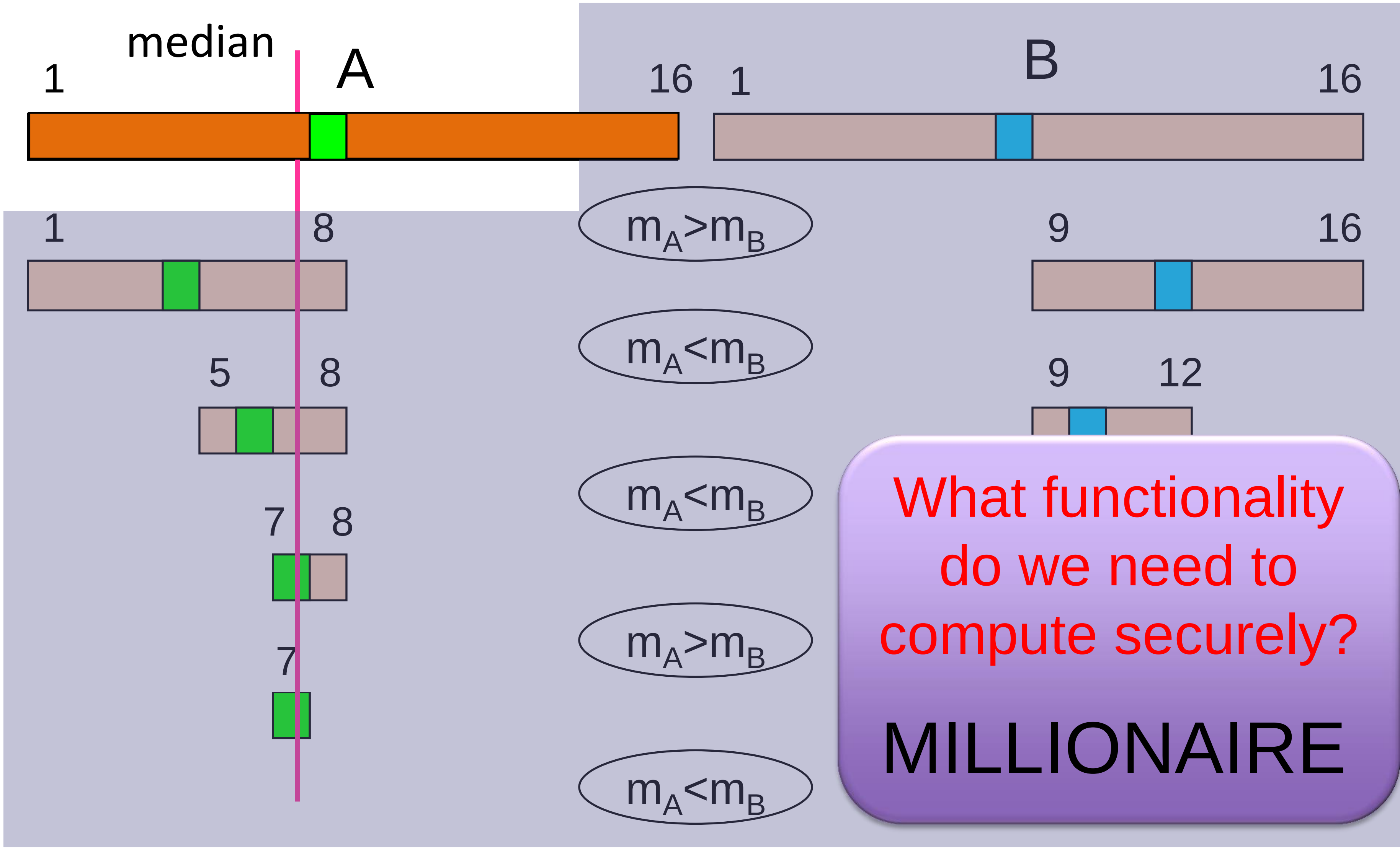
Slides borrowed from Benny Pinkas

WALK THROUGH



Slides borrowed from Benny Pinkas

PROVING Semi-honest SECURITY



What functionality do we need to compute securely?

MILLIONAIRE

Slides borrowed from Benny Pinkas

WHAT ELSE CAN WE COMPUTE USING MILLIONAIRE?

- Convex Hull
- Minimum Spanning Tree [BS05]
- Unit Job Scheduling
- Single Source All Destination Shortest Paths [BS05]
- Set Cover / Vertex Cover / Max Cover*

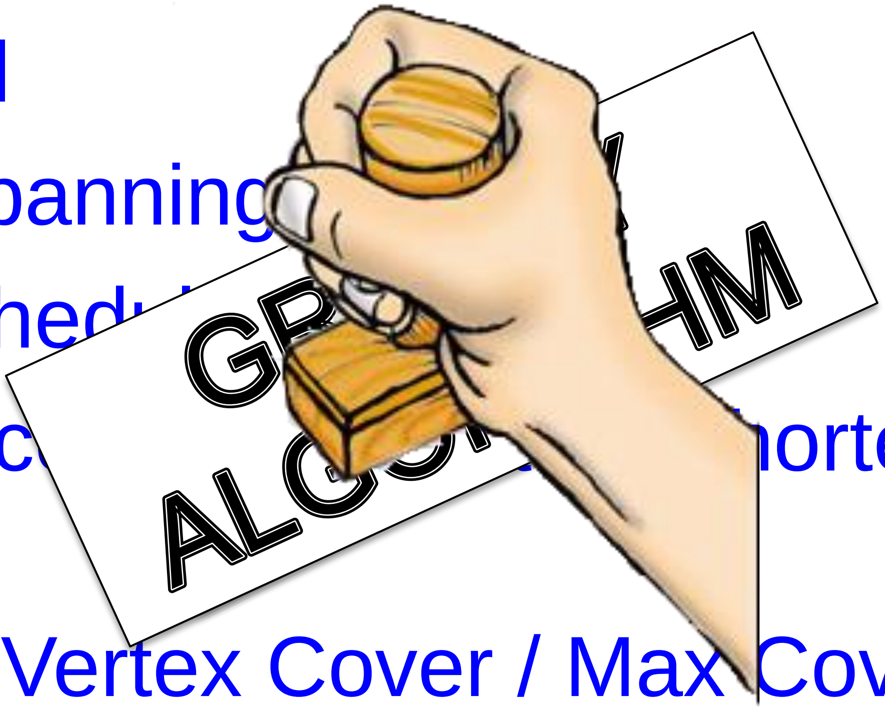
RESULTS (communication complexity)

Algorithm	Our Work (O)	Circuit (Ω)	ORAM (Ω)
Convex Hull	$O\ell$	$I\ \mathit{log}(I)\ell$	$I\ \mathit{log}^3(I)\ell$
MST	$V\ell$	$(V\alpha(V))^2\ell$	$V\alpha(V)\ \mathit{log}^3(V)\ell$
Unit Job Scheduling	$O\ell$	$I^2\ell$	$I\ \mathit{log}^3(I)\ell$
Single Src ADSP	$V\ell$	$E^2\ell$	$E\ \mathit{log}^3(E)\ell$
Cover Problems	$O\ell$	$I_S{}^2\ell$	$I_S\ \mathit{log}^3(I_S)\ell$

I	- input size	$\alpha()$	- Inverse Ackerman fn.
O	- output size	V	- #Vertices
ℓ	- integer representation	E	- #Edges

WHAT PARADIGM ABSTRACTS THESE ALGORITHMS?

- Convex Hull
- Minimum Spanning
- Unit Job Sched
- Single Sourc ortest Paths
- Set Cover / Vertex Cover / Max Cover*



Greedy Algorithms

- Iteratively find the (local) optimal choice and hope for the best
- Leads to optimal in many problems
  - Convex Hull: Jarvis March
  - MST: Kruskal, Prim’s algorithm
  - Job Scheduling (many variants)
  - Shortest Path: Dijkstra
  - Set Cover: Submodular Function Approximation

Our Greedy-Millionaire Framework

A function f is *secure greedy compatible* if there exists a function F such that:

1. UNIQUE SOLUTION

   Given inputs U and V of Alice and Bob $f(U,V)$ is unique

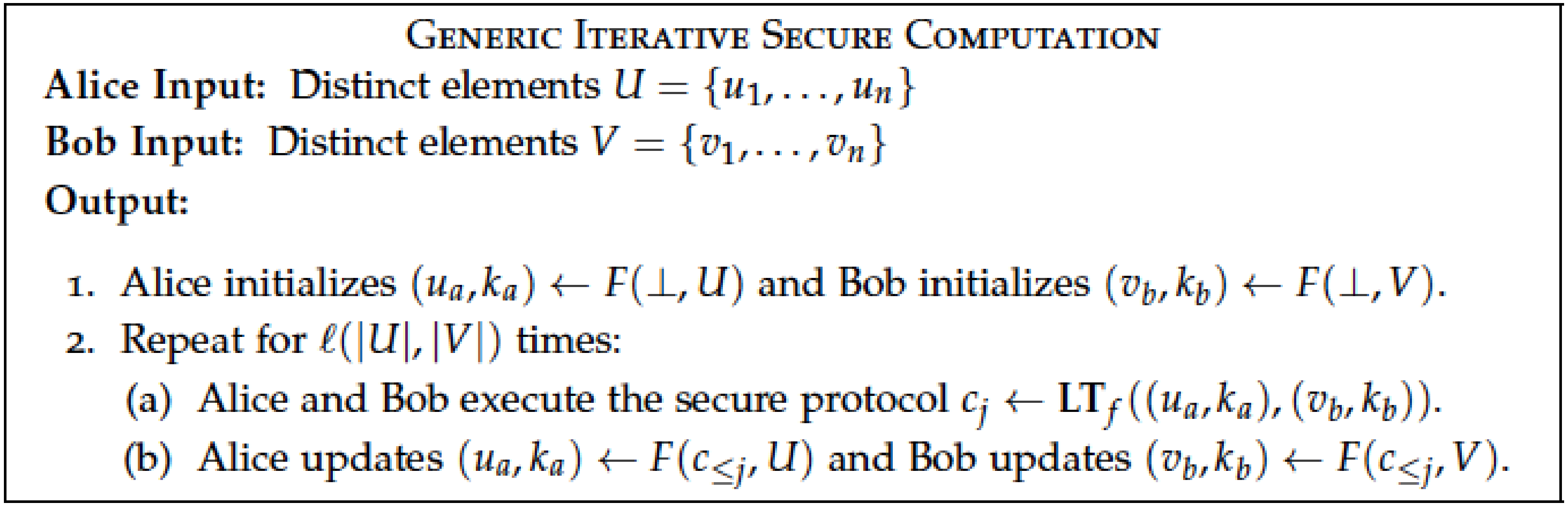
2. UNIQUE ORDER – If $f(U,V)=\left(c_1,\ldots,c_l\right)$, then

$$F(\wedge, U \text{È} V) = c_1 \text{ and } F(c_{\text{£}i}, U \text{È} V) = c_{i+1}$$

3. LOCAL UPDATABILITY

$$F(c_{\text{£}i}, U \text{È} V) = LT\left(F(c_{\text{£}i}, U), F(c_{\text{£}i}, V)\right)$$

Secure Greedy-Millionaire Algorithm



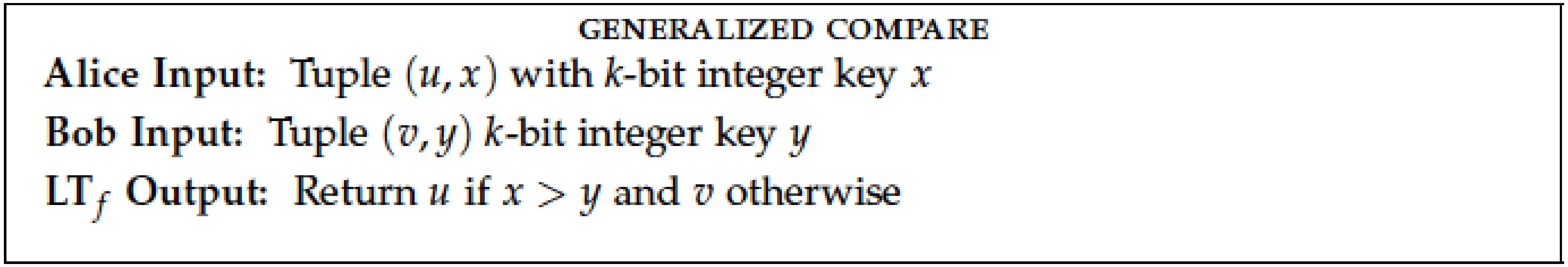
GENERIC ITERATIVE SECURE COMPUTATION

Alice Input: Distinct elements $U = \{u_1, \ldots, u_n\}$

Bob Input: Distinct elements $V = \{v_1, \ldots, v_n\}$

Output:

1. Alice initializes $(u_a, k_a) \leftarrow F(\perp, U)$ and Bob initializes $(v_b, k_b) \leftarrow F(\perp, V)$.
2. Repeat for $\ell(|U|, |V|)$ times:
   (a) Alice and Bob execute the secure protocol $c_j \leftarrow \mathrm{LT}_f((u_a, k_a), (v_b, k_b))$.
   (b) Alice updates $(u_a, k_a) \leftarrow F(c_{\leq j}, U)$ and Bob updates $(v_b, k_b) \leftarrow F(c_{\leq j}, V)$.



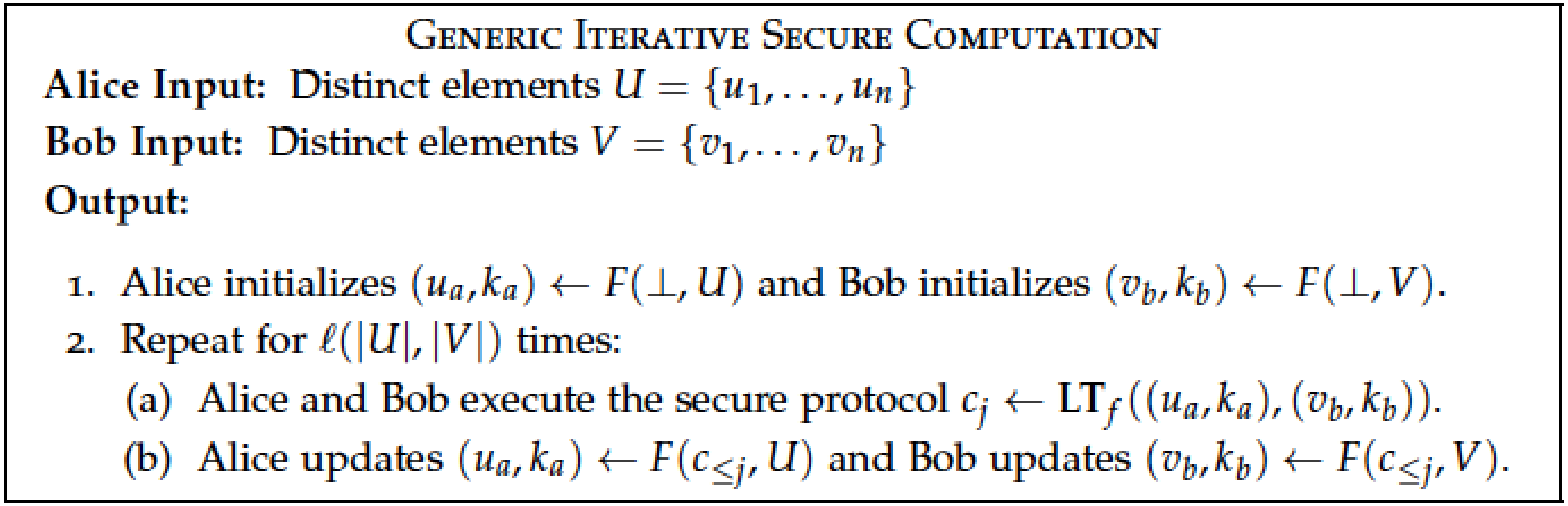
GENERALIZED COMPARE

Alice Input: Tuple (u, x) with k-bit integer key x

Bob Input: Tuple (v, y) k-bit integer key y

LT_f Output: Return u if $x > y$ and v otherwise

Secure Greedy-Millionaire Algorithm



GENERIC ITERATIVE SECURE COMPUTATION

Alice Input: Distinct elements $U = \{u_1, \ldots, u_n\}$

Bob Input: Distinct elements $V = \{v_1, \ldots, v_n\}$

Output:

```
1. Alice initializes $(u_a, k_a) \leftarrow F(\perp, U)$ and Bob initializes $(v_b, k_b) \leftarrow F(\perp, V)$.
2. Repeat for $\ell(|U|, |V|)$ times:
    (a) Alice and Bob execute the secure protocol $c_j \leftarrow \mathrm{LT}_f((u_a, k_a), (v_b, k_b))$.
    (b) Alice updates $(u_a, k_a) \leftarrow F(c_{\le j}, U)$ and Bob updates $(v_b, k_b) \leftarrow F(c_{\le j}, V)$.
```

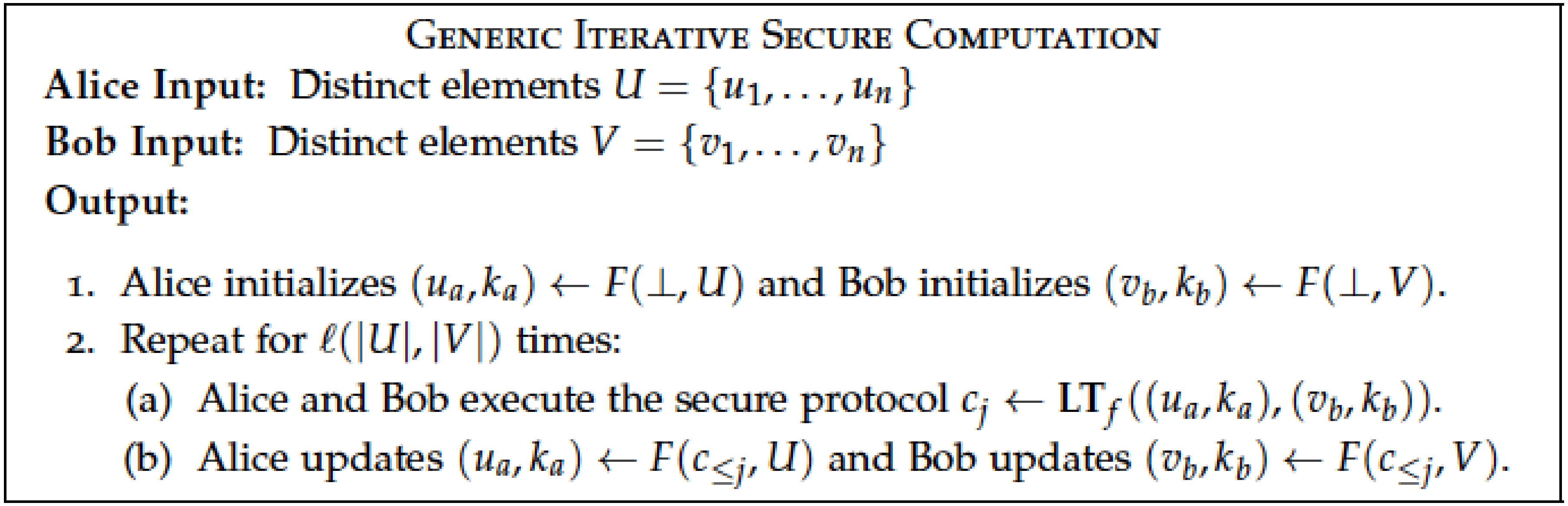
CORRECTNESS:

$$f(U,V) = \left(c_1, \ldots, c_l\right)$$

$$F(\perp, U \cup V) = c_1 \text{ and } F(c_{\le i}, U \cup V) = c_{i+1}$$

$$F(c_{\le i}, U \cup V) = LT\left(F(c_{\le i}, U), F(c_{\le i}, V)\right)$$

Secure Greedy-Millionaire Algorithm



GENERIC ITERATIVE SECURE COMPUTATION

Alice Input: Distinct elements $U = \{u_1, \ldots, u_n\}$

Bob Input: Distinct elements $V = \{v_1, \ldots, v_n\}$

Output:

```
1. Alice initializes $(u_a, k_a) \leftarrow F(\perp, U)$ and Bob initializes $(v_b, k_b) \leftarrow F(\perp, V)$.
2. Repeat for $\ell(|U|, |V|)$ times:
    (a) Alice and Bob execute the secure protocol $c_j \leftarrow \mathrm{LT}_f((u_a, k_a), (v_b, k_b))$.
    (b) Alice updates $(u_a, k_a) \leftarrow F(c_{\leq j}, U)$ and Bob updates $(v_b, k_b) \leftarrow F(c_{\leq j}, V)$.
```

SIMULATION:

Input U and Output $(c_1, \ldots, c_l)$

Unique Solution and Unique Order

- output of iteration i is c_i

Matroid Set Systems

A set system (S,I) where S is a finite set, and I a nonempty family of subsets of S is a matroid if

Hereditary Property:

If $B \in I$ and $A \subseteq B$, then $A \in I$.

Exchange Property:

If $A,B \in I$ and $|A| < |B|$, then

there exists x in $B \setminus A$ such that $A \cup \{x\}$ is in I

Weighted Matroid: a weight function $w : S \rightarrow R^+$

THEOREM: The greedy algorithm finds maximal independent set with minimum cost.

Examples of Matroids

Example 1: Let M be a matrix.
Let S be the set of rows of M and
$I = \{ A \mid A \subseteq S, A \text{ is linearly independent} \}$

Example 2: Let G = (V,E) be an undirected graph. Choose S = E and
I = { A | H = (V,A) is an induced subgraph of G such that H is a forest }

Greedy Algorithm for Matroids

```
Greedy ALGORITHM ((S,I),w)
1. Set A to be empty
2. For each x in S taken in monotonically decreasing order do
   – If A∪{x} in I then set A = A∪{x}
3. Return A
```

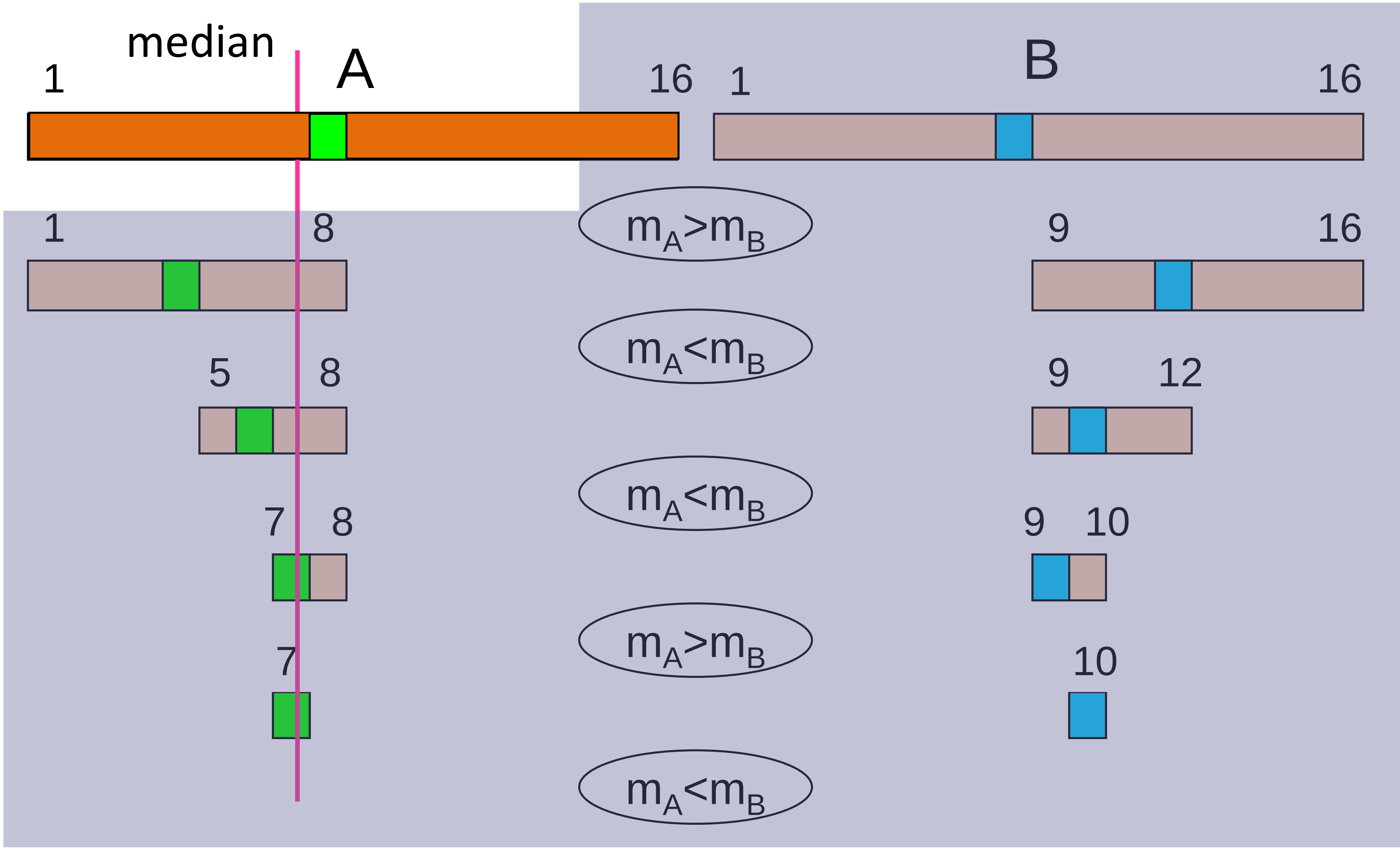
Matroids are secure-greedy-compatible if

- UNIQUE SOLUTION and UNIQUE ORDER: Assume weights are distinct
- LOCAL UPDATABILITY: If membership in I can be done locally

CAN WE ACHIEVE MALICIOUS SECURITY?

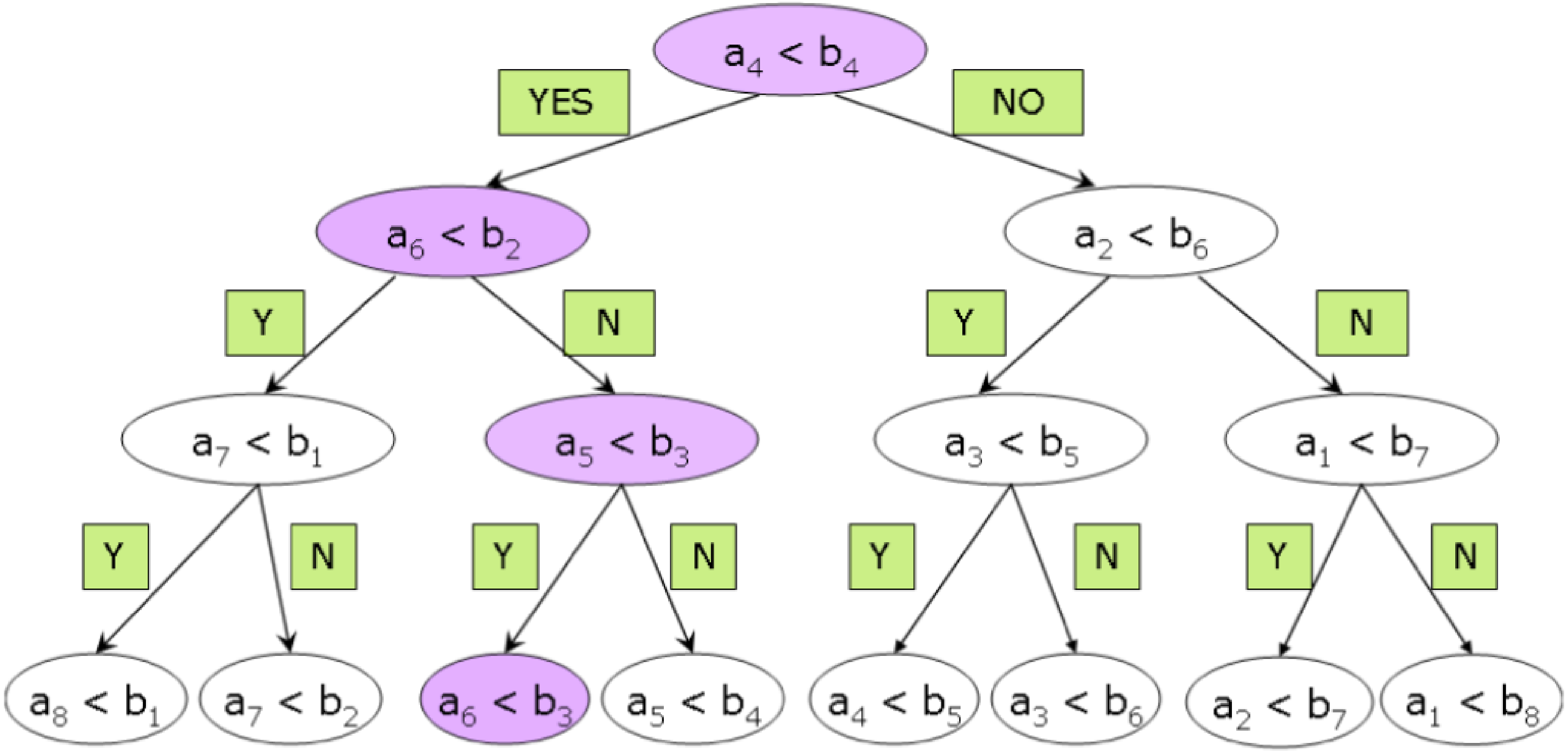
- Unfortunately NOT because we iteratively reveal answer
  - Adversary can adaptively abort in the middle of the computation

SECURE MEDIAN COMPUTATION



Slides borrowed from Benny Pinkas

PROVING MALICIOUS SECURITY



CAN WE ACHIEVE MALICIOUS SECURITY?

- Unfortunately NOT because we iteratively reveal answer
  - Adversary can adaptively abort in the middle of the computation

NEXT BEST THING: Covert Security

Covert Security

Definition (Informal): [Aumann-Lindell`10] A protocol π is said to compute f in the presence of covert adversaries with ε–deterrence if for every PPT Bob and distinguisher D there exists negligible function μ() such that

Pr[Alice outputs "Bob is corrupt"]
≥ ε (Distinguishing gap) – μ(k)

IDEA: After output is revealed, prove that in each step, the greedy update was correctly done

Achieving Covert Security

- Adaptively select inputs
  - Use commitments

- Failure to follow greedy update
  - Use inputs output of order
  - Missing inputs, i.e. use only a subset of inputs committed

- IDEA: Use signatures and consistency checks

Secure Greedy Covert Protocol – High-Level

- Input Commitment Phase: Using an extractable commitment Alice and Bob commit to their inputs.
  - Alice and Bob additional share verification keys for a signature scheme

- Secure Computation Phase: As before iteratively reveal answers. Additionally outputs are signed by both parties.

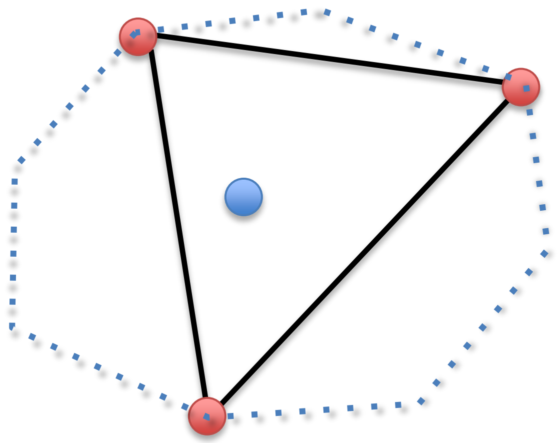
- Consistency Check Phase: A short protocol that shows each input committed in the first phase used correctly

Consistency Checks

For every input commitment prove that the value contained is either

- – In the output, or
- – Not part of the optimal solution

Convex Hull: Show that a particular point is not on the hull.



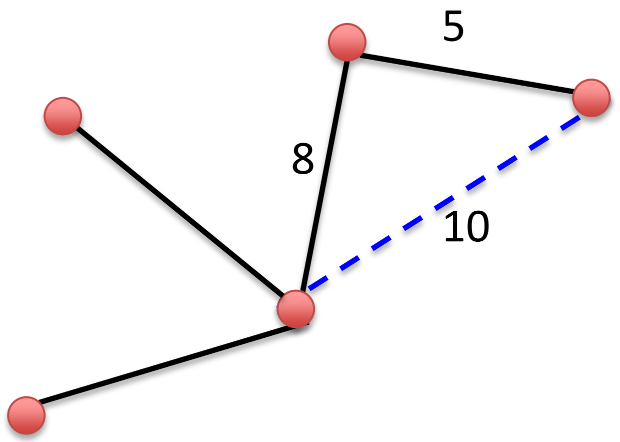
Consistency Checks - Matroids

Let (S,I) be a weighted matroid set system.

Question: How do you show that particular element is not part of minimum cost maximal independent set?

MST: Show that a particular edge does not decrease cost of tree

Show that in the cycle this edge is of maximum cost



Consistency Checks - Matroids

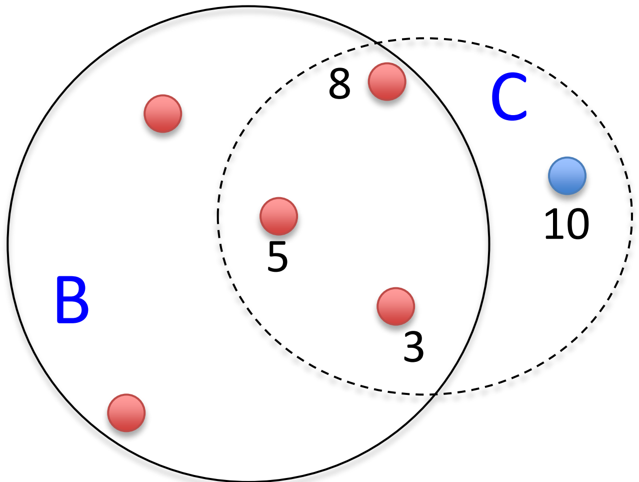
Let (S,I) be a weighted matroid set system.

Question: How do you show that particular element is not part of minimum cost maximal independent set?

Matroid: Show that a particular element does not decrease cost of independent set.

Show that in the *fundamental cycle* this element is of maximum cost

Proof Length: O(|B|) per input



Efficient Consistency Check - MST

- Naïve approach: Cost O(|V|) proof length per edge
- Improve to O(log n) per edge
- IDEA: UNION-FIND data structure
  - Using the pointer data structure: FIND operations cost O(log n) and Union operations cost O(1)
  - Use signatures to get union and find operations attested
- If we use Tarjan's Union-Find, we can improve to $O(\alpha(n))$ where α is the inverse ackerman function.

RESULTS FOR COVERT SECURITY

Algorithm	Our Work (O) COVERT	Circuit (Ω) MALICIOUS
Convex Hull	$O\ell\square I\ell$	$I\ \mathit{log}(I)\ell$
MST	$V\ \mathit{log}(V)\ell$	$(V\alpha(V))^2\ell$
Unit Job Scheduling	$O\ell\square I\ell$	$I^2\ell$
Single Src ADSP	$V\ell\square E\ell$	$E^2\ell$

I	- input size	$\alpha()$	- Inverse Ackerman fn.
O	- output size	V	- #Vertices
ℓ	- integer representation	E	- #Edges

CONCLUSION

- Leverage techniques from algorithms to improve secure computation
- Secure computation using only comparison operations
- OPEN PROBLEM 1: What about other primitives?
- OPEN PROBLEM 2: What about other paradigms?
  - Dynamic Programming
  - Randomized Algorithms