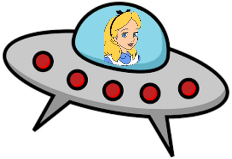
An Optimal Distributed Discrete Log Protocol with Applications to Homomorphic Secret Sharing

Itai Dinur[1], **Nathan Keller**[2] and **Ohad Klein**[2]

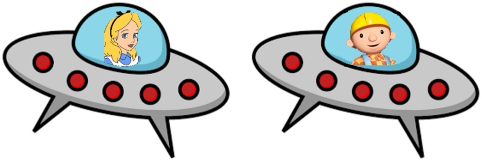
[1] Department of Computer Science, Ben-Gurion University, Israel
[2] Department of Mathematics, Bar-Ilan University, Israel

August 22, 2018

The Spaceships Problem

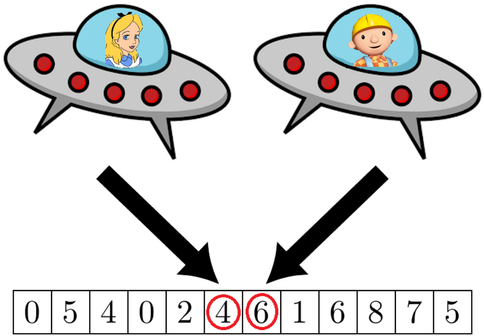


The Spaceships Problem



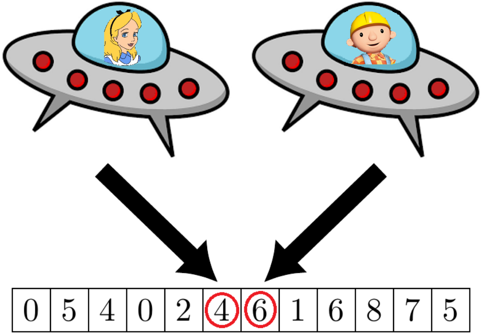
The Spaceships Problem

- Spaceships land on **adjacent** cells of random numbers array.



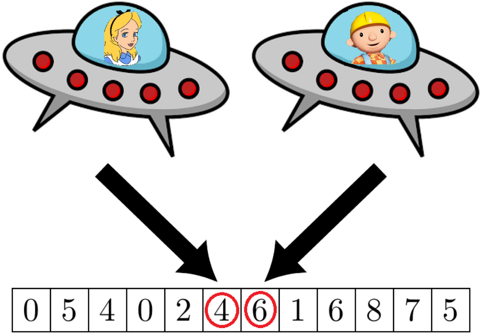
The Spaceships Problem

- Spaceships land on **adjacent** cells of random numbers array.
- **Cannot communicate**.
- Allowed to read T cells.



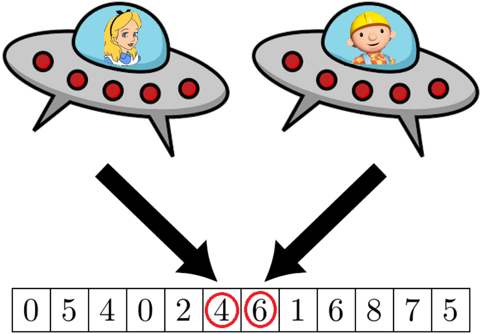
The Spaceships Problem

- Spaceships land on **adjacent** cells of random numbers array.
- **Cannot communicate**.
- Allowed to read T cells.
- Must eventually stop.
- Goal: Stop on the **same cell** with high probability.



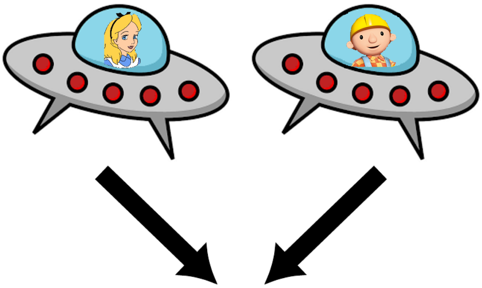
The Spaceships Problem

- Spaceships land on **adjacent** cells of random numbers array.
- **Cannot communicate**.
- Allowed to read T cells.
- Must eventually stop.
- Goal: Stop on the **same cell** with high probability.
- Do not know who is on the left.



The Spaceships Problem

- Spaceships land on **adjacent** cells of random numbers array.
- **Cannot communicate**.
- Allowed to read T cells.
- Must eventually stop.
- Goal: Stop on the **same cell** with high probability.
- Do not know who is on the left.



0	5	4	0	2	4	6	1	6	8	7	5

Main Problem

- How can the spaceships maximize their meeting probability?
- What is this highest probability (depending on T)?

Basic algorithm

Algorithm: Basic

1: **function** $\textsc{Basic}(array, start, T)$

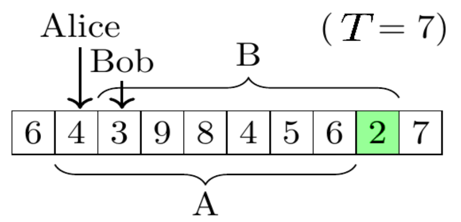
2: **return** $arg\ min_{i \in [start, start+T)} \{array[i]\}$;

Basic algorithm

Algorithm: Basic

1: **function** BASIC($array, start, T$)
2: **return** $arg\ min_{i \in [start, start+T)} \{array[i]\}$;

Analysis



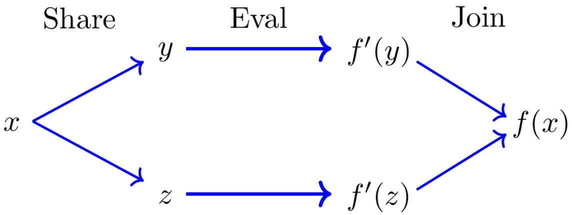
Alice & Bob fail to synchronize iff minimum in on one of the ends.
Probability $= 2/(T+1)$.

Homomorphic Secret Sharing

- **Homomorphic Secret Sharing** – introduced by Boyle,Gilboa, Ishai [**BGI**] (CRYPTO'16) as a more practical alternative to **FHE** (Fully-Homomorphic-Encryption).
- Suppose we wish to securely compute in the cloud.
- HSS enables to distribute the evaluation of a **public function** f on a **secret input** x among two servers, each receiving a secret share y or z, so that $f(x)$ can easily be recovered from $f'(y)$ and $f'(z)$.

Homomorphic Secret Sharing

- **Homomorphic Secret Sharing** – introduced by Boyle,Gilboa, Ishai [**BGI**] (CRYPTO'16) as a more practical alternative to **FHE** (Fully-Homomorphic-Encryption).
- Suppose we wish to securely compute in the cloud.
- HSS enables to distribute the evaluation of a **public function** f on a **secret input** x among two servers, each receiving a secret share y or z, so that $f(x)$ can easily be recovered from $f'(y)$ and $f'(z)$.
  - Each of y and z computationally hides x.



  - 'Share' and 'Join' are cheap.

HSS & Applications

- BGI constructed a group based HSS protocol.
  - Security Relies **only on DDH** (Decisional-Diffie-Hellman) Hardness assumption.
  - **Low communication complexity**.
  - **Applicable only for** functions f inside the class of '**branching programs**'.

HSS & Applications

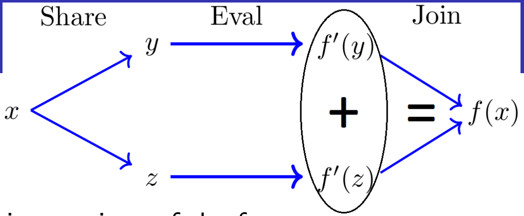
- BGI constructed a group based HSS protocol.
  - Security Relies **only on DDH** (Decisional-Diffie-Hellman) Hardness assumption.
  - **Low communication complexity**.
  - **Applicable only for** functions f inside the class of '**branching programs**'.
- Applications
  - **PIR**: Private information retrieval (with branching program predicate).
  - **SMPC**: Secure multi-party computation in sublinear communication (leveled circuits).

HSS & Applications

- BGI constructed a group based HSS protocol.
  - Security Relies **only on DDH** (Decisional-Diffie-Hellman) Hardness assumption.
  - **Low communication complexity**.
  - **Applicable only for** functions f inside the class of '**branching programs**'.
- Applications
  - **PIR**: Private information retrieval (with branching program predicate).
  - **SMPC**: Secure multi-party computation in sublinear communication (leveled circuits).

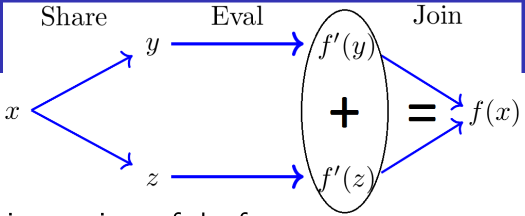
- Requires an algorithm solving the Distributed-Discrete-Log problem (**DDLOG**).

Overview of BGI's HSS Protocol (1)



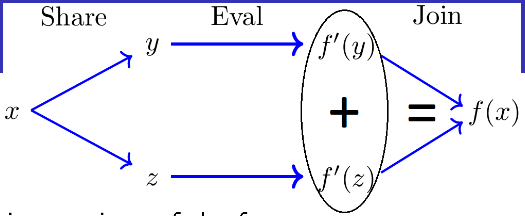
- Let $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ be a secret input.
- We wish to compute $f(x) \in \mathbb{Z}$, where f contains instructions of the form:
  1) $v_i \leftarrow v_j \pm v_k$ for variables $v_i, v_j, v_k \in \mathbb{Z}$.
  2) $v_i \leftarrow v_j \cdot x_k$ where x_k is an input bit.
  3) $v_i \leftarrow x_k$.

Overview of BGI's HSS Protocol (1)



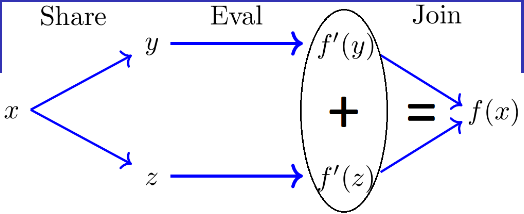
- Let $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ be a secret input.
- We wish to compute $f(x) \in \mathbb{Z}$, where f contains instructions of the form:
  1) $v_i \leftarrow v_j \pm v_k$ for variables $v_i, v_j, v_k \in \mathbb{Z}$.
  2) $v_i \leftarrow v_j \cdot x_k$ where x_k is an input bit.
  3) $v_i \leftarrow x_k$.
- $\textsc{Share}$ x:
  - Let $\mathbb{G}$ be a cryptographic group generated by g.
  - Choose random y_i, z_i with $x_i = y_i + z_i$, and **share** $\mathbf{y_i}, \mathbf{z_i}$ respectively.
  - Also, **publish** $\mathbf{g^{x_i}}$. (Actually, use something similar to El-Gamal.)

Overview of BGI's HSS Protocol (1)



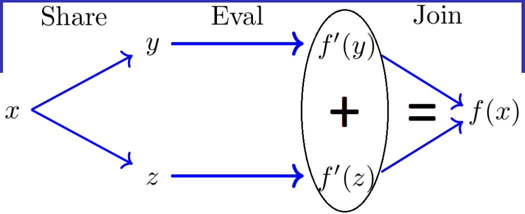
- Let $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ be a secret input.
- We wish to compute $f(x) \in \mathbb{Z}$, where f contains instructions of the form:
  1) $v_i \leftarrow v_j \pm v_k$ for variables $v_i, v_j, v_k \in \mathbb{Z}$.
  2) $v_i \leftarrow v_j \cdot x_k$ where x_k is an input bit.
  3) $v_i \leftarrow x_k$.
- $\textsc{Share}$ x:
  - Let $\mathbb{G}$ be a cryptographic group generated by g.
  - Choose random y_i, z_i with $x_i = y_i + z_i$, and **share $\mathbf{y_i, z_i}$** respectively.
  - Also, **publish $\mathbf{g^{x_i}}$**. (Actually, use something similar to El-Gamal.)
- Evaluation of $\mathbf{f'}$ almost **identical to evaluation of $\mathbf{f}$**.
- We maintain that at any time t, $\mathbf{v_i^t(x)} = \mathbf{v_i^t(y)} + \mathbf{v_i^t(z)}$.
  - Trivial for instructions 1 & 3. What about 2?

Overview of BGI's HSS Protocol (2)



- How would we implement $v_i \leftarrow v_j \cdot x_k$?
- We only have v_j and g^{x_k}!

Overview of BGI's HSS Protocol (2)



- How would we implement $v_i \leftarrow v_j \cdot x_k$?
- We only have v_j and g^{x_k}!
- We can compute $g^{v_i} = g^{v_j x_k}$, but we do not have v_i.
- We seek for an **efficient** probabilistic algorithm $A \colon \mathbb{G} \to \mathbb{Z}/|\mathbb{G}|$ satisfying:

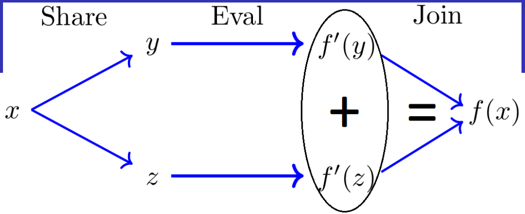
Simplified DDLOG problem

Let $u, v \in \{0, 1\}$, then

$$\Pr\left[A(g^u) + A(g^v) \neq u + v\right] \leq \delta,$$

for a minimal $\delta > 0$, depending on the complexity of A.

Overview of BGI's HSS Protocol (2)



- How would we implement $v_i \leftarrow v_j \cdot x_k$?
- We only have v_j and g^{x_k}!
- We can compute $g^{v_i} = g^{v_j x_k}$, but we do not have v_i.
- We seek for an **efficient** probabilistic algorithm $A\colon \mathbb{G} \to \mathbb{Z}/|\mathbb{G}|$ satisfying:

Simplified DDLOG problem

Let $u, v \in \{0, 1\}$, then

$$\Pr\left[A(g^u) + A(g^v) \neq u + v\right] \leq \delta,$$

for a minimal $\delta > 0$, depending on the complexity of A.

- Can binary expand v_j and assume $v_j \in \{0, 1\}$.
- Degenerate formulation due to usage $t \mapsto g^t$, instead of El-Gamal.

DDLOG & Spaceships problems

DDLOG problem

Let $\mathbb{G}$ be a cyclic cryptographic group, with a generator g.
Find probabilistic algorithms $A, B\colon \mathbb{G} \to \mathbb{Z}/|\mathbb{G}|$, so that

$$\forall x \in \mathbb{Z}: \qquad \Pr\left[A(g^{x+1}) - B(g^{x}) \neq 1\right] \leq \delta,$$

for a minimal $\delta > 0$, depending on the time complexity of A, B.

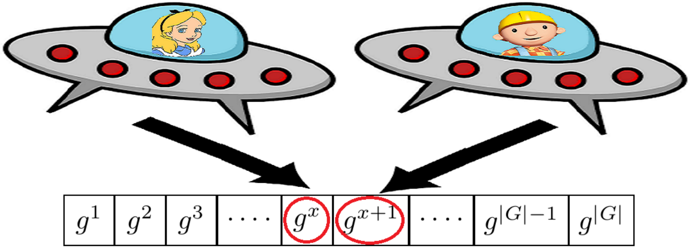
DDLOG & Spaceships problems

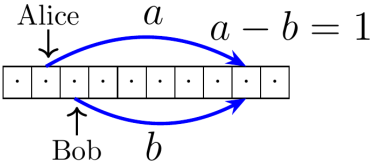
DDLOG problem

Let $\mathbb{G}$ be a cyclic cryptographic group, with a generator g.
Find probabilistic algorithms $A, B\colon \mathbb{G} \to \mathbb{Z}/|\mathbb{G}|$, so that

$$\forall x \in \mathbb{Z}: \qquad \Pr\left[A(g^{x+1}) - B(g^{x}) \neq 1\right] \leq \delta,$$

for a minimal $\delta > 0$, depending on the time complexity of A, B.





- Apply a PRF on g^t to randomize.
- BGI used 'Basic' algorithm, achieving $\delta = 1/T$ for running time $O(T)$.

Results

(Optimal) Spaceships Algorithm

There is an algorithm enabling Alice and Bob to synchronize except for probability $O(1/T^2)$, where T is the number of array-queries.

Results

(Optimal) Spaceships Algorithm

There is an algorithm enabling Alice and Bob to synchronize except for probability $O(1/T^2)$, where T is the number of array-queries.

Corollary

If Alice and Bob landed with initial distance M of each other, probability of failure would be $O(M/T^2)$.

Results

(Optimal) Spaceships Algorithm

There is an algorithm enabling Alice and Bob to synchronize except for probability $O(1/T^2)$, where T is the number of array-queries.

Corollary

If Alice and Bob landed with initial distance M of each other, probability of failure would be $O(M/T^2)$.

Spaceships Optimality

This is optimal! (No algorithm can achieve $o(1/T^2)$ error.)

Results

(Optimal) Spaceships Algorithm

There is an algorithm enabling Alice and Bob to synchronize except for probability $O(1/T^2)$, where T is the number of array-queries.

Corollary

If Alice and Bob landed with initial distance M of each other, probability of failure would be $O(M/T^2)$.

Spaceships Optimality

This is optimal! (No algorithm can achieve $o(1/T^2)$ error.)

DDLOG optimality

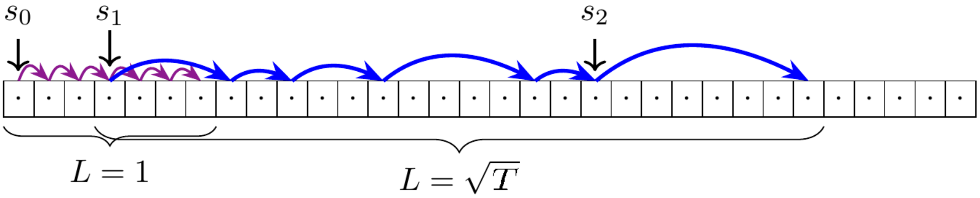
Algorithm is optimal in groups satisfying the DLI-Hardness assumption.

Optimal Algorithm

- Algorithm is composed of several ‘jumping’ phases.
- The ‘Jump’ algorithm is a variant of Pollard’s kangaroo algorithm.

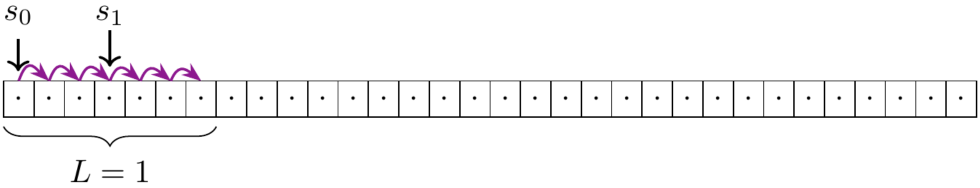
Optimal Algorithm

- Algorithm is composed of several 'jumping' phases.
- The 'Jump' algorithm is a variant of Pollard's kangaroo algorithm.
- We view a 2-staged algorithm, achieving $O(1/T^{3/2})$ error, with T queries.



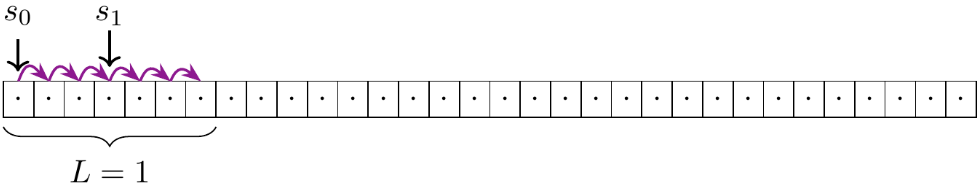
2 staged algorithm

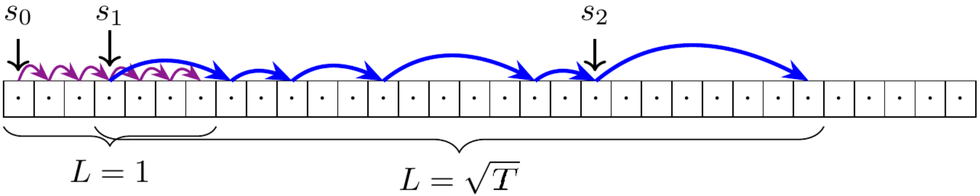
- Phase 1 is just 'Basic' algorithm with $T/2$ steps.



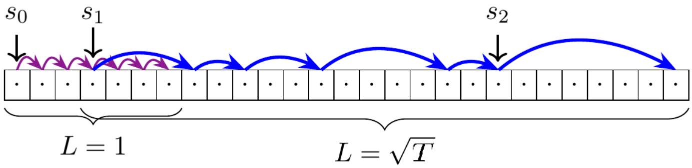
2 staged algorithm

- Phase 1 is just 'Basic' algorithm with $T/2$ steps.
- Phase 2 performs $T/2$ jumps of size $\sim U(1, \sqrt{T})$ depending on current location.
- Phase 2 starts from minimum of phase 1; stops at its own minimum.



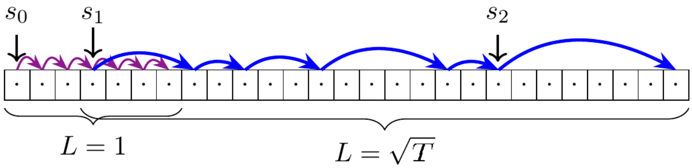


Analysis of algorithm



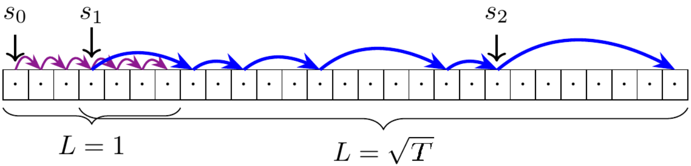
- In each stage, both Alice and Bob perform a *Jump*.
- In first stage, fail to synchronize with probability $\mathbf{O(1/T)}$.

Analysis of algorithm



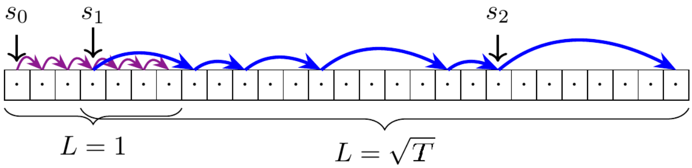
- In each stage, both Alice and Bob perform a *Jump*.
- In first stage, fail to synchronize with probability $\mathbf{O(1/T)}$.
- After synchronizing, parties remain synchronized forever!
- Parties failed on first stage $\Rightarrow$ their distance is $\mathbf{O(T)}$.

Analysis of algorithm



- In each stage, both Alice and Bob perform a *Jump*.
- In first stage, fail to synchronize with probability $\mathbf{O(1/T)}$.
- After synchronizing, parties remain synchronized forever!
- Parties failed on first stage $\Rightarrow$ their distance is $\mathbf{O(T)}$.
- On second stage, parties make $T/2$ steps of size $\leq L = \sqrt{T}$, their random walks are expected to meet after $\mathbf{O(\sqrt{T})}$ steps.
- Thus, walks share $T - O(\sqrt{T})$ steps $\Rightarrow$ fail with prob. $\mathbf{O(\sqrt{T}/T)}$.

Analysis of algorithm

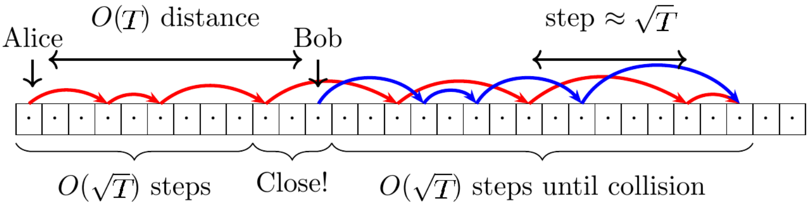


- In each stage, both Alice and Bob perform a *Jump*.
- In first stage, fail to synchronize with probability $\mathbf{O(1/T)}$.
- After synchronizing, parties remain synchronized forever!
- Parties failed on first stage $\Rightarrow$ their distance is $\mathbf{O(T)}$.
- On second stage, parties make $T/2$ steps of size $\leq L = \sqrt{T}$, their random walks are expected to meet after $\mathbf{O(\sqrt{T})}$ steps.
- Thus, walks share $T - O(\sqrt{T})$ steps $\Rightarrow$ fail with prob. $\mathbf{O(\sqrt{T}/T)}$.

$$\Downarrow$$

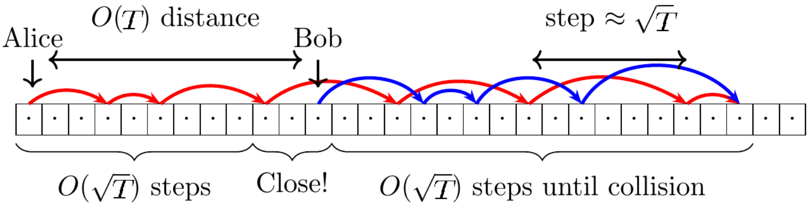
- After second stage, synchronized except for probability $\mathbf{O(T^{-3/2})}$!

- Why would two random walks, starting $O(T)$ apart, with step-size $O(\sqrt{T})$, collide after $O(\sqrt{T})$ steps?



Analysis of algorithm

- Why would two random walks, starting $O(T)$ apart, with step-size $O(\sqrt{T})$, collide after $O(\sqrt{T})$ steps?



- In general, if distance $\sim D$, and step size $\sim L$, collision after $\sim D/L + L$ steps.
  - Choose $L = \sqrt{D}$ to minimize collision time.

Distant Alice & Bob

Summary

In case Alice and Bob start with distance 1 apart, they can meet except for probability $O(1/T^2)$, by reading T cells.

- What if their initial distance is M?

Distant Alice & Bob

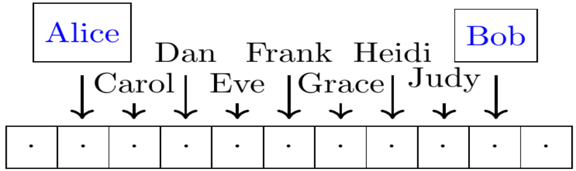
Summary

In case Alice and Bob start with distance 1 apart, they can meet except for probability $O(1/T^2)$, by reading T cells.

- What if their initial distance is M?

Answer

By using the same algorithm, they meet except for probability $O(M/T^2)$.



$$\Pr[f(\mathrm{A}) \neq f(\mathrm{B})] \leq \Pr[f(\mathrm{A}) \neq f(\mathrm{C})] + \cdots + \Pr[f(\mathrm{J}) \neq f(\mathrm{B})]$$

Lower Bound on DDLOG (assuming DLI)

Discrete-Log-in-Interval Hardness assumption

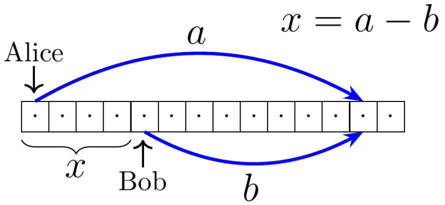
Many concrete (families of) cryptographic groups $\mathbb{G}$ with generator g, satisfy:
No algorithm can find x, given g^x, in time $o(\sqrt{R})$, assuming $x \sim U(0, R)$.

Lower Bound on DDLOG (assuming DLI)

Discrete-Log-in-Interval Hardness assumption

Many concrete (families of) cryptographic groups $\mathbb{G}$ with generator g, satisfy:
No algorithm can find x, given g^x, in time $o(\sqrt{R})$, assuming $x \sim U(0, R)$.

- Spaceships algorithm solves DLI in $O(\sqrt{R})$.
  - Land Alice & Bob on the array $\texttt{arr}[t] = g^t$.
  - Put Alice at g^0, and Bob at g^x.
  - Let them perform $O(2\sqrt{R})$ queries.
  - Finds DLOG except for prob. $x/(2\sqrt{R})^2 < 1/2$.
- Hence optimal!



- Solution for the spaceships problem can truly be proved to be optimal up to constant factors.

Summary

- The Distributed Discrete Log Problem.
- Application to HSS.
- An optimal algorithm solving DDLOG. (Improving $O(1/T)$ to $O(1/T^2)$)
- A formal analysis of the algorithm is in the paper.
- A matching lower bound assuming DLI hardness assumption.

Techniques

- Iterated variant of **Pollard's Kangaroo** algorithm.
- Analysis of algorithm with **Martingales**.
- Lower bounds with **Discrete Fourier Analysis**.

Thank You!

Algorithm: Optimal

function $\textsc{Optimal}(arr, start, T)$
    $s = \textsc{Jump}(arr, start, T, 1);$
    $s = \textsc{Jump}(arr, s, T, T^{1/2});$
    $s = \textsc{Jump}(arr, s, T, T^{3/4});$
    $s = \textsc{Jump}(arr, s, T, T^{7/8});$
    $\vdots \quad \vdots \quad \vdots \quad \vdots$
    return s;

Algorithm: Jump

function $\textsc{Jump}(arr, s, T, L)$
    $i, min = s, s;$
    repeat T times
        if $arr[i] < arr[min]$ **then**
            $min_i = i;$
        $i \mathrel{+}= 1 + \textbf{Hash}(arr[i]) \% L;$
    return min;