Data-oblivious Computation

A common paradigm for private and secure computation

David Darais (Galois, Inc.)

TWO WORLDS

Traditional Computation **???**

TWO WORLDS

Traditional Computation

Bubble sort is the worst

???

Bubble sort is the best

TWO WORLDS

Traditional Computation	**???**
Bubble sort is the worst	Bubble sort is the best
Insertion sort is mediocre	Insertion sort is (also) the best
Merge sort is the best	Merge sort is the worst

TWO WORLDS

Traditional Computation	???
Bubble sort is the worst	Bubble sort is the best
Insertion sort is mediocre	Insertion sort is (also) the best
Merge sort is the best	Merge sort is the worst
RAM/array access is O(1)	RAM/array access is O(log(N))
Data structures + recursion = easy	Data structures + recursion = hard

TWO WORLDS

Traditional Computation	Data-oblivious Computation
Bubble sort is the worst	Bubble sort is the best
Insertion sort is mediocre	Insertion sort is (also) the best
Merge sort is the best	Merge sort is the worst
RAM/array access is O(1)	RAM/array access is O(log(N))
Data structures + recursion = easy	Data structures + recursion = hard

Fully Homomorphic Encryption

Secure Multiparty Computation

Constant Time Execution

Information Flow Control

Differential Privacy

Protects secrets ***during computation***

Protects secrets ***after release***

Fully Homomorphic Encryption	Secure Multiparty Computation	Constant Time Execution	Information Flow Control	Differential Privacy

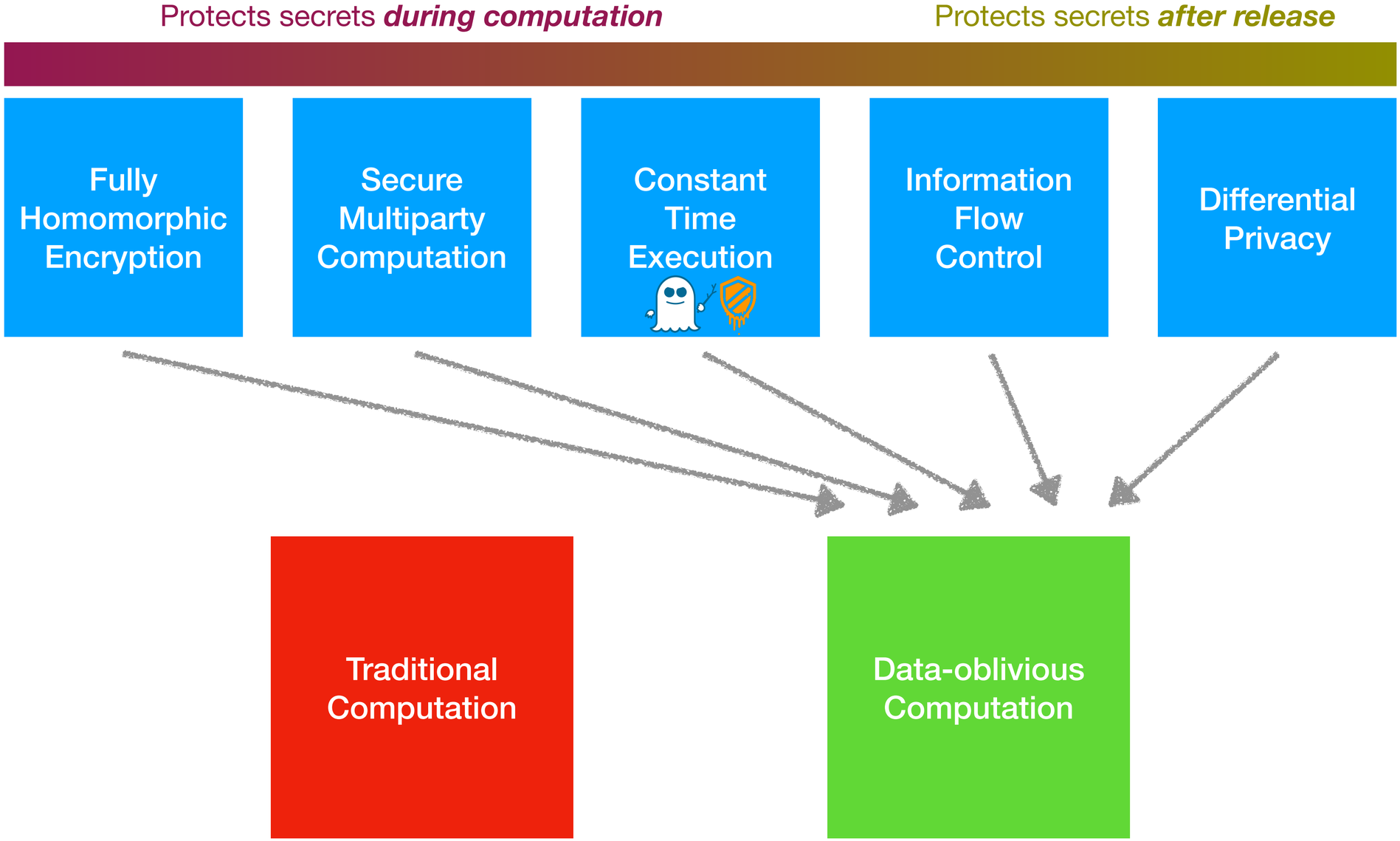
Protects secrets ***during computation***

Protects secrets ***after release***

Fully Homomorphic Encryption	Secure Multiparty Computation	Constant Time Execution	Information Flow Control	Differential Privacy

Performance is critical

Correctness is critical


Protects secrets *during computation*
Protects secrets *after release*
Fully Homomorphic Encryption
Secure Multiparty Computation
Constant Time Execution
Information Flow Control
Differential Privacy
Traditional Computation
Data-oblivious Computation

TWO WORLDS

Traditional Computation	**Data-oblivious Computation**
Bubble sort is the worst	Bubble sort is the best
Insertion sort is mediocre	Insertion sort is (also) the best
Merge sort is the best	Merge sort is the worst
RAM/array access is O(1)	RAM/array access is O(log(N))
Data structures + recursion = easy	Data structures + recursion = hard

TWO WORLDS

PROBLEM

No programming language or compiler understands data-oblivious computation

CONTROL FLOW

Traditional Computation

```
if (raining) {
    ice_in_coffee = false;
    flavor = “hazelnut";
    drink = new_coffee(ice_in_coffee,
                       flavor,
                       null);
} else {
    ice_in_coffee = true;
    additive = “cream";
    drink = new_coffee(ice_in_coffee,
                       null,
                       additive);
}
```

CONTROL FLOW

Traditional Computation

```
if (raining) {
  ice_in_coffee = false;
  flavor = "hazelnut";
  drink = new_coffee(ice_in_coffee,
                     flavor,
                     null);
} else {
  ice_in_coffee = true;
  additive = "cream";
  drink = new_coffee(ice_in_coffee,
                     null,
                     additive);
}
```

Data-oblivious Computation

```
ice_in_coffee_A = false;
flavor_A = "hazelnut";
drink_A = new_coffee(ice_in_coffee_A,
                     flavor_A,
                     null);
```

```
ice_in_coffee_B = true;
additive_B = "cream";
drink_B = new_coffee(ice_in_coffee_B,
                     null,
                     additive_B);
```

```
drink = raining ? drink_A : drink_B;
```

EARLY DAYS FOR DO

Traditional Computation

Data-oblivious Computation

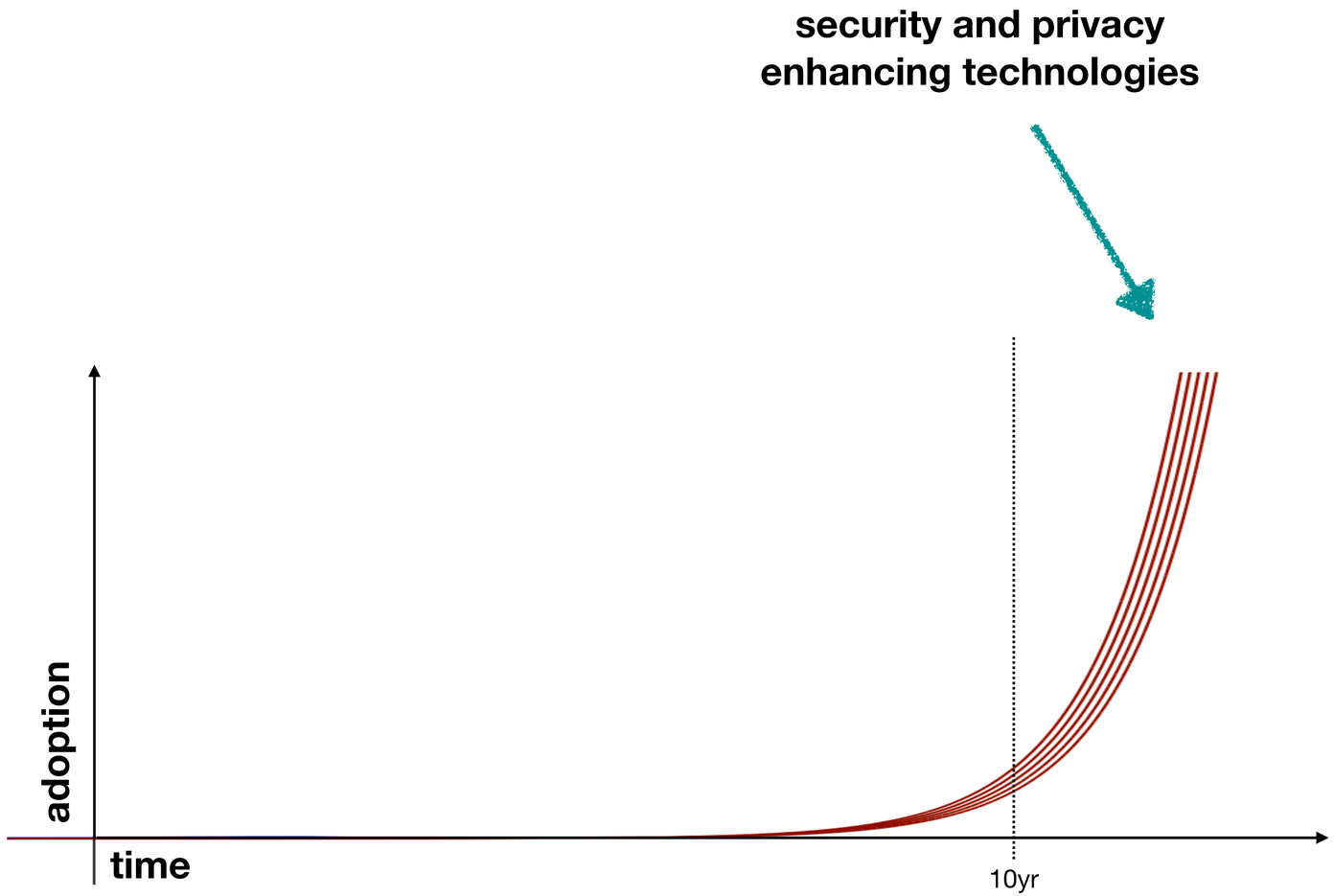
EARLY DAYS FOR DO

Traditional Computation	Data-oblivious Computation
Decades of hardware/ISA research	
Decades of compiler/IR research	
Formal methods and analysis tools	
100% bug-free tools (CompCert)	
High performance applications	

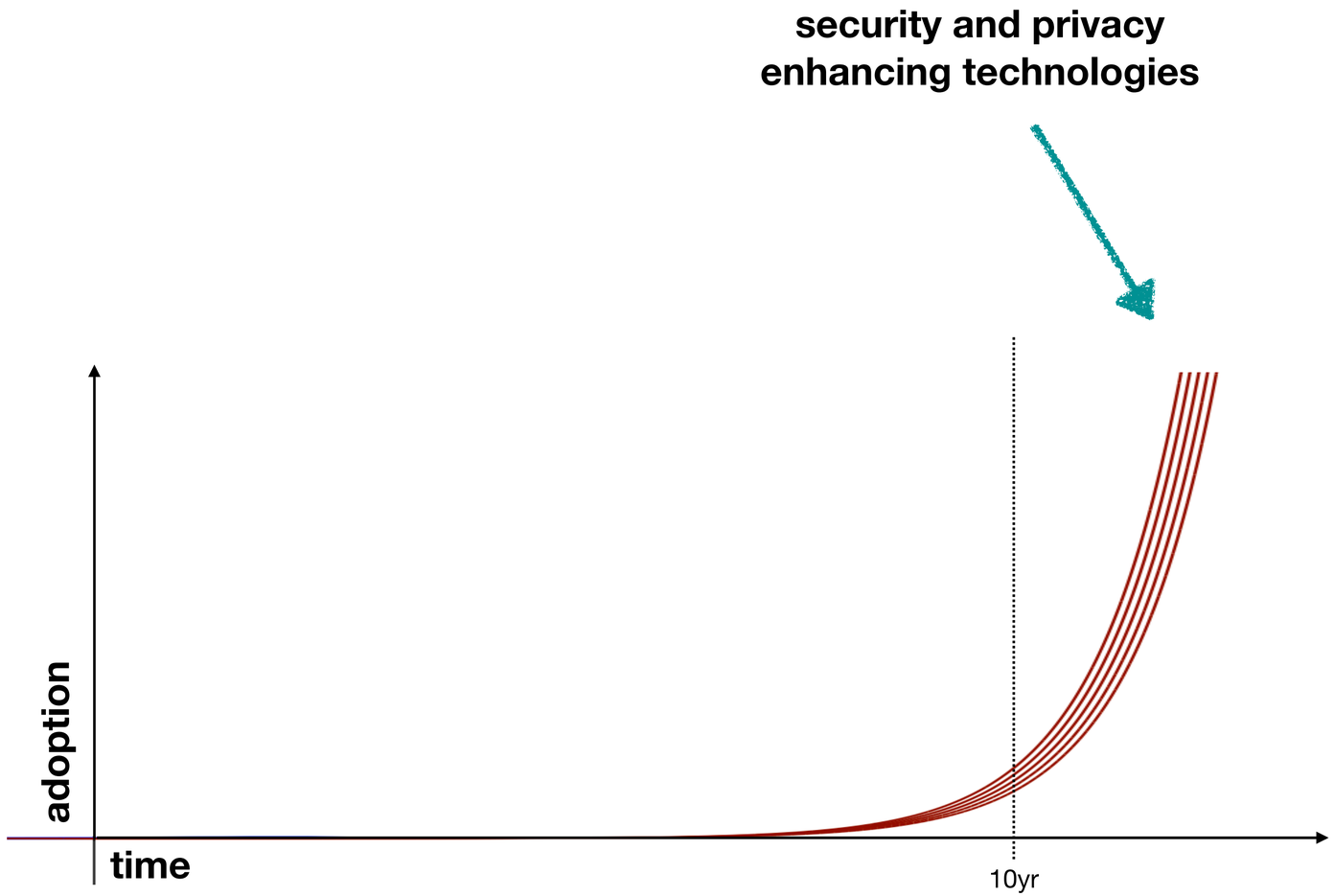
EARLY DAYS FOR DO

Traditional Computation	Data-oblivious Computation
Decades of hardware/ISA research	<1 decade of hardware/ISA research
Decades of compiler/IR research	<1 decade of compiler/IR research
Formal methods and analysis tools	~0 formal methods and analysis tools
100% bug-free tools (CompCert)	~0 bug-free tools
High performance applications	Low performance applications

A CALL TO ARMS



A CALL TO ARMS



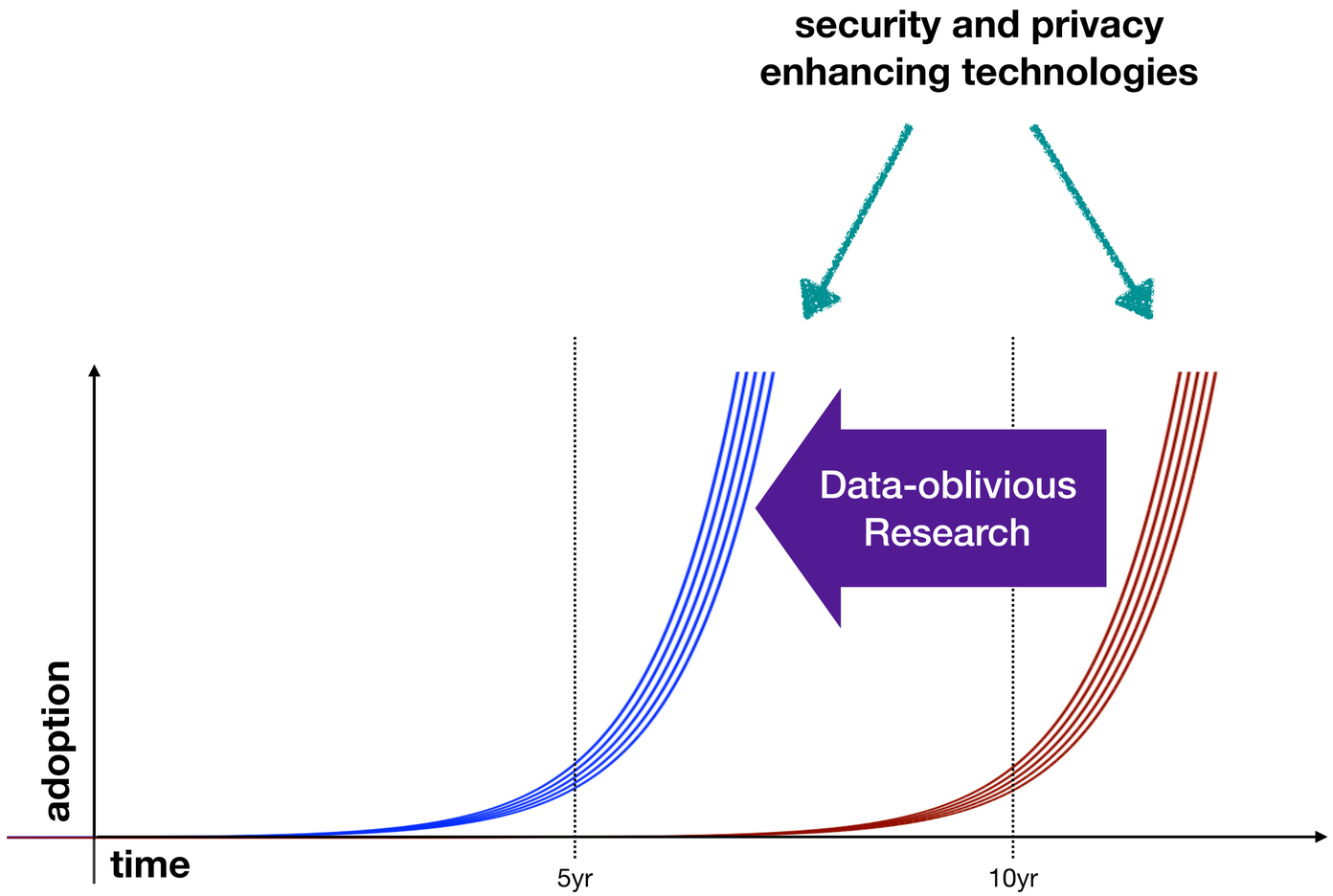
WHERE WE ARE

Oblivious Data Structures (ORAM)

DO ISAs (TinyRam)

DO Formal Methods (PMTO)

A CALL TO ARMS



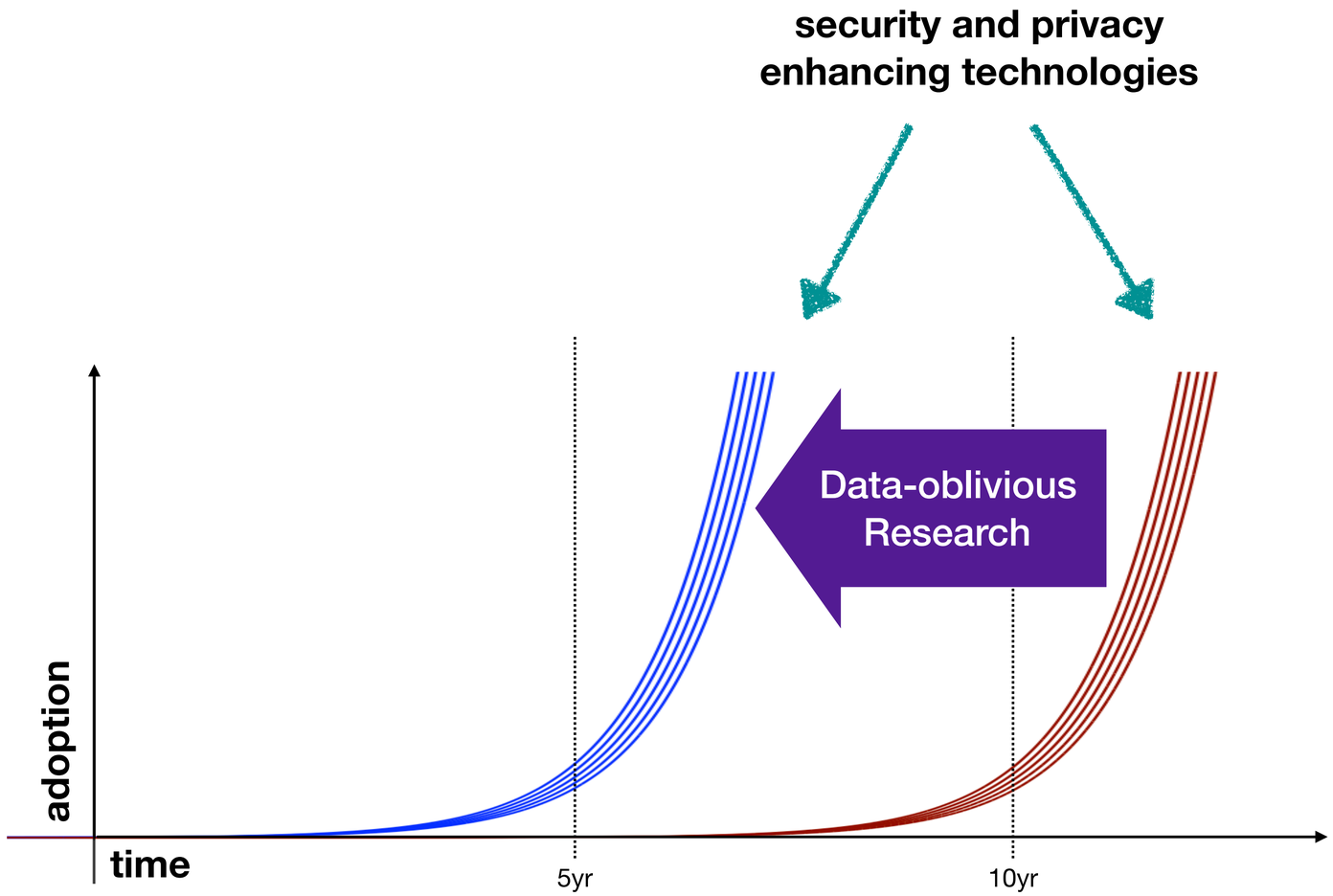
WHERE WE ARE

Oblivious Data Structures (ORAM)

DO ISAs (TinyRam)

DO Formal Methods (PMTO)

A CALL TO ARMS



WHERE WE ARE

Oblivious Data Structures (ORAM)

DO ISAs (TinyRam)

DO Formal Methods (PMTO)

WHAT WE NEED

Prog. languages for DO (**usability**)

Compilers for DO (**performance**)

Static analyses for DO (**accuracy**)

Data structures + libs for DO (**flexibility**)

Formal methods for DO (**assurance**)

Domain independence (**ubiquity**)

END

RAM ACCESS

Traditional Computation

```
if (raining) {
  shopping[next_id] = "umbrella";
} else {
  shopping[next_id] = "sunglasses";
}
next_id++;
```

RAM ACCESS

Traditional Computation

```
if (raining) {
  shopping[next_id] = "umbrella";
} else {
  shopping[next_id] = "sunglasses";
}
next_id++;
```

Data-oblivious Computation

```
tmp = shopping[next_id];
shopping[next_id] = raining  ? "umbrella"
                             : tmp;
tmp = shopping[next_id];
shopping[next_id] = !raining ? "sunglasses"
                             : tmp;
next_id++;
```