A New Approach to Practical Active-Secure Two-Party Computation

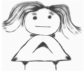
Jesper Buus Nielsen[1], Peter Sebastian Nordholt[1], Claudio Orlandi[1], Sai Sheshank Burra[2]

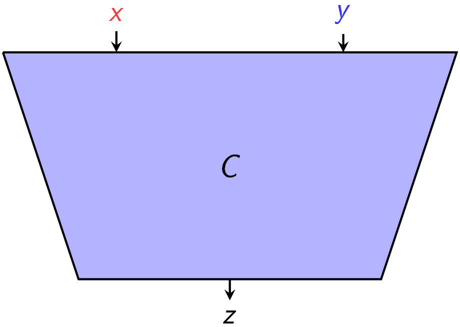
[1]Aarhus University, Denmark

[2]Indian Institute of Technology Guwahati

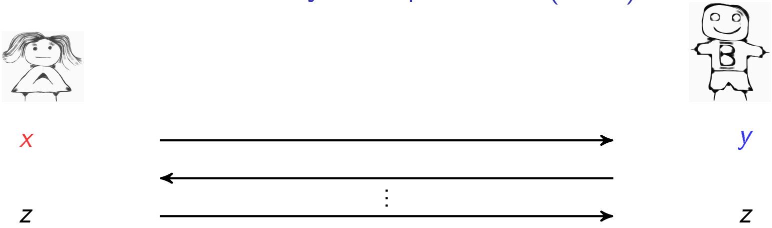
August 21, 2012

Active-Secure Two-Party Computation (2PC)

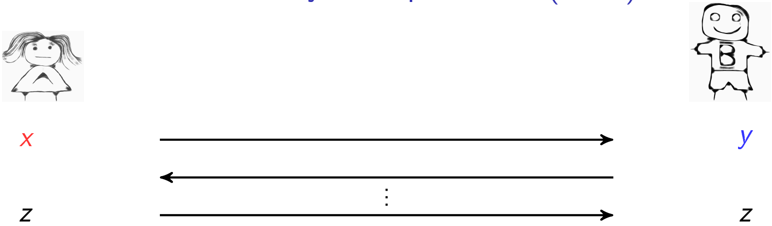




Active-Secure Two-Party Computation (2PC)

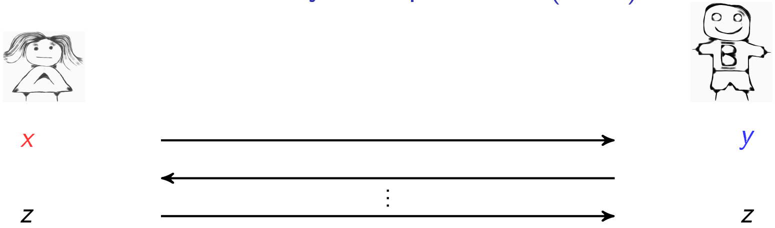


Active-Secure Two-Party Computation (2PC)



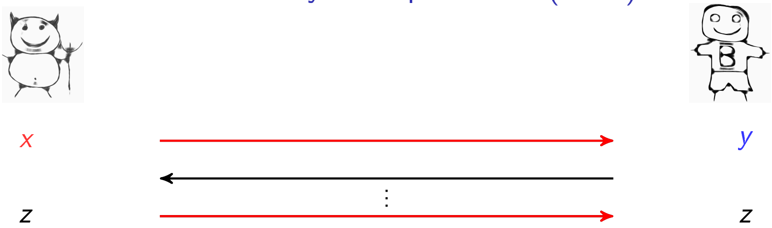
- Correctness: $C(x, y) = z$

Active-Secure Two-Party Computation (2PC)



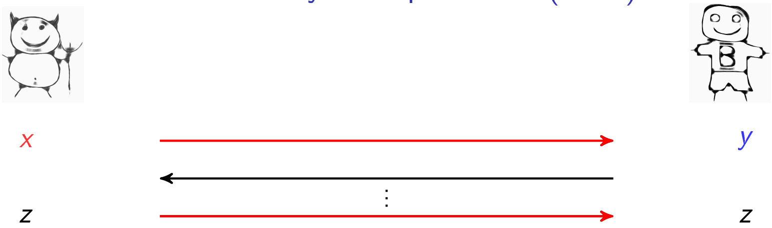
- Correctness: $C(x, y) = z$
- Privacy: Inputs are kept private.

Active-Secure Two-Party Computation (2PC)



- Correctness: $C(x, y) = z$
- Privacy: Inputs are kept private.

Active-Secure Two-Party Computation (2PC)



- Correctness: $C(x, y) = z$
- Privacy: Inputs are kept private.
- Practical: Runs in reasonable time for reasonable size circuits.

Motivation for this Work

- Solving real-world problems. E.g. computing outcome of auctions [BCD+09].

Motivation for this Work

- Solving real-world problems. E.g. computing outcome of auctions [BCD+09].
- Lack of diversity in practical 2PC. In fact all previous practical approaches uses Yao's Garbled Circuits technique.

Our approach

Building blocks

- Passive-secure 2PC: The protocol of [GMW87] heavily utilizing Oblivious Transfer (OT).

Our approach

Building blocks

- Passive-secure 2PC: The protocol of [GMW87] heavily utilizing Oblivious Transfer (OT).
- Information theoretic MACs: To ensure active security.

Our approach

Building blocks

- Passive-secure 2PC: The protocol of [GMW87] heavily utilizing Oblivious Transfer (OT).
- Information theoretic MACs: To ensure active security.
- OT-extension: A huge amount of OT at low amortized cost from the passive-secure protocol of [IKNP03].

Our Results

- New OT-extension technique with active security:

Our Results

- New OT-extension technique with active security:
  - Only a factor 2 slower than the passive-secure protocol of [IKNP03]. (No asymptotic improvement).

Our Results

- New OT-extension technique with active security:
  - Only a factor 2 slower than the passive-secure protocol of [IKNP03]. (No asymptotic improvement).
  - Implements $\sim$ 500.000 OT/sec pr. core (vs. $\sim$ 1000 OT/sec without extension).

Our Results

- New OT-extension technique with active security:
  - Only a factor 2 slower than the passive-secure protocol of [IKNP03]. (No asymptotic improvement).
  - Implements $\sim$ 500.000 OT/sec pr. core (vs. $\sim$ 1000 OT/sec without extension).
- New practical 2PC protocol:

Our Results

- New OT-extension technique with active security:
  - Only a factor 2 slower than the passive-secure protocol of [IKNP03]. (No asymptotic improvement).
  - Implements $\sim$ 500.000 OT/sec pr. core (vs. $\sim$ 1000 OT/sec without extension).
- New practical 2PC protocol:
  - UC secure against an active and static adversary in the Random Oracle model.

Our Results

- New OT-extension technique with active security:
  - Only a factor 2 slower than the passive-secure protocol of [IKNP03]. (No asymptotic improvement).
  - Implements ~ 500.000 OT/sec pr. core (vs. ~ 1000 OT/sec without extension).
- New practical 2PC protocol:
  - UC secure against an active and static adversary in the Random Oracle model.
  - Implements 20.000 gates/sec (online $\sim 1.000.000$ gates/sec).

Our Results

- New OT-extension technique with active security:
  - Only a factor 2 slower than the passive-secure protocol of [IKNP03]. (No asymptotic improvement).
  - Implements ~ 500.000 OT/sec pr. core (vs. ~ 1000 OT/sec without extension).
- New practical 2PC protocol:
  - UC secure against an active and static adversary in the Random Oracle model.
  - Implements 20.000 gates/sec (online $\sim 1.000.000$ gates/sec).
  - Faster than all implementations based on Garbled Circuits ... except for [KsS12].

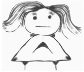
Overview

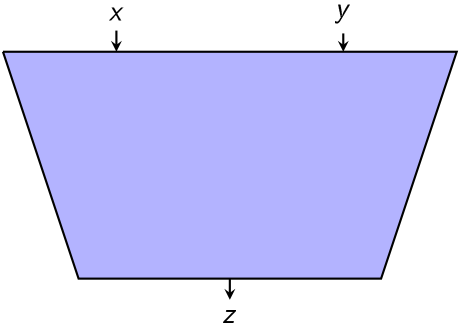
Protocol Overview

MACs

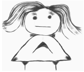
Concluding

Passive-Secure 2PC [GMW87]

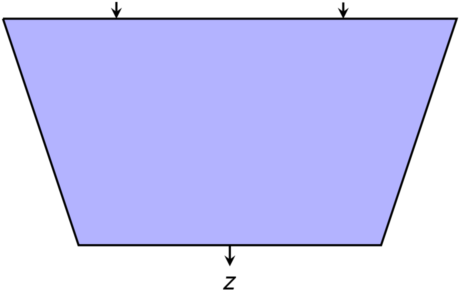




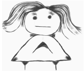
Passive-Secure 2PC [GMW87]



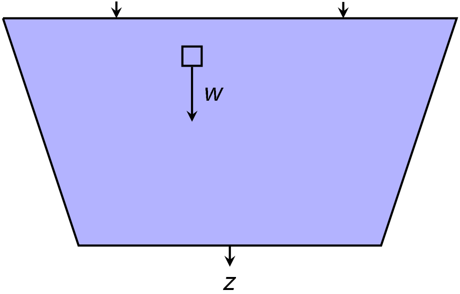
x_A, y_A $\quad$ $x = x_A \oplus x_B$ $\quad$ $y = y_A \oplus y_B$ $\quad$ x_B, y_B



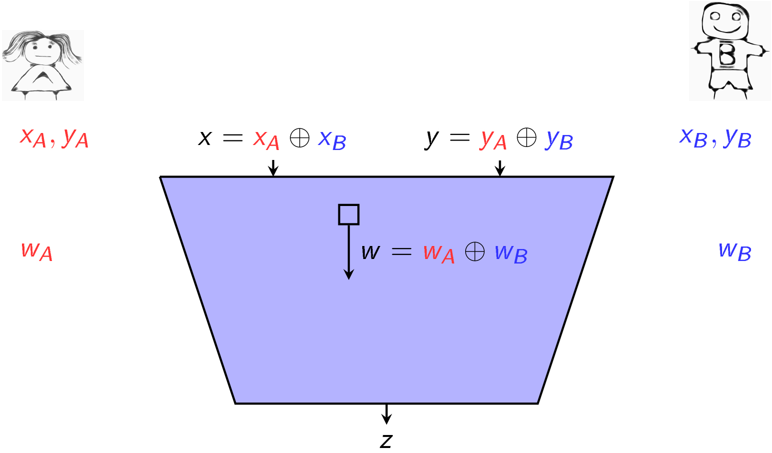
Passive-Secure 2PC [GMW87]



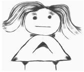
x_A, y_A $\quad x = x_A \oplus x_B \quad y = y_A \oplus y_B \quad$ x_B, y_B



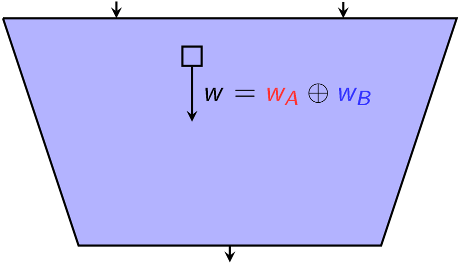
Passive-Secure 2PC [GMW87]



Passive-Secure 2PC [GMW87]



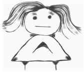
x_A, y_A $\quad x = x_A \oplus x_B \quad y = y_A \oplus y_B \quad$ x_B, y_B

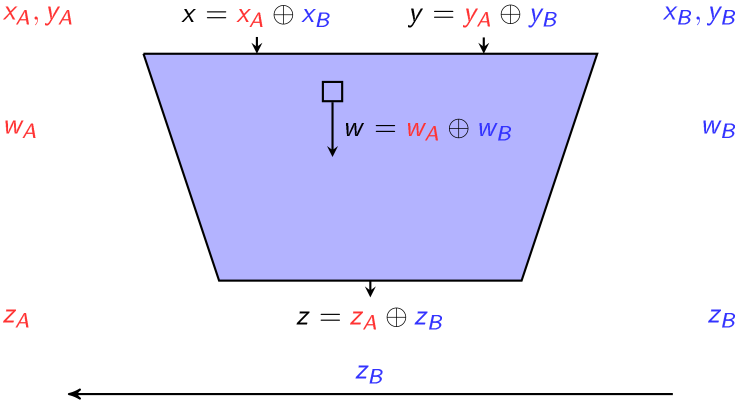


w_A $\quad w = w_A \oplus w_B \quad$ w_B

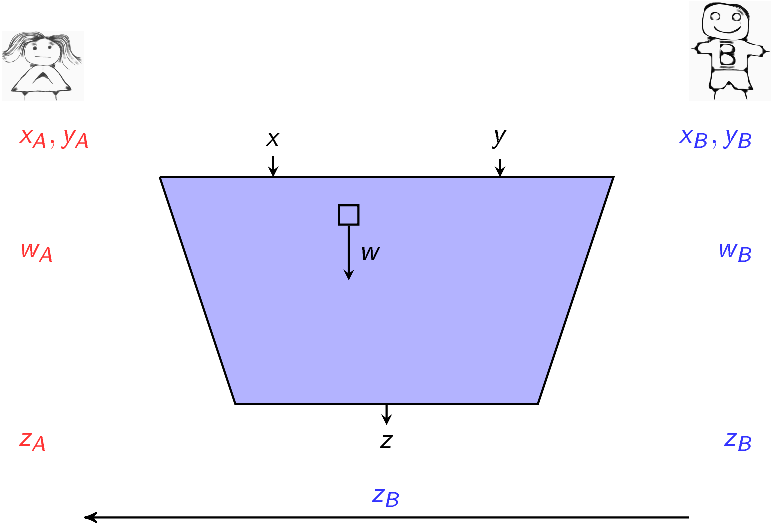
z_A $\quad z = z_A \oplus z_B \quad$ z_B

Passive-Secure 2PC [GMW87]

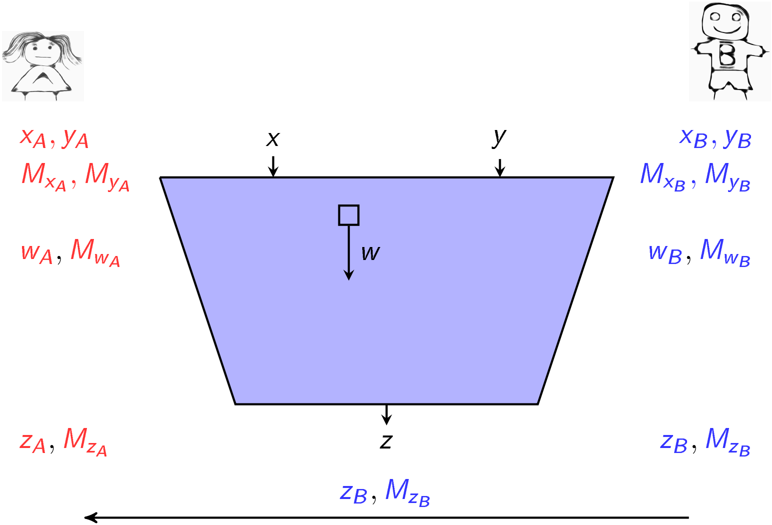




Active-Secure 2PC



Active-Secure 2PC



Add message authentication codes (MACs)

The Preprocessing model

Preprocessing:

The Preprocessing model

Preprocessing:

- Prepare random authenticated messages:

The Preprocessing model

Preprocessing:

- Prepare random authenticated messages:
  - 1 per Input-gate.

The Preprocessing model

Preprocessing:

- Prepare random authenticated messages:
  - 1 per Input-gate.
  - $16B$ per AND-gate, for security $\sim 2^{-B\log(|C|)}$.

The Preprocessing model

Preprocessing:

- Prepare random authenticated messages:
  - 1 per Input-gate.
  - $16B$ per AND-gate, for security $\sim 2^{-B\log(|C|)}$.
- Prove correlations between authenticated messages (think multiplication-triples).

The Preprocessing model

Preprocessing:

- Prepare random authenticated messages:
  - 1 per Input-gate.
  - $16B$ per AND-gate, for security $\sim 2^{-B \log(|C|)}$.
- Prove correlations between authenticated messages (think multiplication-triples).

Online:

The Preprocessing model

Preprocessing:

- Prepare random authenticated messages:
  - 1 per Input-gate.
  - $16B$ per AND-gate, for security $\sim 2^{-B\log(|C|)}$.
- Prove correlations between authenticated messages (think multiplication-triples).

Online:

- Use preprocessed values and simple (non-crypto) protocols to evaluate circuit on actual input.

The Preprocessing model

Preprocessing:

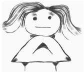
- Prepare random authenticated messages:
  - 1 per Input-gate.
  - $16B$ per AND-gate, for security $\sim 2^{-B \log(|C|)}$.
- Prove correlations between authenticated messages (think multiplication-triples).

Online:

- Use preprocessed values and simple (non-crypto) protocols to evaluate circuit on actual input.

Overview

Information Theoretic MACs



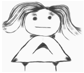
Message $x \in_R \{0,1\}$

MAC $M = K \oplus x\Delta$

Global key $\Delta \in_R \{0,1\}^n$

Local key $K \in_R \{0,1\}^n$

Information Theoretic MACs

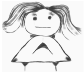


Message $x \in_R \{0,1\}$
MAC $M = K \oplus x\Delta$

Global key $\Delta \in_R \{0,1\}^n$
Local key $K \in_R \{0,1\}^n$

Unforgeability:

Information Theoretic MACs



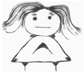
Message $x \in_R \{0,1\}$
MAC $M = K \oplus x\Delta$

Global key $\Delta \in_R \{0,1\}^n$
Local key $K \in_R \{0,1\}^n$

Unforgeability:

- $M = K \oplus x\Delta$ does not give information on Δ.

Information Theoretic MACs



Message $x \in_R \{0,1\}$

MAC $M = K \oplus x\Delta$

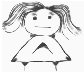
Global key $\Delta \in_R \{0,1\}^n$

Local key $K \in_R \{0,1\}^n$

Unforgeability:

- $M = K \oplus x\Delta$ does not give information on Δ.
- Let $M_0 = K \oplus 0\Delta = K$ and $M_1 = K \oplus 1\Delta = K \oplus \Delta$.

Information Theoretic MACs



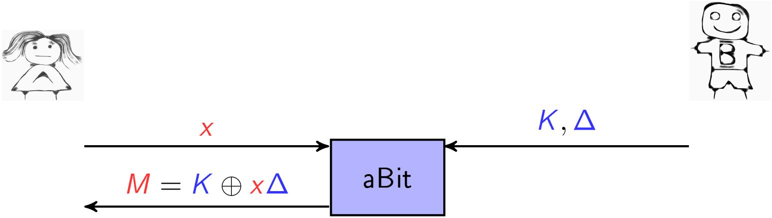
Message $x \in_R \{0,1\}$
MAC $M = K \oplus x\Delta$

Global key $\Delta \in_R \{0,1\}^n$
Local key $K \in_R \{0,1\}^n$

Unforgeability:

- $M = K \oplus x\Delta$ does not give information on Δ.
- Let $M_0 = K \oplus 0\Delta = K$ and $M_1 = K \oplus 1\Delta = K \oplus \Delta$.
- $M_0 \oplus M_1 = \Delta$.

Obtaining MACs: The Functionality

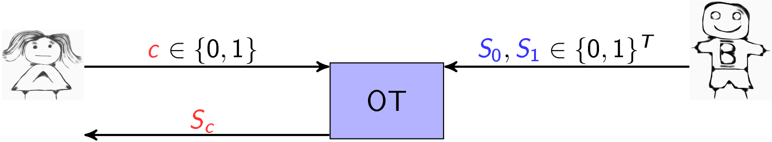


Obtaining MACs: Protocol Steps

- Step 1: Obtain a *few*, *long* MACs on Alice's random bits.
- Step 2: Turn into *many*, *short* MACs on Bob's random bits.

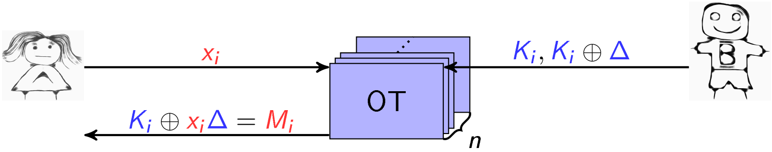
Step 1: Long MACs for Alice

To authenticate bits $x_1, x_2, \dots, x_n$:



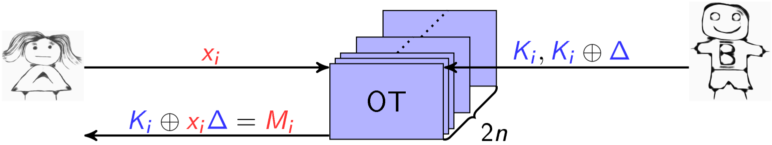
Step 1: Long MACs for Alice

To authenticate bits $x_1, x_2, \ldots, x_n$:



Step 1: Long MACs for Alice

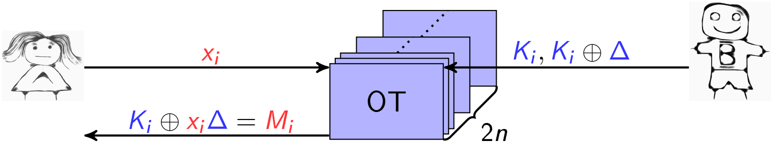
To authenticate bits $x_1, x_2, \ldots, x_n$:



- Problem: Bob must use same Δ in every OT.

Step 1: Long MACs for Alice

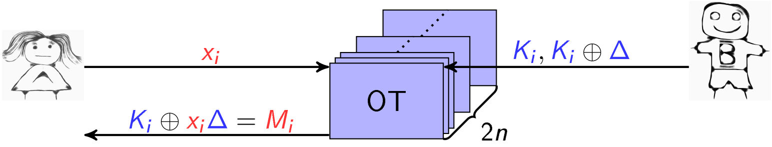
To authenticate bits $x_1, x_2, \ldots, x_n$:



- Problem: Bob must use same Δ in every OT.
- Solution:
  - Do $2n$ OTs.

Step 1: Long MACs for Alice

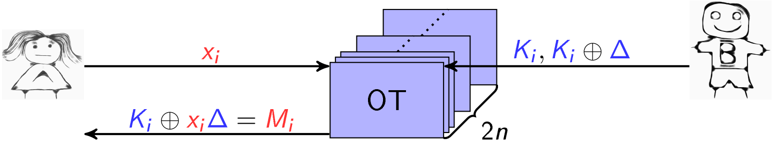
To authenticate bits $x_1, x_2, \ldots, x_n$:



- Problem: Bob must use same Δ in every OT.
- Solution:
  - Do $2n$ OTs.
  - Force Bob to use consistent Δ, using a "cut-n-chose"-like technique.

Step 1: Long MACs for Alice

To authenticate bits $x_1, x_2, \ldots, x_n$:



- Problem: Bob must use same Δ in every OT.
- Solution:
  - Do $2n$ OTs.
  - Force Bob to use consistent Δ, using a "cut-n-chose"-like technique.
  - Sacrifice half of the authenticated messages.

Step 2: Short MACs For Bob

$$M_1 \; M_2 \; \cdots \; M_n = K_1 \; K_2 \; \cdots \; K_n \oplus x_1\Delta \; x_2\Delta \; \cdots \; x_n\Delta$$

Step 2: Short MACs For Bob

$$K_1 \quad K_2 \quad \cdots \quad K_n \;=\; M_1 \quad M_2 \quad \cdots \quad M_n \;\oplus\; x_1\Delta \quad x_2\Delta \quad \cdots \quad x_n\Delta$$

Step 2: Short MACs For Bob

$$\begin{array}{c} (K_{1,1},\dots,K_{1,n}) \\ (K_{2,1},\dots,K_{2,n}) \\ \vdots \\ (K_{T,1},\dots,K_{T,n}) \end{array} = \begin{array}{c} (M_{1,1},\dots,M_{1,n}) \\ (M_{2,1},\dots,M_{2,n}) \\ \vdots \\ (M_{T,1},\dots,M_{T,n}) \end{array} \oplus \begin{array}{c} (x_1,\dots,x_n)\Delta_1 \\ (x_1,\dots,x_n)\Delta_2 \\ \vdots \\ (x_1,\dots,x_n)\Delta_T \end{array}$$

Step 2: Short MACs For Bob

$$\begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_T \end{bmatrix} = \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_T \end{bmatrix} \oplus \begin{bmatrix} (x_1, \ldots, x_n)\Delta_1 \\ (x_1, \ldots, x_n)\Delta_2 \\ \vdots \\ (x_1, \ldots, x_n)\Delta_T \end{bmatrix}$$

Step 2: Short MACs For Bob

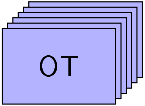
$$\begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_T \end{bmatrix} = \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_T \end{bmatrix} \oplus \begin{bmatrix} \Gamma\Delta_1 \\ \Gamma\Delta_2 \\ \vdots \\ \Gamma\Delta_T \end{bmatrix}$$

Step 2: Short MACs For Bob

$$\begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_T \end{bmatrix} = \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_T \end{bmatrix} \oplus \begin{bmatrix} \Gamma y_1 \\ \Gamma y_2 \\ \vdots \\ \Gamma y_T \end{bmatrix}$$

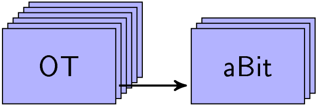
- $N_i = L_i \oplus y_i \Gamma$, i.e. N_i is a MAC on y_i w. keys L_i, Γ.

Obtaining MACs: Summary



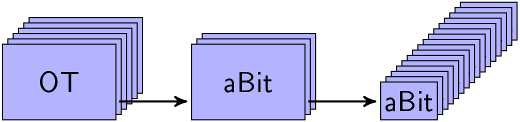
- A *few* ($2n$) OTs with *long* messages ($T = \text{poly}(n)$ bits).

Obtaining MACs: Summary



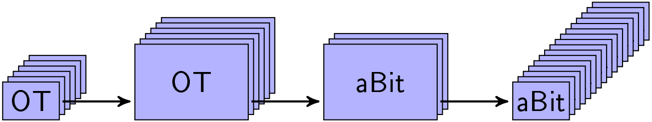
- A *few* ($2n$) OTs with *long* messages ($T = \text{poly}(n)$ bits).
- A few (n) long (T bits) MACs for Alice.

Obtaining MACs: Summary



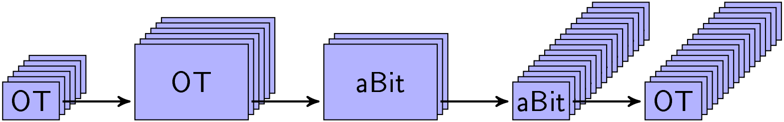
- A *few* $(2n)$ OTs with *long* messages ($T = \text{poly}(n)$ bits).
- A few (n) long (T bits) MACs for Alice.
- Many (T) *short* (n bits) MACs for Bob.

Obtaining MACs: Summary



- A *few* ($2n$) OTs with *long* messages ($T = \text{poly}(n)$ bits).
- A few (n) long (T bits) MACs for Alice.
- Many (T) *short* (n bits) MACs for Bob.
- Note 1: Can get long OTs from short OT using a PRG.

Obtaining MACs: Summary



- A *few* ($2n$) OTs with *long* messages ($T = \text{poly}(n)$ bits).
- A few (n) long (T bits) MACs for Alice.
- Many (T) *short* (n bits) MACs for Bob.
- Note 1: Can get long OTs from short OT using a PRG.
- Note 2: Can get short OT from short aBit (i.e. OT-extension).

Overview

Protocol Overview

MACs

Concluding

Concluding . . .

Take away:

- Finally a *non-Garbled Circuits* approach do practical 2PC!

Concluding . . .

Take away:

- Finally a *non-Garbled Circuits* approach do practical 2PC!
- It's based on GMW and OT-extension.

Concluding . . .

Take away:

- Finally a *non-Garbled Circuits* approach do practical 2PC!
- It's based on GMW and OT-extension.
- It's really fast!

Concluding ...

Take away:

- Finally a *non-Garbled Circuits* approach do practical 2PC!
- It's based on GMW and OT-extension.
- It's really fast!
- ... So if you're implementing a 2PC protocol, why not give this a try?

Concluding . . .

Take away:

- Finally a *non-Garbled Circuits* approach do practical 2PC!
- It's based on GMW and OT-extension.
- It's really fast!
- . . . So if you're implementing a 2PC protocol, why not give this a try?

Thank you.