Reusable Secure Computation in the Plain Model

Vipul Goyal

NTT Research & CMU

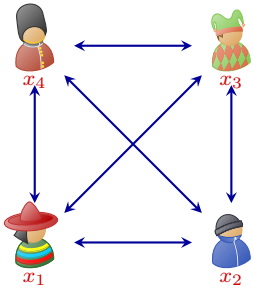
Akshayaram Srinivasan

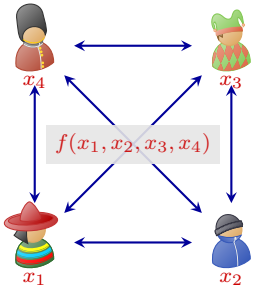
TIFR –> UToronto

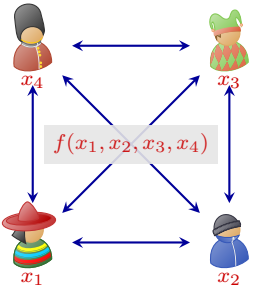
Mingyuan Wang

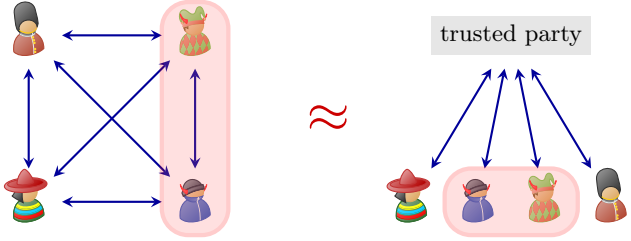
UC Berkeley

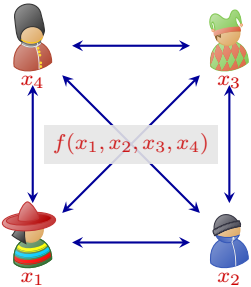
Aug 2023 @ Crypto'23


x_4
x_3
x_1
x_2


x_4
x_3
$f(x_1, x_2, x_3, x_4)$
x_1
x_2

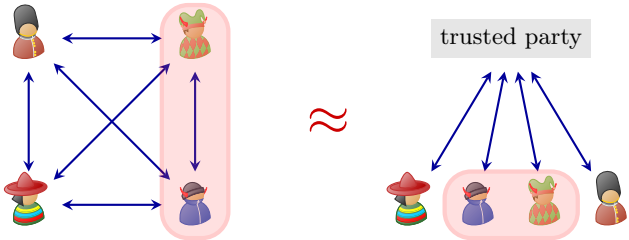

x_4
x_3
$f(x_1, x_2, x_3, x_4)$
x_1
x_2

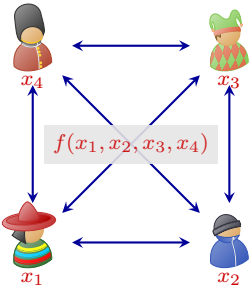

trusted party

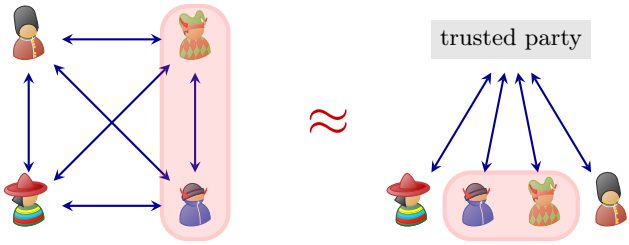


Secure Multiparty Computation

- plain model
- dishonest majority
- malicious security (black-box simulation)
- polynomial-time simulator







Secure Multiparty Computation

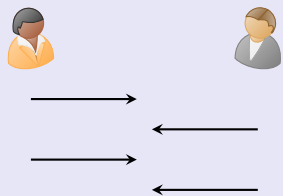
- plain model
- dishonest majority
- malicious security (black-box simulation)
- polynomial-time simulator

Objective

Construct round-optimal protocol.

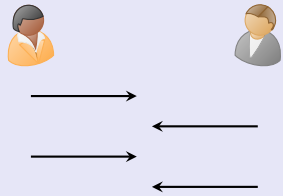
Lower bound

- 2PC unidirectional message
- 4 rounds [Katz-Ostrovsky'04]

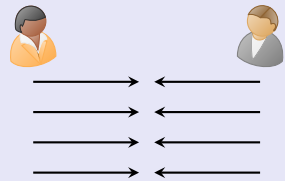


Lower bound

- 2PC unidirectional message
- 4 rounds [Katz-Ostrovsky'04]

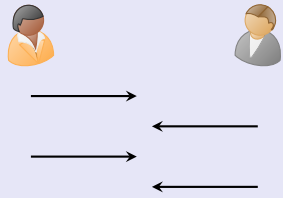


- MPC Simultaneous message
- 4 rounds
  [Garg-Mukherjee-Pandey-Polychroniadou'16]

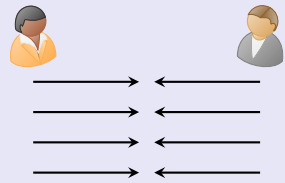


Lower bound

- 2PC unidirectional message
- 4 rounds [Katz-Ostrovsky'04]



- MPC Simultaneous message
- 4 rounds
  [Garg-Mukherjee-Pandey-Polychroniadou'16]



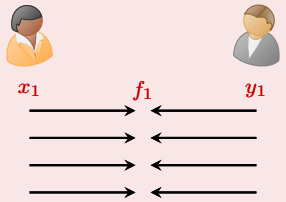
Upper bound

[Yao86, BMR90, KO04, IPS08, IKOPS11, ORS15, GMPP16, BHP17, ACJ17, BL18, GS18, BGJKKS18, HHPV18, FMV19, CCGJO20]

Matching upper bound with minimal assumption (4-round OT)

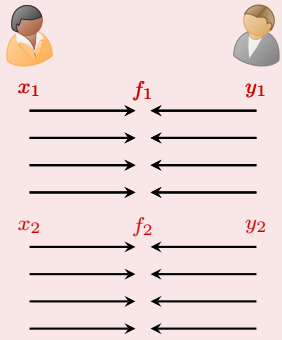
Reusable Setting

Suppose Alice and Bob want to continuously evaluate multiple functions $f_1(x_1, y_1),\ f_2(x_2, y_2), \ldots$



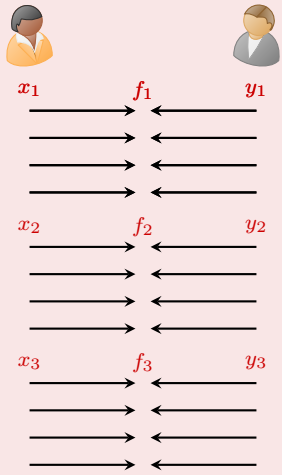
Reusable Setting

Suppose Alice and Bob want to continuously evaluate multiple functions $f_1(x_1, y_1),\ f_2(x_2, y_2), \ldots$



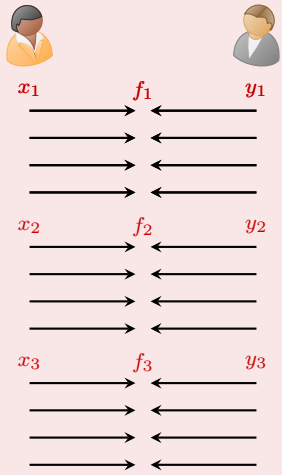
Reusable Setting

Suppose Alice and Bob want to continuously evaluate multiple functions $f_1(x_1, y_1),\ f_2(x_2, y_2), \ldots$



Reusable Setting

Suppose Alice and Bob want to continuously evaluate multiple functions $f_1(x_1, y_1),\ f_2(x_2, y_2), \ldots$

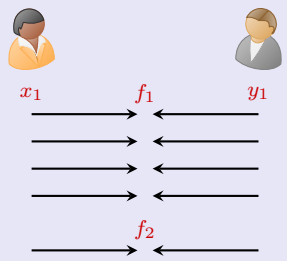


Can we reuse the previous interactions to reduce the number of rounds?

Optimal Round Complexity for Different Modes of Reusability

if both inputs and function change

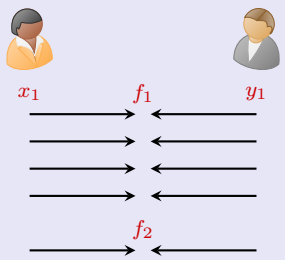
if only the function changes



Optimal Round Complexity for Different Modes of Reusability

if both inputs and function change

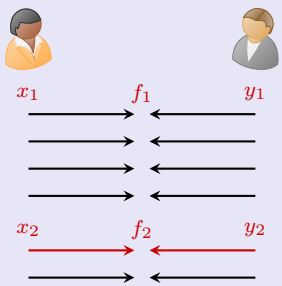
if only the function changes



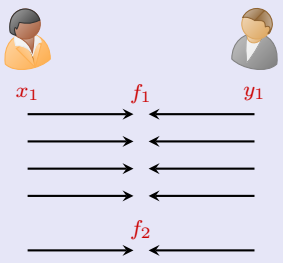
The first three rounds can be reused!

Optimal Round Complexity for Different Modes of Reusability

if both inputs and function change



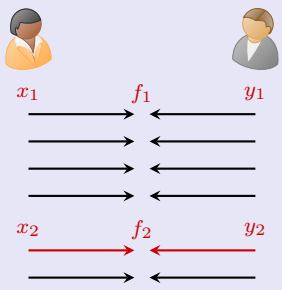
if only the function changes



The first three rounds can be reused!

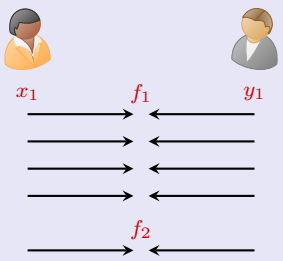
Optimal Round Complexity for Different Modes of Reusability

if both inputs and function change



Residual attack if only one round of interaction.

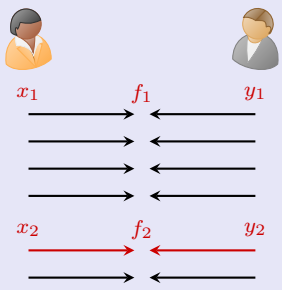
if only the function changes



The first three rounds can be reused!

Optimal Round Complexity for Different Modes of Reusability

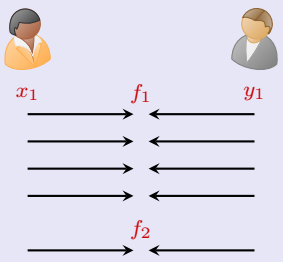
if both inputs and function change



Residual attack if only one round of interaction.

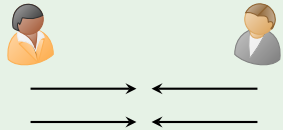
The first two rounds can be reused!

if only the function changes

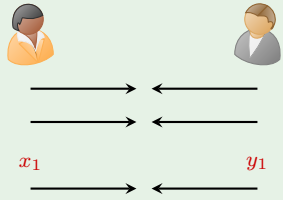


The first three rounds can be reused!

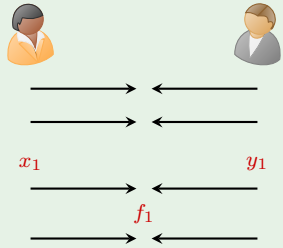
Our Model for Round-optimal Reusable Secure Computation



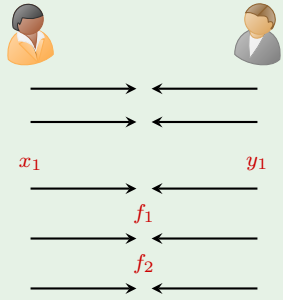
Our Model for Round-optimal Reusable Secure Computation



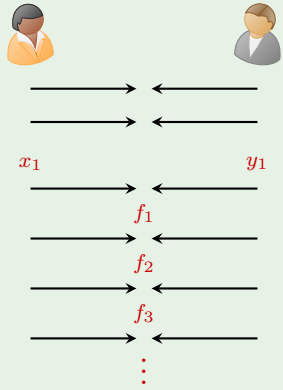
Our Model for Round-optimal Reusable Secure Computation



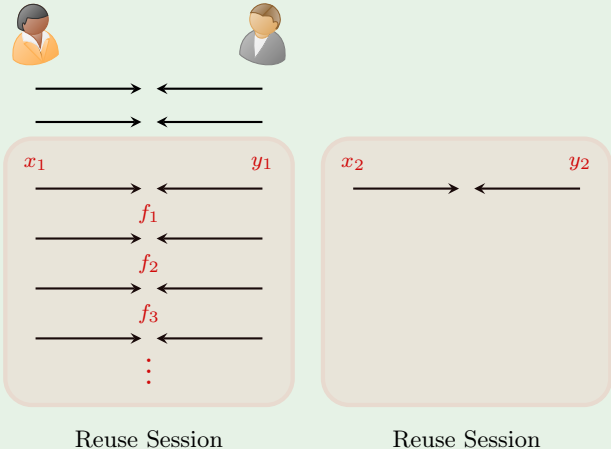
Our Model for Round-optimal Reusable Secure Computation



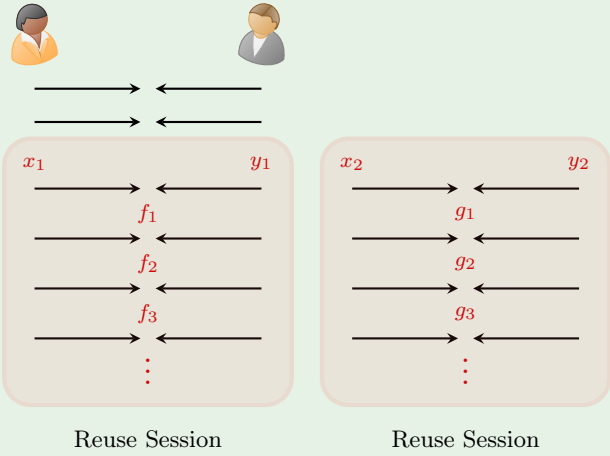
Our Model for Round-optimal Reusable Secure Computation



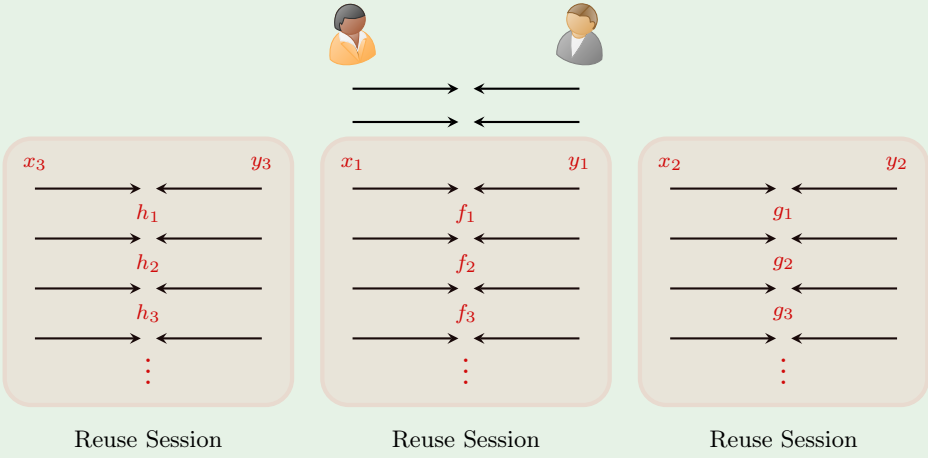
Our Model for Round-optimal Reusable Secure Computation



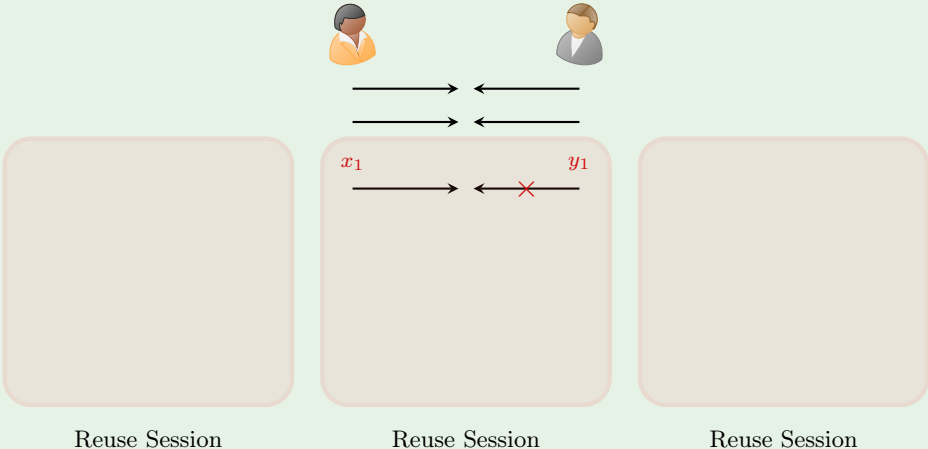
Our Model for Round-optimal Reusable Secure Computation



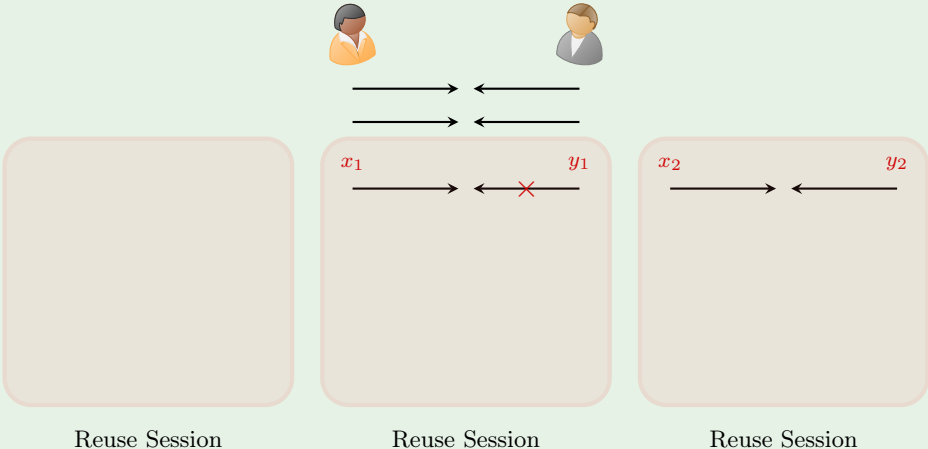
Our Model for Round-optimal Reusable Secure Computation



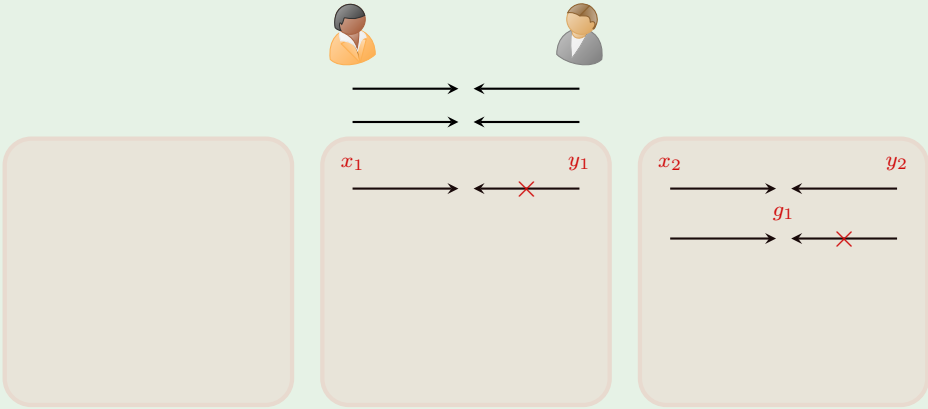
Our Model for Round-optimal Reusable Secure Computation



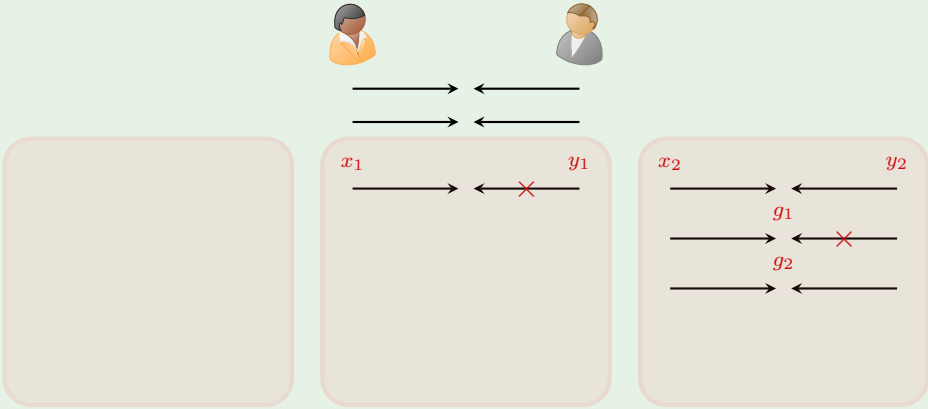
Our Model for Round-optimal Reusable Secure Computation



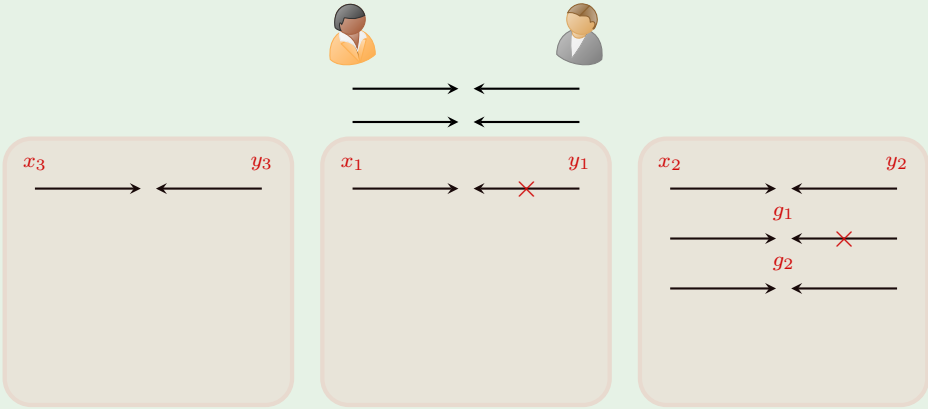
Our Model for Round-optimal Reusable Secure Computation



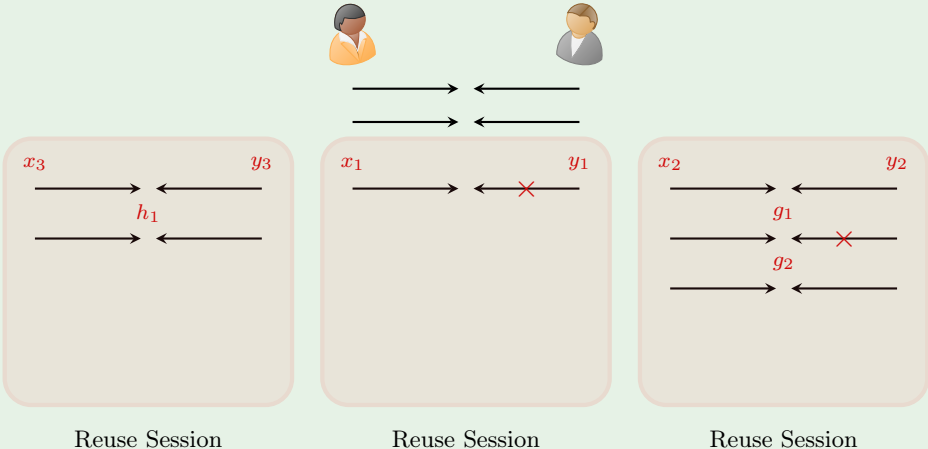
Our Model for Round-optimal Reusable Secure Computation



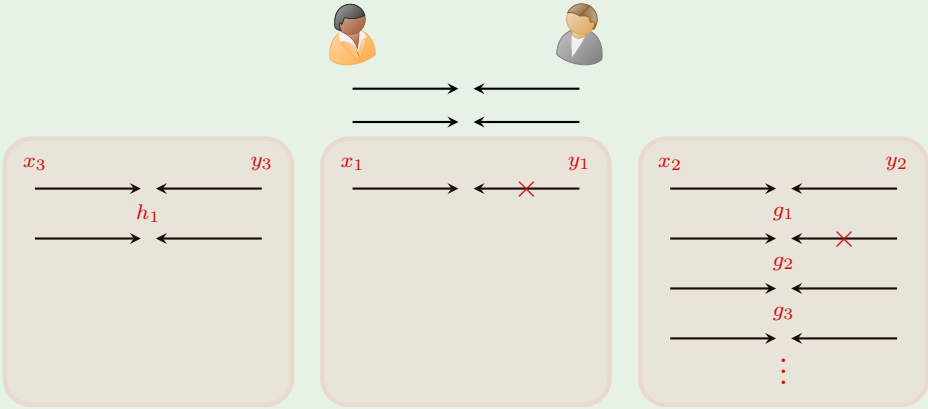
Our Model for Round-optimal Reusable Secure Computation



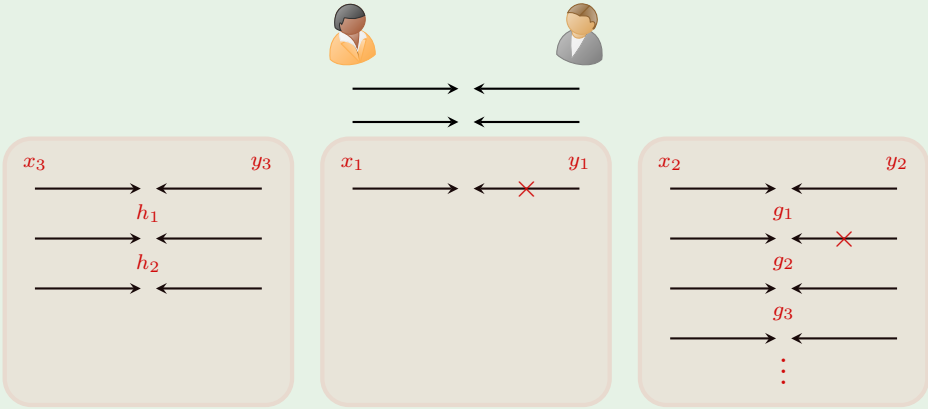
Our Model for Round-optimal Reusable Secure Computation



Our Model for Round-optimal Reusable Secure Computation



Our Model for Round-optimal Reusable Secure Computation



Related Work — Reusable MPC with trusted setup

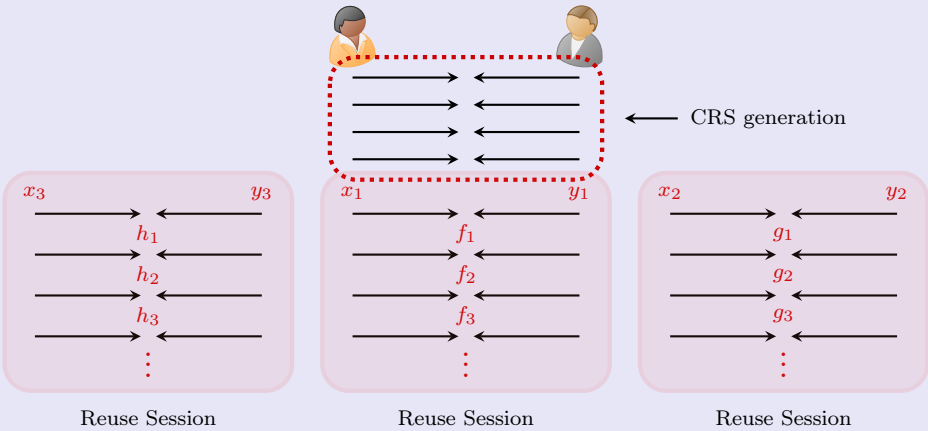
[Benhamouda-Lin'20, Bartusek-Garg-Masny-Mukherjee'20, Ananth-Jain-Jin-Malavolta'21'22, Benhamouda-Jain-Komargodski-Lin'21, Bartusek-Garg-Srinivasan-Zhang'22]

- In the CRS model
- Two-round malicious-secure MPC protocol
- First round can be reused

Related Work — Reusable MPC with trusted setup

[Benhamouda-Lin'20, Bartusek-Garg-Masny-Mukherjee'20, Ananth-Jain-Jin-Malavolta'21'22, Benhamouda-Jain-Komargodski-Lin'21, Bartusek-Garg-Srinivasan-Zhang'22]

- In the CRS model
- Two-round malicious-secure MPC protocol
- First round can be reused



Related Work

Reusable MPC in the plain model

[Fernando-Jain-Komargodski'23]

- Plain model
- Two rounds where the first round can be reused
- Super-polynomial-time Simulator

Our Results

Reusable 2PC

(DDH or QR) + ZAP ===> Four-round Reusable 2PC

ZAP

[Dwork-Naor'00] Two-round public coin witness indistinguishable proof.

Our Results

Reusable 2PC

(DDH or QR) + ZAP ===> Four-round Reusable 2PC

ZAP

[Dwork-Naor'00] Two-round public coin witness indistinguishable proof.

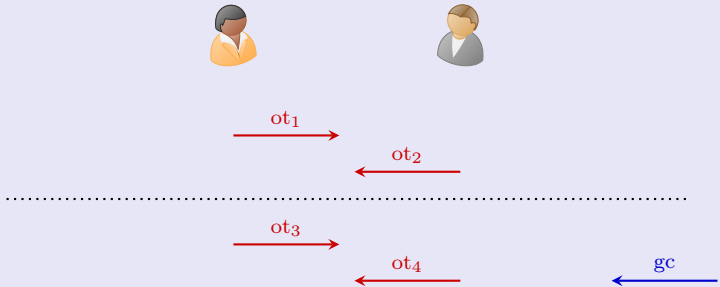
Reusable MPC

Four-round Reusable 2PC/OT + (semi-malicious) Two-round Reusable MPC + ZAP ===> Four-round Reusable MPC

Semi-malicious Two-round Reusable MPC

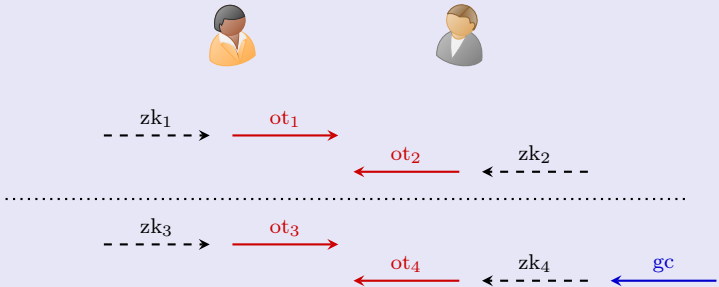
DDH [BGMM20], pairing [BL20], LWE [AJJM20, AJJM21, BJKL21], LPN [BGSZ22]

Technical Challenges: 2PC



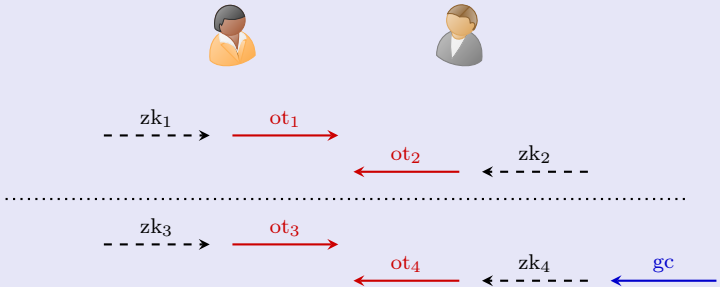
- GC: garbled circuit
- OT: four-round oblivious transfer (simulation security for malicious receiver; indistinguishability-based security for malicious senders)

Technical Challenges: 2PC



- GC: garbled circuit
- OT: four-round oblivious transfer (simulation security for malicious receiver; indistinguishability-based security for malicious senders)
- ZK: zero-knowledge protocol (proof of knowledge)

Technical Challenges: 2PC

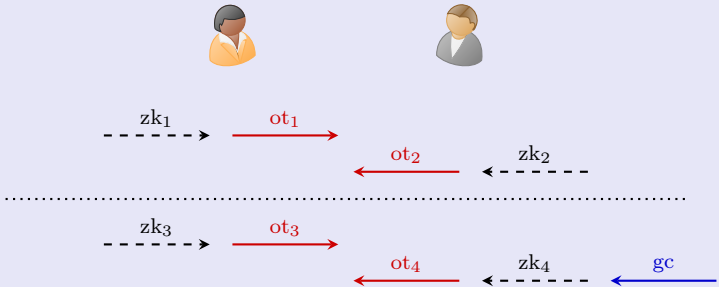


- GC: garbled circuit
- OT: four-round oblivious transfer (simulation security for malicious receiver; indistinguishability-based security for malicious senders)
- ZK: zero-knowledge protocol (proof of knowledge)

Issue

ZK and OT need to be reusably secure!

Technical Challenges: 2PC



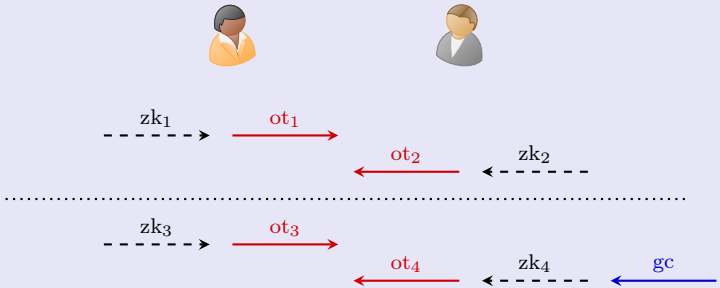
- GC: garbled circuit
- OT: four-round oblivious transfer (simulation security for malicious receiver; indistinguishability-based security for malicious senders)
- ZK: zero-knowledge protocol (proof of knowledge)

Issue

ZK and OT need to be reusably secure!

- A four-round OT that both the receiver and the sender may *change input*.

Technical Challenges: 2PC



- GC: garbled circuit
- OT: four-round oblivious transfer (simulation security for malicious receiver; indistinguishability-based security for malicious senders)
- ZK: zero-knowledge protocol (proof of knowledge)

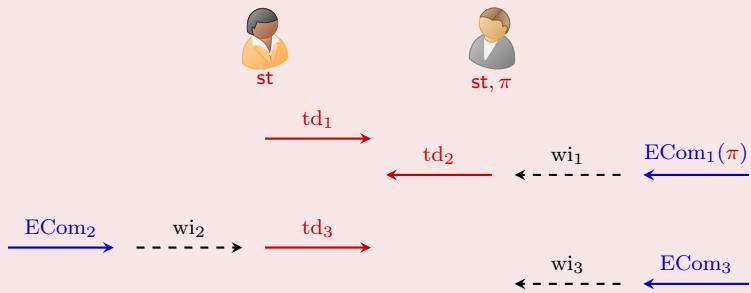
Issue

ZK and OT need to be reusably secure!

- A four-round OT that both the receiver and the sender may *change input*.
- A four-round ZK that the prover may send multiple fourth-round messages proving different statements

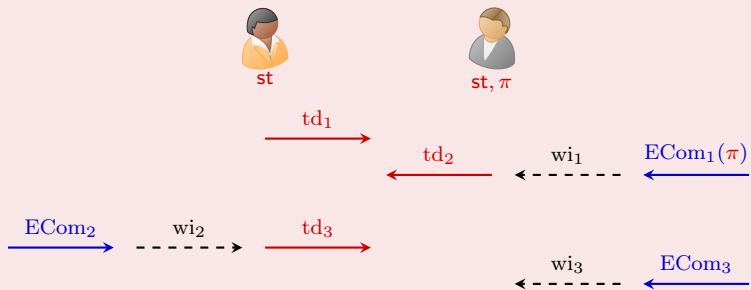
Reusable Zero-knowledge [Feige-Lapidot-Shamir'90]

- TD: trapdoor generation protocol
- ECom: extractable commitment scheme
- WI: witness indistinguishable proof



Reusable Zero-knowledge [Feige-Lapidot-Shamir'90]

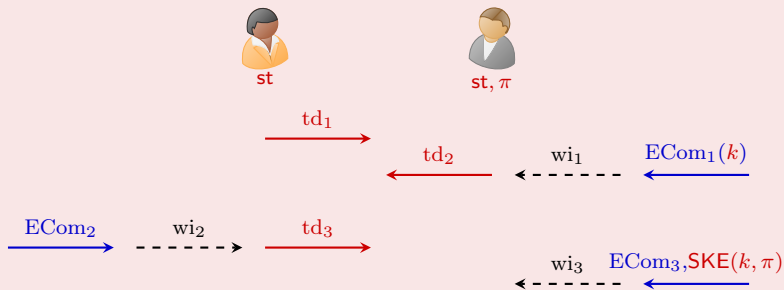
- TD: trapdoor generation protocol
- ECom: extractable commitment scheme
- WI: witness indistinguishable proof



- Extractable commitment needs to be reusable and delayed-input. Use symmetric-key encryption!

Reusable Zero-knowledge [Feige-Lapidot-Shamir'90]

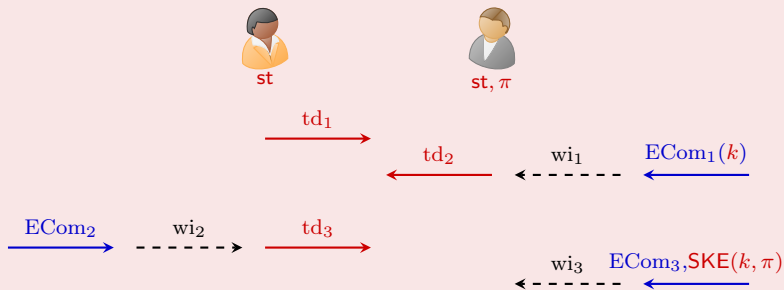
- TD: trapdoor generation protocol
- ECom: extractable commitment scheme
- WI: witness indistinguishable proof



- Extractable commitment needs to be reusable and delayed-input. Use symmetric-key encryption!

Reusable Zero-knowledge [Feige-Lapidot-Shamir'90]

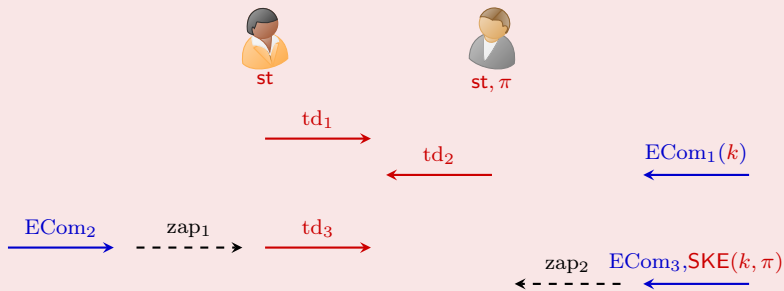
- TD: trapdoor generation protocol
- ECom: extractable commitment scheme
- WI: witness indistinguishable proof



- Extractable commitment needs to be reusable and delayed-input. Use symmetric-key encryption!
- Witness indistinguishable proof needs to be reusable. Use ZAP!

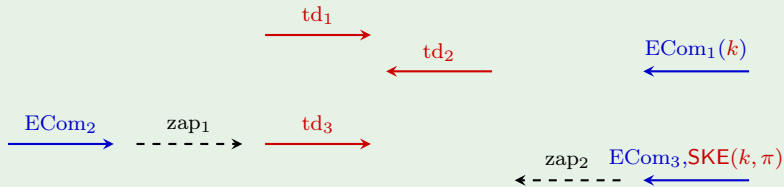
Reusable Zero-knowledge [Feige-Lapidot-Shamir'90]

- TD: trapdoor generation protocol
- ECom: extractable commitment scheme
- WI: witness indistinguishable proof

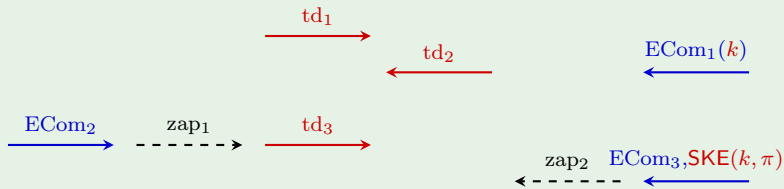


- Extractable commitment needs to be reusable and delayed-input. Use symmetric-key encryption!
- Witness indistinguishable proof needs to be reusable. Use ZAP!

Reusable Zero-knowledge

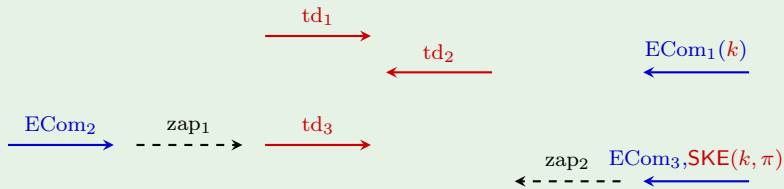


Reusable Zero-knowledge



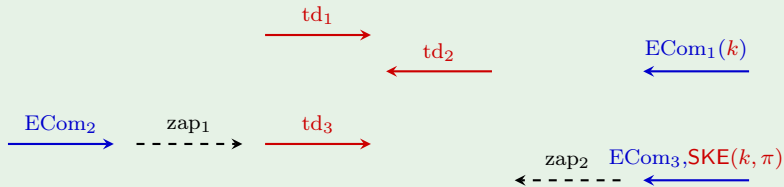
- ZK only require OWF; Our r-ZK requires ZAP
  - inevitably need $>$ OWF assumption due to state-of-the-art; implies preprocessing NIZK

Reusable Zero-knowledge



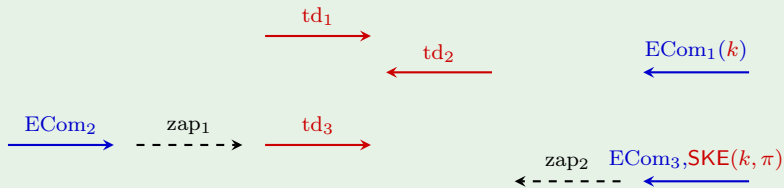
- ZK only require OWF; Our r-ZK requires ZAP
  - inevitably need $>$ OWF assumption due to state-of-the-art; implies preprocessing NIZK
- Only three-round reusable, not two-round reusable

Reusable Zero-knowledge



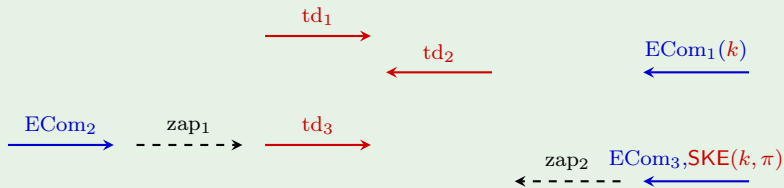
- ZK only require OWF; Our r-ZK requires ZAP
  - inevitably need $>$ OWF assumption due to state-of-the-art; implies preprocessing NIZK
- Only three-round reusable, not two-round reusable
  - Inherent in unidirectional message model;

Reusable Zero-knowledge



- ZK only require OWF; Our r-ZK requires ZAP
  - inevitably need $>$ OWF assumption due to state-of-the-art; implies preprocessing NIZK
- Only three-round reusable, not two-round reusable
  - Inherent in unidirectional message model;
  - For 2PC, Bob needs to keep a secret state across different reuse sessions to check the consistency of the third-round message

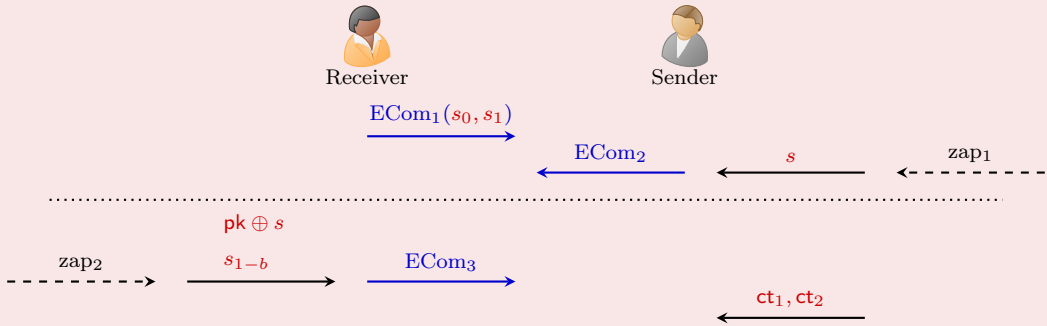
Reusable Zero-knowledge



- ZK only require OWF; Our r-ZK requires ZAP
  - inevitably need $>$ OWF assumption due to state-of-the-art; implies preprocessing NIZK
- Only three-round reusable, not two-round reusable
  - Inherent in unidirectional message model;
  - For 2PC, Bob needs to keep a secret state across different reuse sessions to check the consistency of the third-round message
  - not an issue in simultaneous message model/MPC

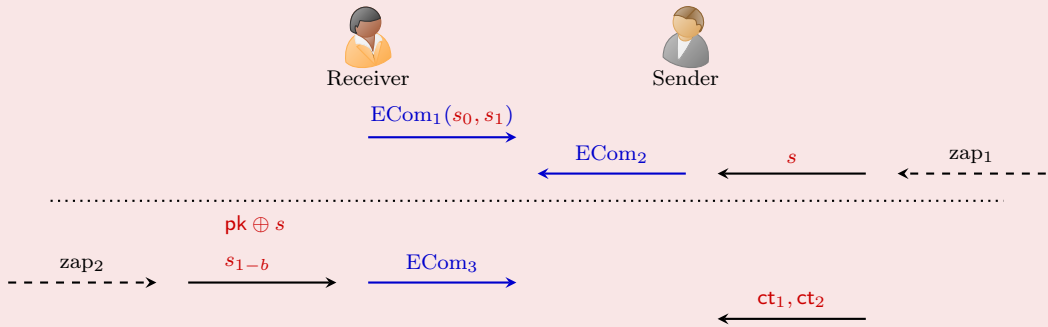
Reusable OT: both sender and receiver may change input

[Katz-Ostrovsky'04,Ostrovsky-Richelson-Scafuro'15, Friolo-Masny-Venturi'19]



Reusable OT: both sender and receiver may change input

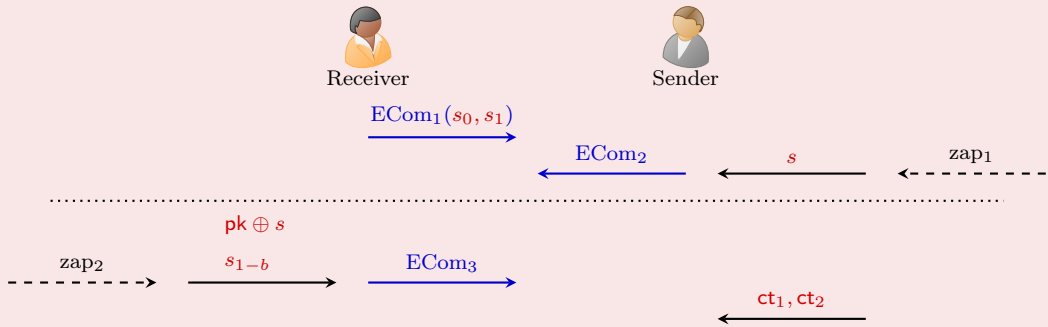
[Katz-Ostrovsky'04,Ostrovsky-Richelson-Scafuro'15, Friolo-Masny-Venturi'19]



- Sender's reusability comes for free. Use symmetric-key encryption

Reusable OT: both sender and receiver may change input

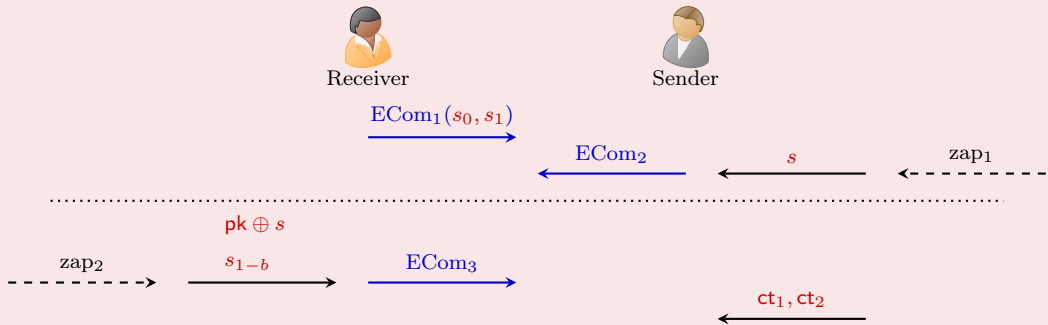
[Katz-Ostrovsky'04,Ostrovsky-Richelson-Scafuro'15, Friolo-Masny-Venturi'19]



- Sender's reusability comes for free. Use symmetric-key encryption
- Receiver reusability: not reusably secure

Reusable OT: both sender and receiver may change input

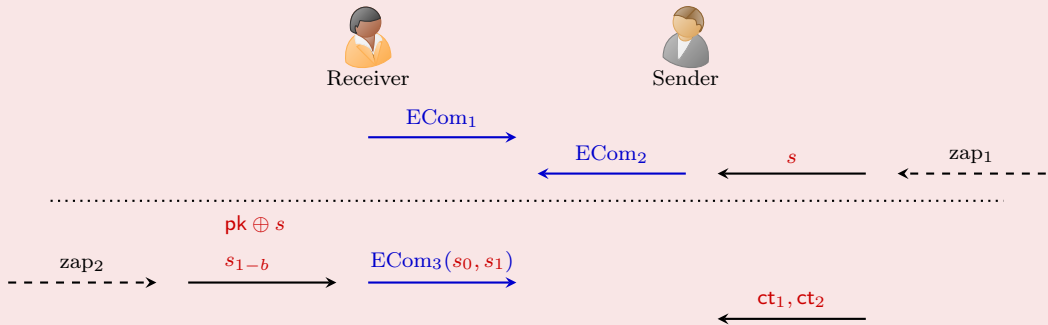
[Katz-Ostrovsky'04,Ostrovsky-Richelson-Scafuro'15, Friolo-Masny-Venturi'19]



- Sender's reusability comes for free. Use symmetric-key encryption
- Receiver reusability: not reusably secure
  - If ECom is not delayed-input: insecure for the sender both s_0, s_1 will be leaked

Reusable OT: both sender and receiver may change input

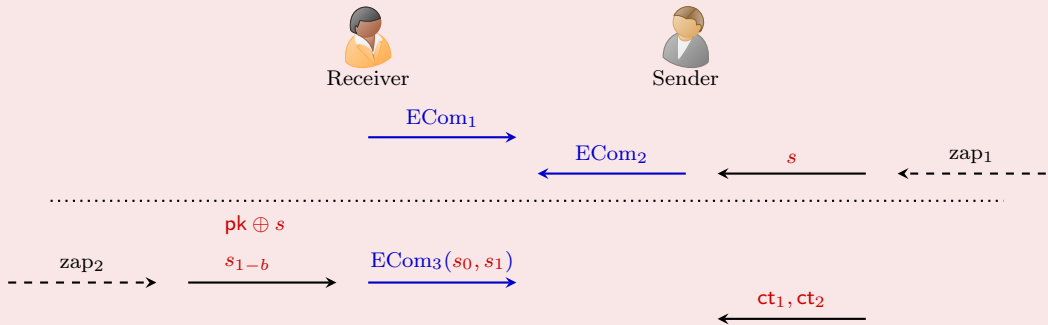
[Katz-Ostrovsky'04,Ostrovsky-Richelson-Scafuro'15, Friolo-Masny-Venturi'19]



- Sender's reusability comes for free. Use symmetric-key encryption
- Receiver reusability: not reusably secure
  - If ECom is not delayed-input: insecure for the sender both s_0, s_1 will be leaked
  - If ECom is delayed-input: insecure for the receiver pick s_{1-b} maliciously

Reusable OT: both sender and receiver may change input

[Katz-Ostrovsky'04,Ostrovsky-Richelson-Scafuro'15, Friolo-Masny-Venturi'19]



- Sender's reusability comes for free. Use symmetric-key encryption
- Receiver reusability: not reusably secure
  - If ECom is not delayed-input: insecure for the sender both s_0, s_1 will be leaked
  - If ECom is delayed-input: insecure for the receiver pick s_{1-b} maliciously
- Need to reconcile between giving the receiver too much freedom and too little freedom

Naor's bit commitment scheme [Naor'91]

s

$\begin{cases} b = 0 & \mathsf{PRG}(\mathsf{sd}) \\ b = 1 & \mathsf{PRG}(\mathsf{sd}) \oplus s \end{cases}$

sd

- Insecure (equivocal) if $\mathsf{PRG}(\mathsf{sd}) \oplus s = \mathsf{PRG}(\mathsf{sd}')$

Naor's bit commitment scheme [Naor'91]

s

$\begin{cases} b = 0 & \mathsf{PRG}(\mathsf{sd}) \\ b = 1 & \mathsf{PRG}(\mathsf{sd}) \oplus s \end{cases}$

sd

- Insecure (equivocal) if $\mathsf{PRG}(\mathsf{sd}) \oplus s = \mathsf{PRG}(\mathsf{sd}')$
- Most choices of s satisfies $s \neq \mathsf{PRG}(\mathsf{sd}) \oplus \mathsf{PRG}(\mathsf{sd}')$

Naor's bit commitment scheme [Naor'91]

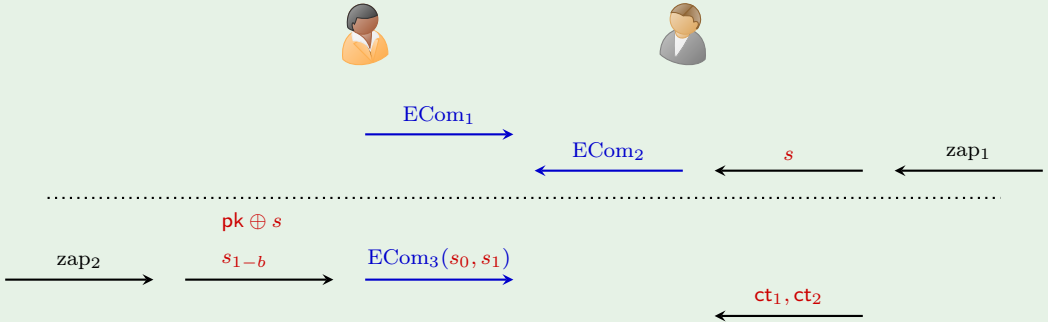
s

$\begin{cases} b = 0 & \mathsf{PRG}(\mathsf{sd}) \\ b = 1 & \mathsf{PRG}(\mathsf{sd}) \oplus s \end{cases}$

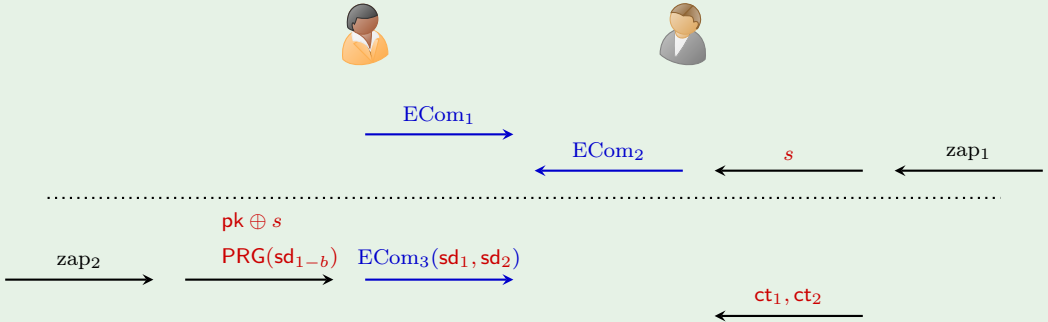
sd

- Insecure (equivocal) if $\mathsf{PRG}(\mathsf{sd}) \oplus s = \mathsf{PRG}(\mathsf{sd}')$
- Most choices of s satisfies $s \neq \mathsf{PRG}(\mathsf{sd}) \oplus \mathsf{PRG}(\mathsf{sd}')$
- If $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$, $2^\lambda \times 2^\lambda$ choices of $\mathsf{PRG}(\mathsf{sd}) \oplus \mathsf{PRG}(\mathsf{sd}')$ and $2^{3\lambda}$ choices of s.

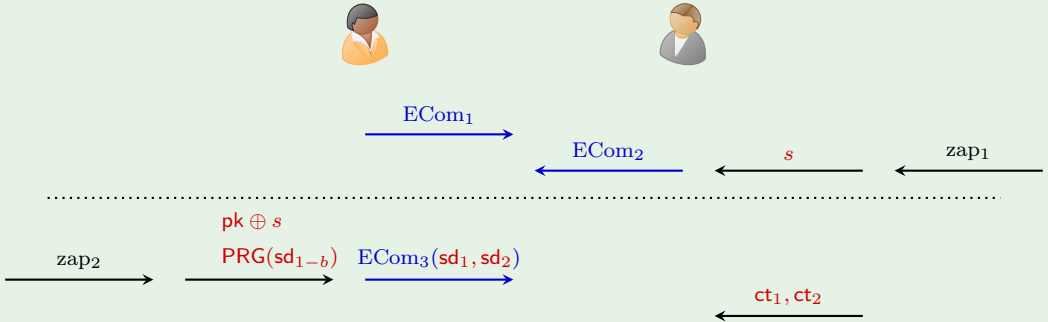
Reusable OT


ECom_1
ECom_2
s
zap_1
$\mathsf{pk} \oplus s$
zap_2
s_{1-b}
$\mathsf{ECom}_3(s_0, s_1)$
$\mathsf{ct}_1, \mathsf{ct}_2$

Reusable OT

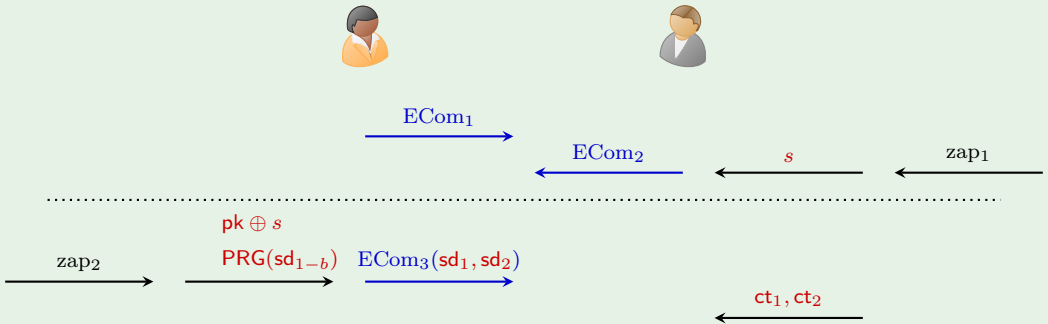

ECom_1
ECom_2
s
zap_1
$\mathsf{pk} \oplus s$
zap_2
$\mathsf{PRG}(\mathsf{sd}_{1-b})$
$\mathrm{ECom}_3(\mathsf{sd}_1, \mathsf{sd}_2)$
$\mathsf{ct}_1, \mathsf{ct}_2$

Reusable OT



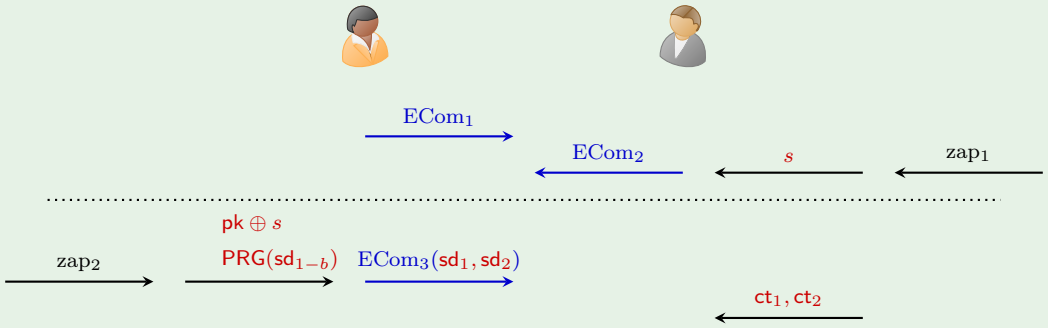
- Insecure if $\mathsf{pk} = s \oplus \mathsf{PRG}(\mathsf{sd}_{1-b})$ is some valid public key.

Reusable OT



- Insecure if $\mathsf{pk} = s \oplus \mathsf{PRG}(\mathsf{sd}_{1-b})$ is some valid public key.
- We need that: most s are "good", i.e., $\neq \mathsf{PRG}(\mathsf{sd}) \oplus \mathsf{pk}$.

Reusable OT



- Insecure if $\mathsf{pk} = s \oplus \mathsf{PRG}(\mathsf{sd}_{1-b})$ is some valid public key.
- We need that: most s are "good", i.e., $\neq \mathsf{PRG}(\mathsf{sd}) \oplus \mathsf{pk}$.
- A special kind of PKE
  - pseudorandom public key
  - valid public keys are scarce
  - maliciously chosen invalid public keys still hide the message

1. pseudorandom public key
2. valid public keys are scarce
3. maliciously chosen invalid public keys still hide the message

1. pseudorandom public key
2. valid public keys are scarce
3. maliciously chosen invalid public keys still hide the message

Special PKE scheme from DDH

- Public key domain $\begin{pmatrix} g & g^a \\ g^b & g^c \end{pmatrix}$.
- Valid public key $c = ab$, invalid public key $c \neq ab$
- Encryption

$$(u, v) \cdot \begin{pmatrix} g & g^a \\ g^b & g^c \end{pmatrix} \oplus (0, \mathsf{msg})$$

1. pseudorandom public key
2. valid public keys are scarce
3. maliciously chosen invalid public keys still hide the message

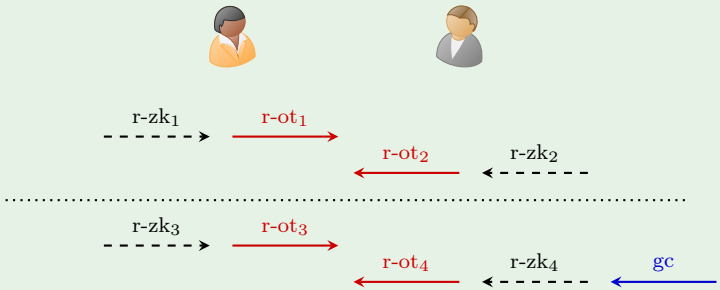
Special PKE scheme from DDH

- Public key domain $\begin{pmatrix} g & g^a \\ g^b & g^c \end{pmatrix}$.
- Valid public key $c = ab$, invalid public key $c \neq ab$
- Encryption

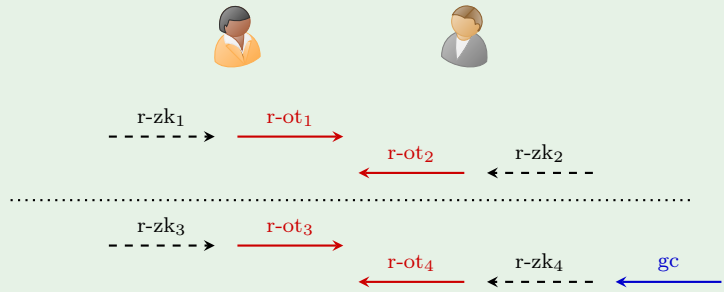
$$(u, v) \cdot \begin{pmatrix} g & g^a \\ g^b & g^c \end{pmatrix} \oplus (0, \mathsf{msg})$$

We also show how to construct it from SSP-OT and QR.

Reusable 2PC



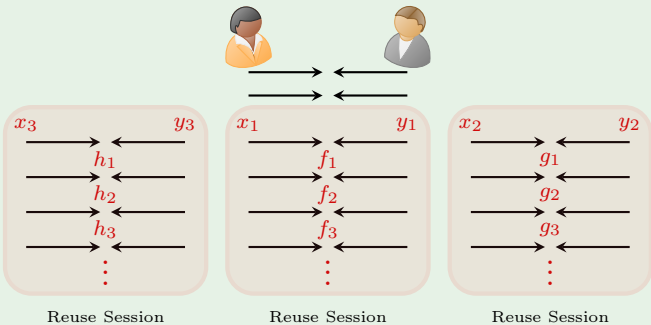
Reusable 2PC



Reusable MPC

- Based on appropriate adaptations of [Choudhuri-Ciampi-Goyal-Jain-Ostrovsky'20]
- Replace OT and ZK with our r-OT and r-ZK
- Additionally, we need two-round reusable semi-malicious MPC

Summary



- Reusable 2PC from
  - DDH or QR
  - ZAP
- Reusable MPC from
  - Reusable 2pc
  - (semi-malicious) two-round reusable MPC
  - ZAP

Thanks! Questions? ia.cr/2023/1006