Efficient and Universally Composable Committed Oblivious Transfer and Applications

Juan Garay	Bell Labs, Lucent Technologies
Philip MacKenzie	Bell Labs, Lucent Technologies
Ke Yang	Carnegie Mellon University

Oblivious Transfer (OT)

BEFORE: S has input (x_0, x_1), R has input b.
AFTER: S learns nothing, R learns x_b.

Oblivious Transfer (OT)

BEFORE: S has input (x_0, x_1), R has input b.
AFTER: S learns nothing, R learns x_b.

- "original" definition [Rabin '81]

Oblivious Transfer (OT)

BEFORE: S has input (x_0, x_1), R has input b.
AFTER: S learns nothing, R learns x_b.

- "original" definition [Rabin '81]
- $\binom{2}{1}$-OT definition [Even-Goldreich-Lempel '85]

Oblivious Transfer (OT)

BEFORE: S has input (x_0, x_1), R has input b.
AFTER: S learns nothing, R learns x_b.

- "original" definition [Rabin '81]
- $\binom{2}{1}$-OT definition [Even-Goldreich-Lempel '85]
- Two are equivalent [Crépeau '87]

Oblivious Transfer (OT)

BEFORE: S has input (x_0, x_1), R has input b.
AFTER: S learns nothing, R learns x_b.

- "original" definition [Rabin '81]
- $\binom{2}{1}$-OT definition [Even-Goldreich-Lempel '85]
- Two are equivalent [Crépeau '87]
- We use the $\binom{2}{1}$-OT version.

OT as a building block

Oblivious Transfer is an important building block for Multi-Party Computation (MPC)

OT as a building block

Oblivious Transfer is an important building block for Multi-Party Computation (MPC)

- [Yao ’82] use OT (and other techniques) to solve 2-PC

OT as a building block

Oblivious Transfer is an important building block for Multi-Party Computation (MPC)

- [Yao ’82] use OT (and other techniques) to solve 2-PC
- [Goldreich-Micali-Wigderson ’87] use OT to solve MPC
  OT is complete for honest-but-curious adversaries.
  Gate-by-gate approach, reasonably efficient for simple circuits.
  For malicious adversaries, needs commitments and ZK proofs.

OT as a building block

Oblivious Transfer is an important building block for Multi-Party Computation (MPC)

- [Yao ’82] use OT (and other techniques) to solve 2-PC
- [Goldreich-Micali-Wigderson ’87] use OT to solve MPC
  OT is complete for honest-but-curious adversaries.
  Gate-by-gate approach, reasonably efficient for simple circuits.
  For malicious adversaries, needs commitments and ZK proofs.
- [Kilian ’88] OT is complete for malicious adversaries.
  Use OT to do commitments and ZK proofs, general but inefficient.

OT as a building block

Oblivious Transfer is an important building block for Multi-Party Computation (MPC)

- [Yao ’82] use OT (and other techniques) to solve 2-PC
- [Goldreich-Micali-Wigderson ’87] use OT to solve MPC
  OT is complete for honest-but-curious adversaries.
  Gate-by-gate approach, reasonably efficient for simple circuits.
  For malicious adversaries, needs commitments and ZK proofs.
- [Kilian ’88] OT is complete for malicious adversaries.
  Use OT to do commitments and ZK proofs, general but inefficient.
- [Crépeau ’89] defined Committed Oblivious Transfer (explained in the next slide) under the name “verifiable OT.”

OT as a building block

Oblivious Transfer is an important building block for Multi-Party Computation (MPC)

- [Yao ’82] use OT (and other techniques) to solve 2-PC
- [Goldreich-Micali-Wigderson ’87] use OT to solve MPC
  OT is complete for honest-but-curious adversaries.
  Gate-by-gate approach, reasonably efficient for simple circuits.
  For malicious adversaries, needs commitments and ZK proofs.
- [Kilian ’88] OT is complete for malicious adversaries.
  Use OT to do commitments and ZK proofs, general but inefficient.
- [Crépeau ’89] defined Committed Oblivious Transfer (explained in the next slide) under the name “verifiable OT.”
- [Crépeau-Graaf-Tapp ’95] use COT to solve MPC
  more efficient than [Kilian ’88]

Committed Oblivious Transfer

A natural extension to OT

Committed Oblivious Transfer

A natural extension to OT

(plain) OT:

BEFORE: S has input (x_0, x_1), R has input b.
AFTER: S learns nothing, R learns x_b.

Committed Oblivious Transfer

A natural extension to OT

(plain) OT:

BEFORE: S has input (x_0, x_1), R has input b.
AFTER: S learns nothing, R learns x_b.

COT: all the private inputs are committed

BEFORE: S has committed input (x_0, x_1), R has committed input b.
AFTER: S learns nothing, R learns x_b, which is also committed.

Committed Oblivious Transfer

A natural extension to OT

(plain) OT:

BEFORE: S has input (x_0, x_1), R has input b.
AFTER: S learns nothing, R learns x_b.

COT: all the private inputs are committed

BEFORE: S has committed input (x_0, x_1), R has committed input b.
AFTER: S learns nothing, R learns x_b, which is also committed.

Intuition: since COT has commitment "built-in," we don't need to use OT to implement commitments. This makes the construction of MPC protocols more efficient.

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.
- But we still need ZK proofs.

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.
- But we still need ZK proofs.
+ Construction of COT from OT, no hardness assumption needed.

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.
- But we still need ZK proofs.
+ Construction of COT from OT, no hardness assumption needed.
- General construction, inefficient.

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.
- But we still need ZK proofs.
+ Construction of COT from OT, no hardness assumption needed.
- General construction, inefficient.
- Stand-alone security, not concurrently secure, can be malleable, does not compose...

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.
- But we still need ZK proofs.
+ Construction of COT from OT, no hardness assumption needed.
- General construction, inefficient.
- Stand-alone security, not concurrently secure, can be malleable, does not compose...

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.
- But we still need ZK proofs.
+ Construction of COT from OT, no hardness assumption needed.
- General construction, inefficient.
- Stand-alone security, not concurrently secure, can be malleable, does not compose...

[this work] We improve over previous work...

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.
- But we still need ZK proofs.
+ Construction of COT from OT, no hardness assumption needed.
- General construction, inefficient.
- Stand-alone security, not concurrently secure, can be malleable, does not compose...

[this work] We improve over previous work...

+ Add ZK proofs: Extended Committed Oblivious Transfer (ECOT).

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.
- But we still need ZK proofs.
+ Construction of COT from OT, no hardness assumption needed.
- General construction, inefficient.
- Stand-alone security, not concurrently secure, can be malleable, does not compose...

[this work] We improve over previous work...

+ Add ZK proofs: Extended Committed Oblivious Transfer (ECOT).
+ Efficient constructions (based on specific assumptions).

What do we know about COT?

[Crépeau-Graaf-Tapp ’95]

+ More efficient than OT for building MPC protocols.
- But we still need ZK proofs.
+ Construction of COT from OT, no hardness assumption needed.
- General construction, inefficient.
- Stand-alone security, not concurrently secure, can be malleable, does not compose...

[this work] We improve over previous work...

+ Add ZK proofs: Extended Committed Oblivious Transfer (ECOT).
+ Efficient constructions (based on specific assumptions).
+ Universally composable (concurrently secure, non-malleable...)

Summary of our results

Summary of our results

- The Extended Committed Oblivious Transfer (ECOT) functionality.
  OT + commitment + ZK proofs

Summary of our results

- The Extended Committed Oblivious Transfer (ECOT) functionality.
  OT + commitment + ZK proofs
- An efficient protocol realizing ECOT.
  based on specific number theoretical assumptions.
  constant rounds, constant number of pub-key operations .
  universally composable (in the CRS model)

Summary of our results

- The Extended Committed Oblivious Transfer (ECOT) functionality.
  OT + commitment + ZK proofs
- An efficient protocol realizing ECOT.
  based on specific number theoretical assumptions.
  constant rounds, constant number of pub-key operations .
  universally composable (in the CRS model)
- Efficient construction of 2-PC/MPC protocols using ECOT.
  Universally composable (concurrently secure, non-malleable...)
  More efficient than [Canetti-Lindell-Ostrovsky-Sahai ’02]
  Weaker set-up condition than [Damgård-Nielsen ’03]

The ECOT functionality

Assume n parties, $P_1, P_2, ..., P_n$.

Commit: Any party P_i can commit to a bit b.

The ECOT functionality

Assume n parties, $P_1, P_2, ..., P_n$.

Commit: Any party P_i can commit to a bit b.

Prove: Any party can prove a relation $R(b_0, b_1, b_2)$ over the bits $\{b_0, b_1, b_2\}$ it committed.
(simple relations only!)

The ECOT functionality

Assume n parties, $P_1, P_2, ..., P_n$.

Commit: Any party P_i can commit to a bit b.

Prove: Any party can prove a relation $R(b_0, b_1, b_2)$ over the bits $\{b_0, b_1, b_2\}$ it committed.
(simple relations only!)

Transfer: If P_i has committed to bits $\{b_0, b_1\}$, and P_j has committed to bit t, then P_j receives bit b_t, which is also committed by P_j.

The ECOT functionality

Assume n parties, $P_1, P_2, ..., P_n$.

Commit: Any party P_i can commit to a bit b.

Prove: Any party can prove a relation $R(b_0, b_1, b_2)$ over the bits $\{b_0, b_1, b_2\}$ it committed.
(simple relations only!)

Transfer: If P_i has committed to bits $\{b_0, b_1\}$, and P_j has committed to bit t, then P_j receives bit b_t, which is also committed by P_j.

Open: Any party P_i can open any bit it has committed.

The ECOT functionality

Assume n parties, $P_1, P_2, ..., P_n$.

Commit: Any party P_i can commit to a bit b.

Prove: Any party can prove a relation $R(b_0, b_1, b_2)$ over the bits $\{b_0, b_1, b_2\}$ it committed.
(simple relations only!)

Transfer: If P_i has committed to bits $\{b_0, b_1\}$, and P_j has committed to bit t, then P_j receives bit b_t, which is also committed by P_j.

Open: Any party P_i can open any bit it has committed.

Commit + Open = commitment

Commit + Transfer + Open = COT

Commit + Prove + Transfer + Open = ECOT

The UCECOT protocol

We first overview the ingredients used.

The UCECOT protocol

We first overview the ingredients used.

- Concurrent Oblivious Transfer [Garay-MacKenzie ’00]
  Based on [Bellare-Micali ’89]
  (more details in the next slide)

The UCECOT protocol

We first overview the ingredients used.

- Concurrent Oblivious Transfer [Garay-MacKenzie '00]
  Based on [Bellare-Micali '89]
  (more details in the next slide)
- Pedersen commitment [Pedersen '91]
  CRS is $g, h \in \mathbb{Z}_p^*$ with unknown $\log_g h$;
  commitment to b is $g^r h^b$ with random r.
  perfect hiding, computationally binding (assuming DLP is hard).
  Used for the commit phase.

The UCECOT protocol

We first overview the ingredients used.

- Concurrent Oblivious Transfer [Garay-MacKenzie '00]
  Based on [Bellare-Micali '89]
  (more details in the next slide)
- Pedersen commitment [Pedersen '91]
  CRS is $g, h \in \mathbb{Z}_p^*$ with unknown $\log_g h$;
  commitment to b is $g^r h^b$ with random r.
  perfect hiding, computationally binding (assuming DLP is hard).
  Used for the commit phase.
- Efficient UCZK protocols [Garay-MacKenzie-Yang '03]
  Construction of efficient UCZK protocols from Ω-protocols.
  Used for the prove phase.
  "simple relations" = "efficient protocols"

The Oblivious Transfer: basics

- Begin:
  P_i has committed $B_0 = g^{r_0} h^{b_0}$, $B_1 = g^{r_1} h^{b_1}$;
  P_j has committed $B_t = g^{r_t} h^t$.

The Oblivious Transfer: basics

- Begin:
  P_i has committed $B_0 = g^{r_0} h^{b_0}$, $B_1 = g^{r_1} h^{b_1}$;
  P_j has committed $B_t = g^{r_t} h^t$.
- Transfer:
  P_i picks random a_0, a_1 and sets

The Oblivious Transfer: basics

- Begin:
  P_i has committed $B_0 = g^{r_0} h^{b_0}$, $B_1 = g^{r_1} h^{b_1}$;
  P_j has committed $B_t = g^{r_t} h^t$.
- Transfer:
  P_i picks random a_0, a_1 and sets
  - $E_0 \leftarrow (g^{a_0}, B_t^{a_0} h^{b_0})$:
    an ElGamal encryption of h^{b_0} using pub-key (g, B_t).

The Oblivious Transfer: basics

- Begin:
  P_i has committed $B_0 = g^{r_0} h^{b_0}$, $B_1 = g^{r_1} h^{b_1}$;
  P_j has committed $B_t = g^{r_t} h^t$.
- Transfer:
  P_i picks random a_0, a_1 and sets
  - $E_0 \leftarrow (g^{a_0}, B_t^{a_0} h^{b_0})$:
    an ElGamal encryption of h^{b_0} using pub-key (g, B_t).
  - $E_1 \leftarrow (g^{a_1}, (B_t/h)^{a_1} h^{b_1})$:
    an ElGamal encryption of h^{b_1} using pub-key $(g, B_t/h)$.

The Oblivious Transfer: basics

- Begin:
  P_i has committed $B_0 = g^{r_0} h^{b_0}$, $B_1 = g^{r_1} h^{b_1}$;
  P_j has committed $B_t = g^{r_t} h^t$.
- Transfer:
  P_i picks random a_0, a_1 and sets
  - $E_0 \leftarrow (g^{a_0}, B_t^{a_0} h^{b_0})$:
    an ElGamal encryption of h^{b_0} using pub-key (g, B_t).
  - $E_1 \leftarrow (g^{a_1}, (B_t/h)^{a_1} h^{b_1})$:
    an ElGamal encryption of h^{b_1} using pub-key $(g, B_t/h)$.
  - P_j knows either $\log_g(B_t)$ ($t = 0$) or $\log_g(B_t/h)$ ($t = 1$),
    but not both.

The Oblivious Transfer: basics

- Begin:
  P_i has committed $B_0 = g^{r_0} h^{b_0}$, $B_1 = g^{r_1} h^{b_1}$;
  P_j has committed $B_t = g^{r_t} h^t$.
- Transfer:
  P_i picks random a_0, a_1 and sets
  - $E_0 \leftarrow (g^{a_0}, B_t^{a_0} h^{b_0})$:
    an ElGamal encryption of h^{b_0} using pub-key (g, B_t).
  - $E_1 \leftarrow (g^{a_1}, (B_t/h)^{a_1} h^{b_1})$:
    an ElGamal encryption of h^{b_1} using pub-key $(g, B_t/h)$.
  - P_j knows either $\log_g(B_t)$ ($t = 0$) or $\log_g(B_t/h)$ ($t = 1$),
    but not both.
  - So P_j can decrypt E_t, and E_{1-t} appears random (assuming DDH, needs proof).

The Oblivious Transfer: basics

- **Begin:**
  P_i has committed $B_0 = g^{r_0} h^{b_0}$, $B_1 = g^{r_1} h^{b_1}$;
  P_j has committed $B_t = g^{r_t} h^t$.
- **Transfer:**
  P_i picks random a_0, a_1 and sets
  - $E_0 \leftarrow (g^{a_0}, B_t^{a_0} h^{b_0})$:
    an ElGamal encryption of h^{b_0} using pub-key (g, B_t).
  - $E_1 \leftarrow (g^{a_1}, (B_t/h)^{a_1} h^{b_1})$:
    an ElGamal encryption of h^{b_1} using pub-key $(g, B_t/h)$.
  - P_j knows either $\log_g(B_t)$ ($t = 0$) or $\log_g(B_t/h)$ ($t = 1$),
    but not both.
  - So P_j can decrypt E_t, and E_{1-t} appears random (assuming DDH, needs proof).
- **End:**
  P_j recovers b_t and commits it using Pedersen.

The Oblivious Transfer: bells and whistles

- The protocol in the previous slide is secure for honest-but-curious adversaries.

The Oblivious Transfer: bells and whistles

- The protocol in the previous slide is secure for honest-but-curious adversaries.
- We add ZK proofs and proof-of-knowledges to make it secure for malicious adversaries.

The Oblivious Transfer: bells and whistles

- The protocol in the previous slide is secure for honest-but-curious adversaries.
- We add ZK proofs and proof-of-knowledges to make it secure for malicious adversaries.
- Using of UCZK proofs makes the protocol universally composable.

The Oblivious Transfer: bells and whistles

- The protocol in the previous slide is secure for honest-but-curious adversaries.
- We add ZK proofs and proof-of-knowledges to make it secure for malicious adversaries.
- Using of UCZK proofs makes the protocol universally composable.
- DL-related proofs$\Rightarrow$ efficient ZK protocols $\Rightarrow$ efficient UCECOT.

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson ’87]:

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson ’87]:
  - Encode the function as a boolean circuit of AND and XOR.

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson ’87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson '87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.
  - Per-gate evaluation: (shared) inputs $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson '87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.
  - Per-gate evaluation: (shared) inputs $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$
    - XOR: no interaction needed.
      $(x \oplus y) = (x_0 \oplus y_0) \oplus (x_1 \oplus y_1)$

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson '87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.
  - Per-gate evaluation: (shared) inputs $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$
    - XOR: no interaction needed.
      $(x \oplus y) = (x_0 \oplus y_0) \oplus (x_1 \oplus y_1)$
    - AND: needs a $\binom{4}{1}$-OT (can be build from $\binom{2}{1}$-OT).
      P_0 prepares random z_0 and...

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson '87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.
  - Per-gate evaluation: (shared) inputs $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$
    - XOR: no interaction needed.
      $(x \oplus y) = (x_0 \oplus y_0) \oplus (x_1 \oplus y_1)$
    - AND: needs a $\binom{4}{1}$-OT (can be build from $\binom{2}{1}$-OT).
      P_0 prepares random z_0 and...
      $b_{00} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 0))$

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson '87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.
  - Per-gate evaluation: (shared) inputs $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$
    - XOR: no interaction needed.
      $(x \oplus y) = (x_0 \oplus y_0) \oplus (x_1 \oplus y_1)$
    - AND: needs a $\binom{4}{1}$-OT (can be build from $\binom{2}{1}$-OT).
      P_0 prepares random z_0 and...
      $$b_{00} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 0))$$
      $$b_{01} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 1))$$

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson '87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.
  - Per-gate evaluation: (shared) inputs $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$
    - XOR: no interaction needed.
      $(x \oplus y) = (x_0 \oplus y_0) \oplus (x_1 \oplus y_1)$
    - AND: needs a $\binom{4}{1}$-OT (can be build from $\binom{2}{1}$-OT).
      P_0 prepares random z_0 and...
      $$b_{00} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 0))$$
      $$b_{01} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 1))$$
      $$b_{10} = z_0 \oplus ((x_0 \oplus 1) \wedge (y_0 \oplus 0))$$

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson '87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.
  - Per-gate evaluation: (shared) inputs $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$
    - XOR: no interaction needed.
      $(x \oplus y) = (x_0 \oplus y_0) \oplus (x_1 \oplus y_1)$
    - AND: needs a $\binom{4}{1}$-OT (can be build from $\binom{2}{1}$-OT).
      P_0 prepares random z_0 and...

$$b_{00} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 0))$$
$$b_{01} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 1))$$
$$b_{10} = z_0 \oplus ((x_0 \oplus 1) \wedge (y_0 \oplus 0))$$
$$b_{11} = z_0 \oplus ((x_0 \oplus 1) \wedge (y_0 \oplus 1))$$

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson '87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.
  - Per-gate evaluation: (shared) inputs $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$
    - XOR: no interaction needed.
      $(x \oplus y) = (x_0 \oplus y_0) \oplus (x_1 \oplus y_1)$
    - AND: needs a $\binom{4}{1}$-OT (can be build from $\binom{2}{1}$-OT).
      P_0 prepares random z_0 and...
      $$b_{00} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 0))$$
      $$b_{01} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 1))$$
      $$b_{10} = z_0 \oplus ((x_0 \oplus 1) \wedge (y_0 \oplus 0))$$
      $$b_{11} = z_0 \oplus ((x_0 \oplus 1) \wedge (y_0 \oplus 1))$$
      P_1 picks $b_{x_1 y_1}$

Constructing 2-PC/MPC protocols using ECOT: protocol

We only consider 2-PC in this talk (MPC similar)...

- Same approach as [Goldreich-Micali-Wigderson '87]:
  - Encode the function as a boolean circuit of AND and XOR.
  - Each bit on the wire is shared using XOR.
  - Per-gate evaluation: (shared) inputs $x = x_0 \oplus x_1$, $y = y_0 \oplus y_1$
    - XOR: no interaction needed.
      $(x \oplus y) = (x_0 \oplus y_0) \oplus (x_1 \oplus y_1)$
    - AND: needs a $\binom{4}{1}$-OT (can be build from $\binom{2}{1}$-OT).
      P_0 prepares random z_0 and...
      $$b_{00} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 0))$$
      $$b_{01} = z_0 \oplus ((x_0 \oplus 0) \wedge (y_0 \oplus 1))$$
      $$b_{10} = z_0 \oplus ((x_0 \oplus 1) \wedge (y_0 \oplus 0))$$
      $$b_{11} = z_0 \oplus ((x_0 \oplus 1) \wedge (y_0 \oplus 1))$$
      P_1 picks $b_{x_1 y_1}$
- ECOT has commitments and proofs built-in. So directly secure for malicious adversaries ([GMW] needs a compiler).

Constructing 2-PC/MPC protocols using ECOT: efficiency

- Our construction is very efficient (constant round and constant number of pub-key operations per gate).

Constructing 2-PC/MPC protocols using ECOT: efficiency

- Our construction is very efficient (constant round and constant number of pub-key operations per gate).
- This is more efficient than [Canetti-Lindell-Ostrovsky-Sahai ’02].

Constructing 2-PC/MPC protocols using ECOT: efficiency

- Our construction is very efficient (constant round and constant number of pub-key operations per gate).
- This is more efficient than [Canetti-Lindell-Ostrovsky-Sahai '02].
  - [CLOS] uses the [GMW] approach: start with a protocol secure against honest-but-curious adversaries (using OT) and then use a compiler to add commitments and ZK proofs.

Constructing 2-PC/MPC protocols using ECOT: efficiency

- Our construction is very efficient (constant round and constant number of pub-key operations per gate).
- This is more efficient than [Canetti-Lindell-Ostrovsky-Sahai '02].
  - [CLOS] uses the [GMW] approach: start with a protocol secure against honest-but-curious adversaries (using OT) and then use a compiler to add commitments and ZK proofs.
  - [CLOS] uses UC-commitment and UCZK, so that resultant protocol is universally composable.

Constructing 2-PC/MPC protocols using ECOT: efficiency

- Our construction is very efficient (constant round and constant number of pub-key operations per gate).
- This is more efficient than [Canetti-Lindell-Ostrovsky-Sahai '02].
  - [CLOS] uses the [GMW] approach: start with a protocol secure against honest-but-curious adversaries (using OT) and then use a compiler to add commitments and ZK proofs.
  - [CLOS] uses UC-commitment and UCZK, so that resultant protocol is universally composable.
  - The ZK proofs are about executions of Turing machines — unlikely to be efficient

Constructing 2-PC/MPC protocols using ECOT: efficiency

- Our construction is very efficient (constant round and constant number of pub-key operations per gate).
- This is more efficient than [Canetti-Lindell-Ostrovsky-Sahai '02].
  - [CLOS] uses the [GMW] approach: start with a protocol secure against honest-but-curious adversaries (using OT) and then use a compiler to add commitments and ZK proofs.
  - [CLOS] uses UC-commitment and UCZK, so that resultant protocol is universally composable.
  - The ZK proofs are about executions of Turing machines — unlikely to be efficient
  - We build the commitment and ZK directly into OT: simple relations = efficient ZK proofs.

Constructing 2-PC/MPC protocols using ECOT: efficiency

- Our construction is very efficient (constant round and constant number of pub-key operations per gate).
- This is more efficient than [Canetti-Lindell-Ostrovsky-Sahai '02].
  - [CLOS] uses the [GMW] approach: start with a protocol secure against honest-but-curious adversaries (using OT) and then use a compiler to add commitments and ZK proofs.
  - [CLOS] uses UC-commitment and UCZK, so that resultant protocol is universally composable.
  - The ZK proofs are about executions of Turing machines — unlikely to be efficient
  - We build the commitment and ZK directly into OT: simple relations = efficient ZK proofs.
- [Damgård-Nielsen '03]: efficient universally composable MPC protocols using homomorphic encryptions in the PKI model (sightly stronger) for adaptive corruption w/o erasing (ours assumes erasing).

Summary

- ECOT = OT + commitment + ZK proofs

Summary

- ECOT = OT + commitment + ZK proofs
- Efficient and universally composable protocols

Summary

- ECOT = OT + commitment + ZK proofs
- Efficient and universally composable protocols
- Efficient construction of universally composable 2-PC/MPC protocols

Summary

- ECOT = OT + commitment + ZK proofs
- Efficient and universally composable protocols
- Efficient construction of universally composable 2-PC/MPC protocols

- Full paper in IACR eprint