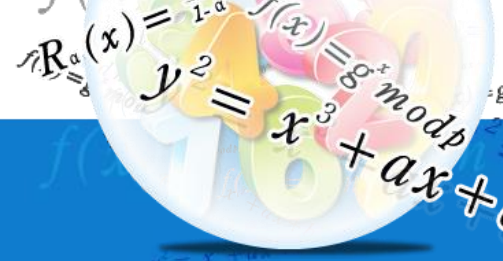


A Compiler of Two-Party Protocols for Composable and Game-Theoretic Security, and Its Application to Oblivious Transfer

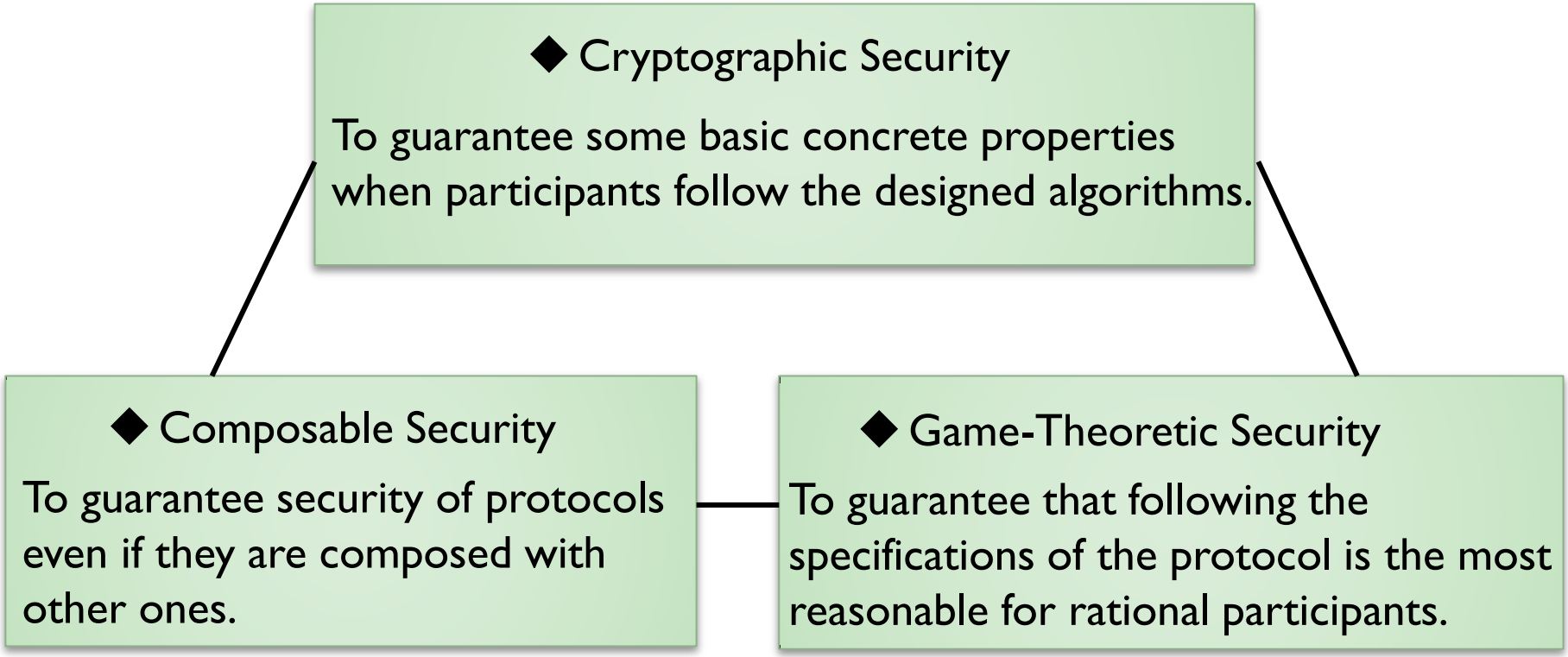
Shota Goto and Junji Shikata

Yokohama National University, Japan

IMACC 2015

Introduction

Security notions of cryptographic protocols

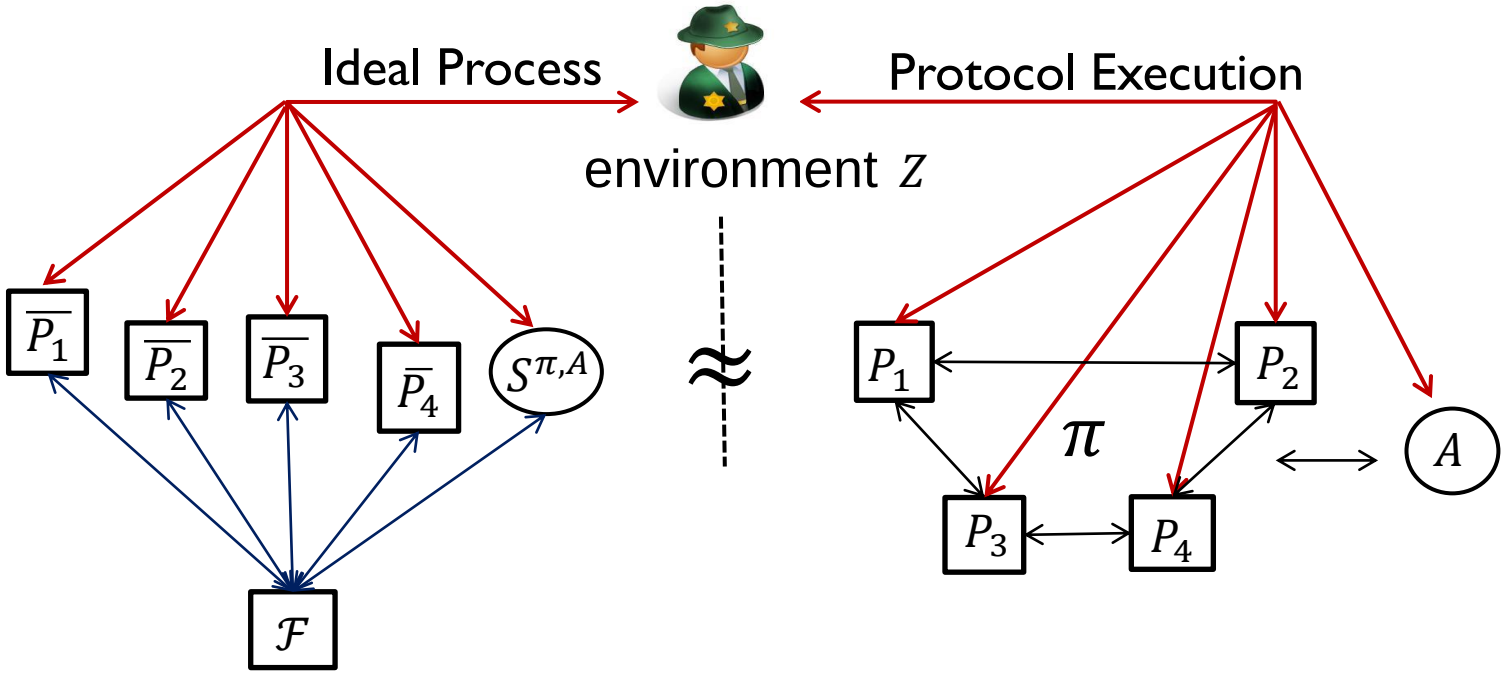


These concepts capture situations from different perspectives.

Universal Composability [C01]

Protocols remain secure even if arbitrarily composed with other ones.

A protocol π UC-realizes an ideal functionality $\mathcal{F}$, if for any adversary A there exists a simulator S such that for any environment Z, Ideal Process ≈ Protocol Execution.



[C01] R. Canetti, “Universally composable Security: A new paradigm for cryptographic protocols,” FOCS 2001.

Game-Theoretic Security

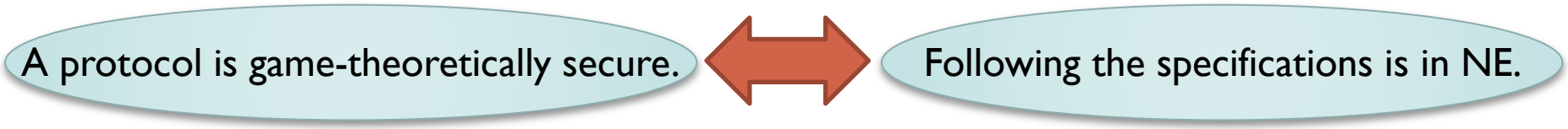
Utility Function :

- It mathematically represents the preferences of each participant.
- Rational participants select the strategies with which they can get the highest value of utility.

Nash Equilibrium (NE) :

- One of the most commonly used solution concept.
- When all participants choose the NE strategies, no party can gain his utility by changing his strategy unilaterally.

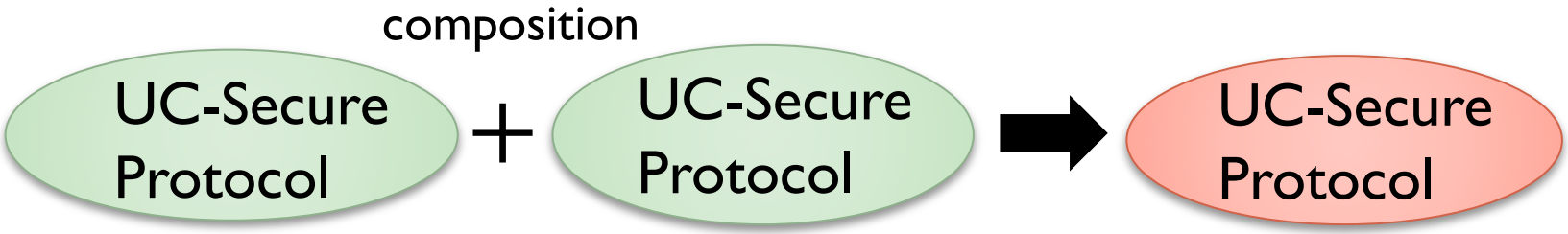
[ACH11]



[ACH11] G. Asharov, R. Canatti, C. Hazay, "Towards a game theoretic view of secure computation," EUROCRYPT 2011.

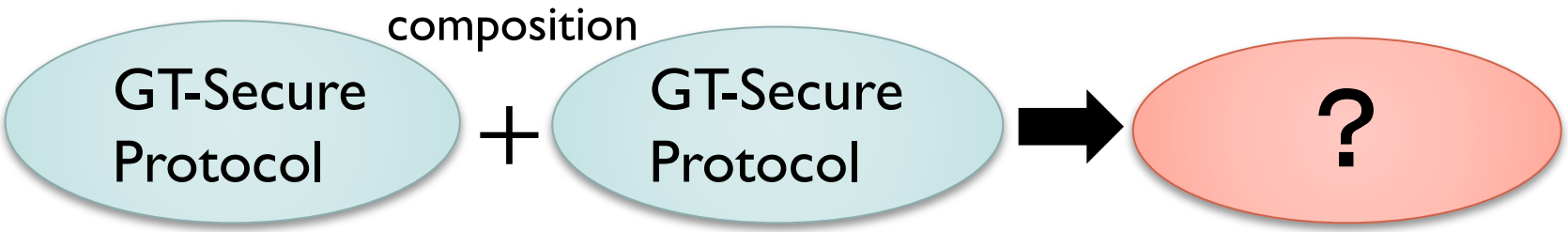
Our Study

Generally…



We consider the following question :

Does composing protocols having GT-security result in a secure protocol in the sense of GT-security?



Our Study

We address how protocols with security in game-theoretic model can be composed to obtain an overall game-theoretically secure protocol.

- Our Primary Goal

  To achieve a primitive with both UC and GT security.

- Our Approach

  1. We try to adapt the UC model compiler of [CLOS02] to the Local UC framework [CV12].
  2. We consider the application of our compiler to oblivious transfer (OT) protocols. In particular, we consider the construction of [GMW87, G04, CLOS02].

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, “Universally composable two-party and multi-party secure computation,” STOC 2002.

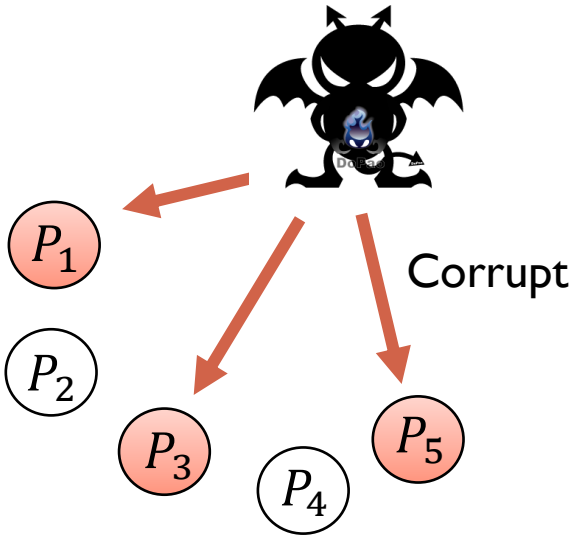
[CV12] R. Canetti, M. Vald, ”Universally Composable Security With Local Adversaries,” SCN 2012.

[GMW87] O. Goldreich, S. Micali, A. Wigderson, “How to play any mental game or a completeness theorem for protocols with honest majority,” STOC 1987.

[G04] O. Goldreich, “The Foundations of Cryptography,” Basic Applications, vol. 2. Cambridge University Press 2004.

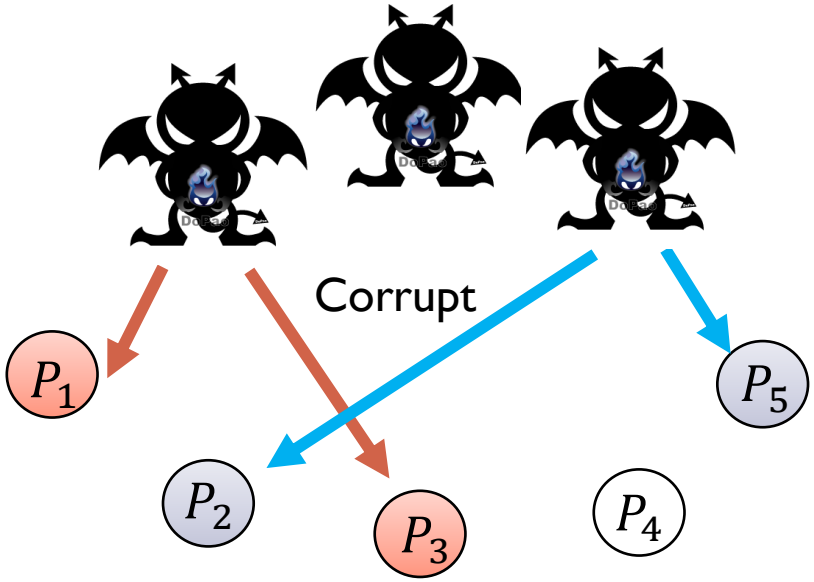
Uncapturable Situations in the UC Framework

capturable



A protocol is attacked by a centralized adversary.

uncapturable

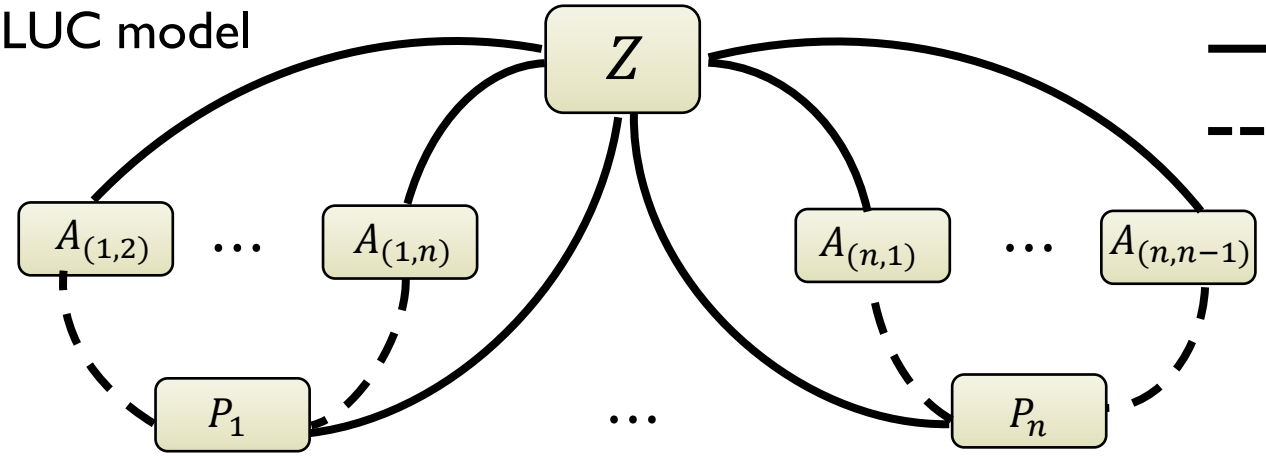


A protocol is attacked by adversaries who are not cooperative.

Another model which is suitable for game-theoretic settings is needed.

Local UC [CV12]

LUC model



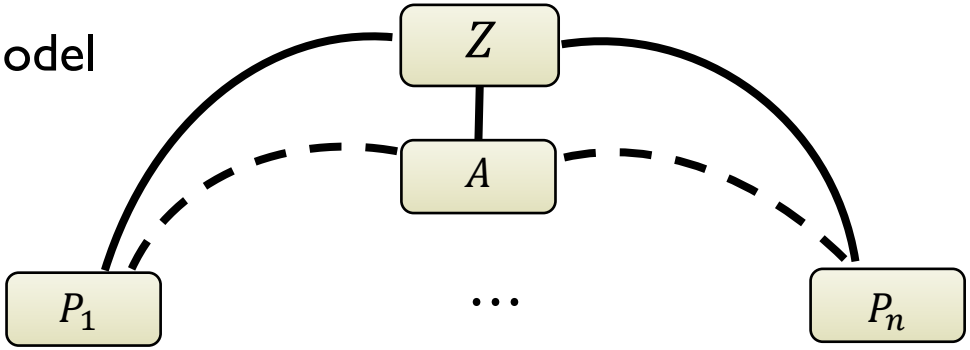
: input / output

: communication

$\boldsymbol{P}$: set of party IDs

$A_{(i,j)}$: adversary for ordered pairs $(i,j) \in \boldsymbol{P}^2$ $(i \neq j)$

UC model



[CV12] R. Canetti, M. Vald, "Universally Composable Security With Local Adversaries," SCN 2012.

Connection between the LUC Framework and GT-security

At first sight, one may think these two notions are not well connected.

- GT-security requires that all participants can get the highest utility when each of them acts honestly.
- LUC-security requires the indistinguishability between the real-world and the ideal-world.

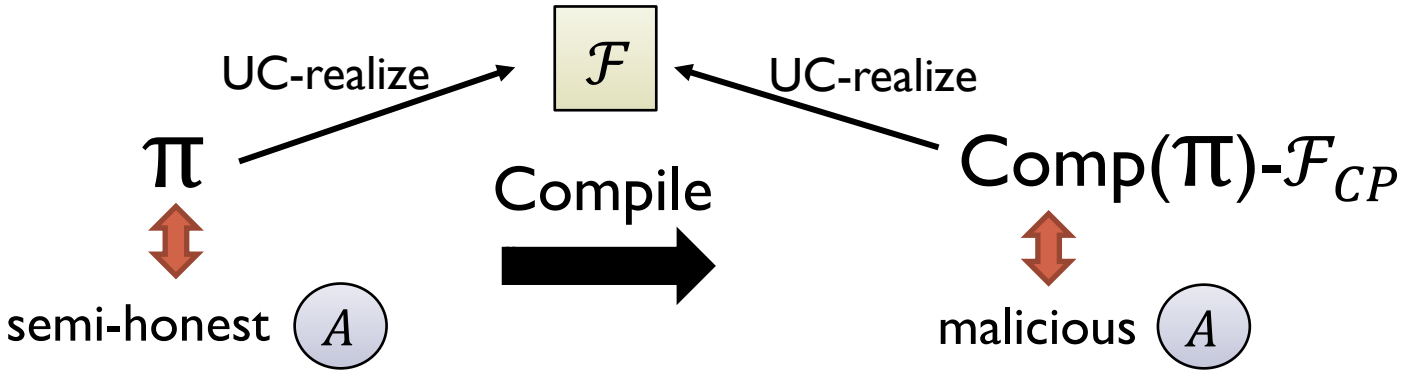
However, there is an important point common to these two notions, namely, all participants are allowed to behave in a malicious (or rational) way.

If we define an ideal functionality in the LUC framework accurately so that it captures correct actions of participants and matches utility functions, we can say LUC-security implies GT-security.

Defining an ideal functionality of complicated protocol may be a hard work, therefore it cannot be unconditionally said so.

Compiler in the UC Framework [CLOS02]

- A universally composable compiler based on the work of [GMW87].
- It transforms any protocol that is designed for the semi-honest adversarial model into one that is secure against malicious adversaries.



How to obtain security :

Commit-and-prove functionality [CLOS02]

- Force the adversary to use a fixed input.
- Force the adversary to use a uniform random tape.
- Force the adversary to follow the protocol exactly.

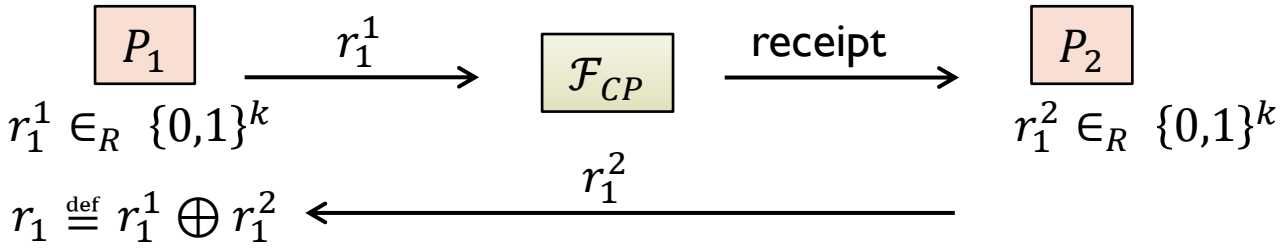
[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, "Universally composable two-party and multi-party secure computation," STOC 2002.

[GMW87] O. Goldreich, S. Micali, A. Wigderson, "How to play any mental game or a completeness theorem for protocols with honest majority," STOC 1987.

Compiler in the UC Framework [CLOS02]

A party P_1 proceeds as follows (the code for a party P_2 is analogous).

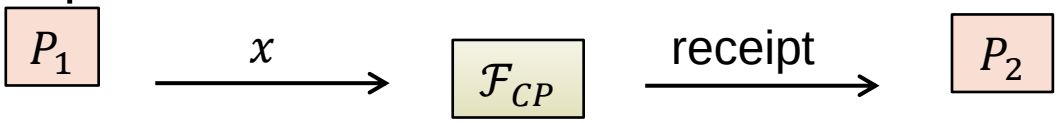
1. Random tape generation



r_1 serves as P_1's random tape for execution of π.

2. Activation due to a new input

(a) Input commitment



(b) Protocol computation

P_1 runs the code of π using $\overline{x}$, $\overline{m_1}$, r_1.

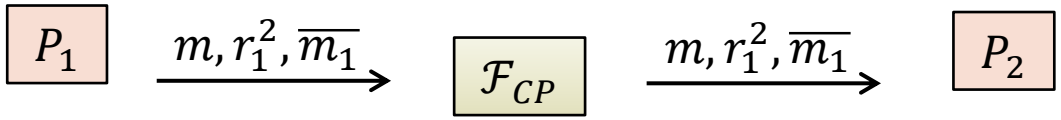
$\overline{x}$: The list of inputs.

$\overline{m_1}$: The series of messages that P_1 received until now.

Compiler in the UC Framework [CLOS02]

(c) Outgoing message transmission

P_1 proves that m is truly the correct message.



The relation R_π for $\mathcal{F}_{CP}$ is defined as follows.

$$R_\pi = \{((m, r_1^2, \overline{m_1}), (\bar{x}, r_1^1)) \mid m = \pi(\bar{x}, r_1^1 \oplus r_1^2, \overline{m_1})\}$$

3. Activation due to incoming message

P_1 verifies that the following conditions hold:

(a)r_2^1 is the string that P_1 sent to P_2 in the step 1.

(b)$\overline{m_2}$ equals the series of messages received by P_2 from P_1 until now.

If the conditions hold, then P_1 appends m to its list $\overline{m_1}$ and proceeds as in the steps 2-(b) and 2-(c).

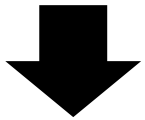
4. Output

Whenever π generates an output, Comp(π) generates the same output.

Compiler in the LUC Framework

In the UC framework, a semi-honest adversary A can internally simulate the behavior of a malicious adversary A' by delivering a message only when A' sends a correct message.

To utilize the compiler in the LUC framework, we need to similarly complete the simulation.



However, we cannot do that without any modifications on the existing process.

Compiler in the LUC Framework

The reason of this impossibility lies in the difference of communication models.

UC

π $\quad P_1 \rightleftarrows P_2$

$\mathrm{Comp}(\pi)$-$\mathcal{F}_{CP}$ $\quad P_1 \rightleftarrows \mathcal{F}_{CP} \rightleftarrows P_2$

LUC

π $\quad P_1 \rightleftarrows Z \rightleftarrows P_2$

$\mathrm{Comp}(\pi)$-$\mathcal{F}_{CP}$ $\quad P_1 \rightleftarrows \mathcal{F}_{CP} \rightleftarrows P_2$

The environment Z can distinguish these two situations.

π-$\hat{\mathcal{F}}_{mt}$

$P_1 \rightleftarrows \hat{\mathcal{F}}_{mt} \rightleftarrows P_2$

We consider switching the interacting process of an original protocol π to the one which uses a subroutine.

Ideal functionality $\hat{\mathcal{F}}_{mt}$(message transmission)

1. Upon receiving $(\mathbf{Send}, sid, m, P_j)$ from P_i, send a public delayed output $(\mathbf{Send}, sid, m, P_i)$ to the party P_j.

2. Upon receiving $(\mathbf{Deliver}, S_{(j,i)}, m)$ from $S_{(i,j)}$, send $(\mathbf{Delivered}, S_{(i,j)}, m)$ to $S_{(j,i)}$.

Compiler in the LUC Framework

For using the commit-and-prove functionality in the LUC model, we adopt the notion of the merger functionality in [CV12].

↳ We add the message delivery algorithm for simulators.

Ideal Functionality $\hat{\mathcal{F}}_{CP}$(commit-and-prove)

committer : C receiver : V adversary : S
security parameter : k relation : R

- Upon receiving a message (**Commit**, $sid, w \in \{0,1\}^k$) from C, append the value w to the list $\overline{w}$, and send a public delayed output (**receipt**, sid) to V.

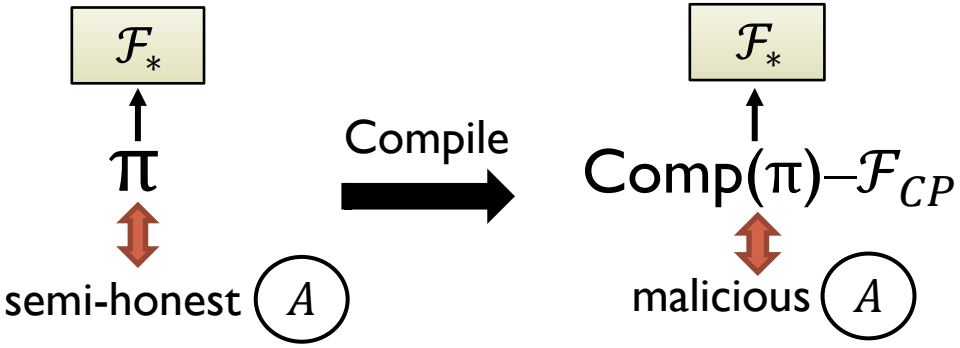
- Upon receiving a message (**CP–prover**, $sid, x \in \{0,1\}^{poly(k)}$) from C, compute $R(x, \overline{w})$; If $R(x, \overline{w}) = 1$, then send a public delayed output (**CP–proof**, sid, x) to V.

- Upon receiving a message (**Deliver**, $S_{(j,i)}, m$) from $S_{(i,j)}$, send the message (**Delivered**, $S_{(i,j)}, m$) to $S_{(j,i)}$.

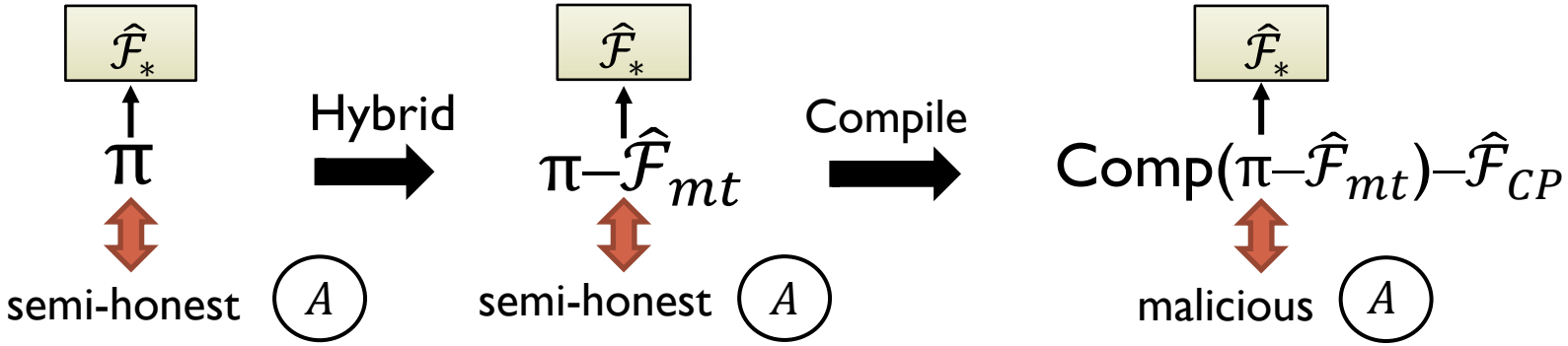
[CV12] R. Canetti, M. Vald, "Universally Composable Security With Local Adversaries," SCN 2012.

Compiler in the LUC Framework

◆ UC model



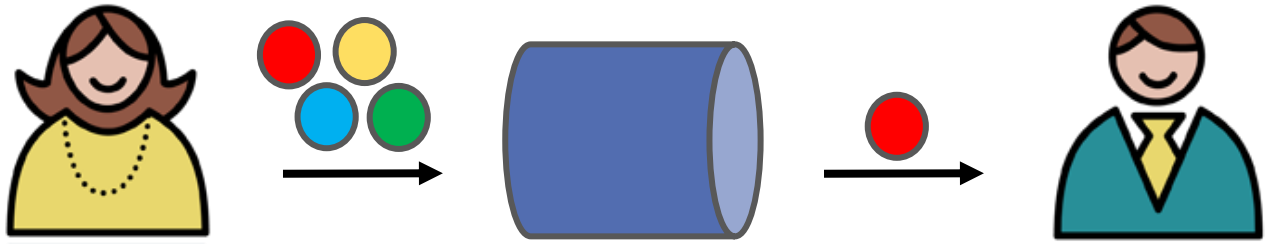
◆ LUC model



Theorem 1 :
Let $\mathcal{F}$ be a two-party functionality and let π be a protocol that LUC-realizes $\mathcal{F}$ against semi-honest adversaries. Then Comp(π–$\hat{\mathcal{F}}_{mt}$)–$\hat{\mathcal{F}}_{CP}$ LUC-realizes $\mathcal{F}$ against malicious adversaries.

Application to Oblivious Transfer

Oblivious Transfer [R81] :
A sender transfers one of potentially many pieces of information to a receiver, but remains oblivious as to what piece has been transferred.

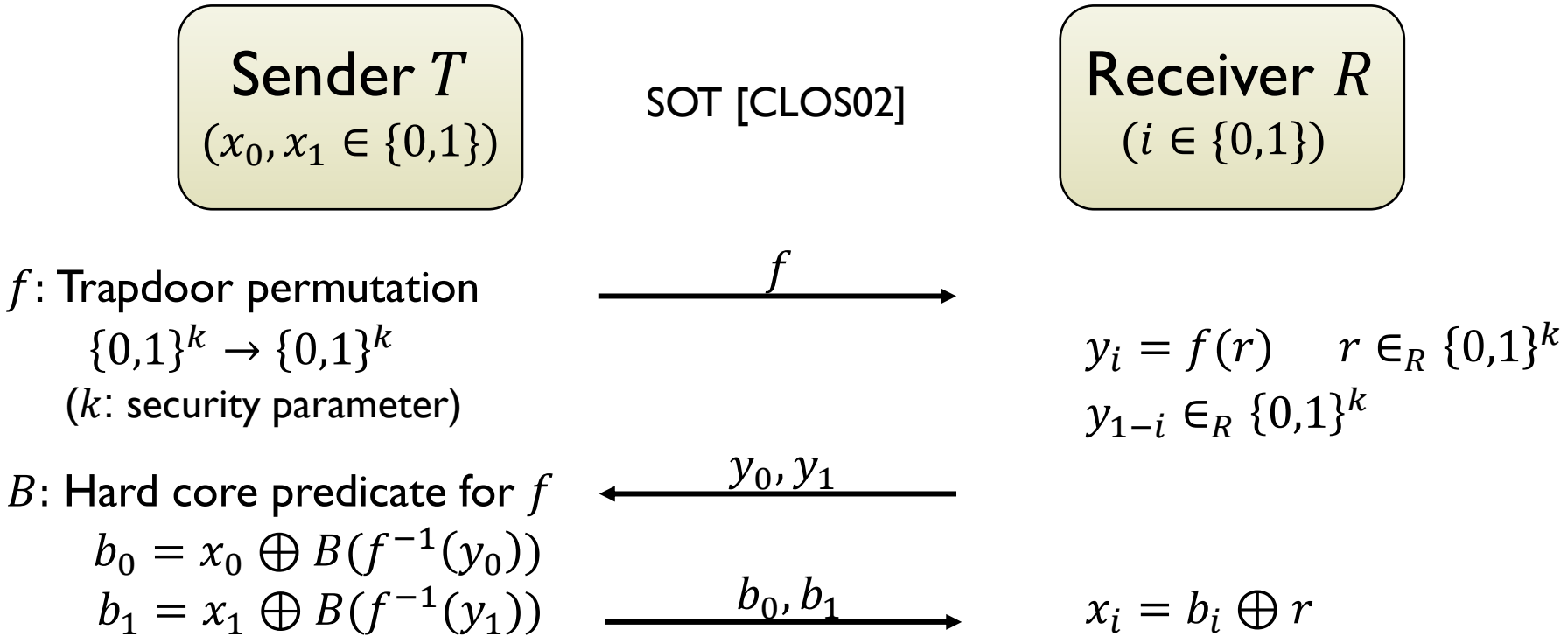


- We consider the construction of 1-out-of-2 OT protocol, denoted by SOT, in [CLOS02] which UC-realizes the OT functionality in static and semi-honest adversarial model.
- We investigate whether SOT is GT-secure, before and after being compiled.

$$\text{SOT} \xrightarrow{\text{Compile}} \text{Comp}(\text{SOT}\text{–}\hat{\mathcal{F}}_{mt})\text{–}\hat{\mathcal{F}}_{CP}$$

[R81] M.O. Rabin, “How to exchange secrets with oblivious transfer,” Tech. Rep. TR-81, Aiken Computation Lab, Harvard University, 1981.
[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, “Universally composable two-party and multi-party secure computation,” STOC 2002.

OT Protocol in the UC Framework



R receives x_i such that R cannot obtain any more information, while T obtains no information about the selection of R.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, A. Sahai, "Universally composable two-party and multi-party secure computation," STOC 2002.

Analysis of GT-Security

Definition : Utility functions [HTYY12]
Let π be an OT protocol having a sender T with inputs $x_0, x_1 \in \{0,1\}$ and a receiver R with an input $i \in \{0,1\}$. Let $\alpha_T, \beta_T, \gamma_T, \alpha_R, \beta_R, \gamma_R$ be positive constants.
The utility functions U_T for T and U_R for R are defined as follows.

$$U_T = -\alpha_T \cdot \left(\Pr[x' = x_{1-i} \mid guess_R(T(x_0, x_1), R(i, x_i)) = x'] - \frac{1}{2}\right)$$
$$+\beta_T \cdot (\Pr[fin(T(x_0, x_1), R(i)) = 1] - 1)$$
$$+\gamma_T \cdot \left(\Pr[i' = i \mid guess_T(T(x_0, x_1), R(i)) = i'] - \frac{1}{2}\right)$$

$$U_R = -\alpha_R \cdot \left(\Pr[i' = i \mid guess_T(T(x_0, x_1), R(i)) = i'] - \frac{1}{2}\right)$$
$$+\beta_R \cdot (\Pr[fin(T(x_0, x_1), R(i)) = 1] - 1)$$
$$+\gamma_R \cdot \left(\Pr[x' = x_{1-i} \mid guess_R(T(x_0, x_1), R(i, x_i)) = x'] - \frac{1}{2}\right)$$

where $guess_T(\cdot)$ and $guess_R(\cdot)$ mean guessing by T and R for the opponent's private value, and $fin(\cdot)$ represents the completion of the protocol execution.

[HTYY12] H. Higo, K. Tanaka, A. Yamada, K. Yasunaga, "A game-theoretic perspective on oblivious transfer," ACISP 2012.

Analysis of GT-Security

Definition : Nash equilibrium

For a pair of utility functions (U_T, U_R), we say that a pair of strategies (σ_T, σ_R) is in Nash equilibrium, if for every pair of strategies (σ_T^*, σ_R^*), it holds:

$$U_T(\sigma_T, \sigma_R) \geq U_T(\sigma_T^*, \sigma_R) - negl(n)$$
$$U_R(\sigma_T, \sigma_R) \geq U_R(\sigma_T, \sigma_R^*) - negl(n)$$

Definition : Game-theoretic security for OT

Let π be an OT protocol having a sender T and a receiver R.
Let σ_T and σ_R be strategies planned to follow all the specifications of π.
We say that π is game-theoretically secure, if the pair of strategies (σ_T, σ_R) is in Nash equilibrium with respect to the pair of utility functions (U_T, U_R).

Analysis of GT-Security

Theorem 2 :
The protocol SOT is not game-theoretically secure in the presence of rational parties, however, the compiled protocol Comp(SOT-$\hat{\mathcal{F}}_{MT}$)-$\hat{\mathcal{F}}_{CP}$ is game-theoretically secure in the presence of rational parties.

- SOT

Receiver R :

- If R applies f for generating y_{1-i}, R can obviously obtain T's private value x_{1-i} in addition to x_i.
- y_{1-i} and r are randomly chosen, so R's dishonest behavior is not detectable.
- This results in increasing the value

$$+\gamma_R \cdot \left(\Pr[x' = x_{1-i} \mid guess_R(T(x_0, x_1), R(i, x_i)) = x'] - \frac{1}{2}\right)$$

Sender T :

i. $-\alpha_T \cdot \left(\Pr[x' = x_{1-i} \mid guess_R(T(x_0, x_1), R(i, x_i)) = x'] - \frac{1}{2}\right)$

ii. $+\beta_T \cdot \left(\Pr[fin(T(x_0, x_1), R(i)) = 1] - 1\right)$

- If T prefers the completion of the protocol … i decrease / ii increase
- If T prefers to protect the secret value … i increase / ii decrease

Analysis of GT-Security

- Comp(SOT-$\hat{\mathcal{F}}_{MT}$)-$\hat{\mathcal{F}}_{CP}$

Receiver R :

- R cannot enhance its own utility even if applying f for generating y_i, y_{1-i}.
- Compared to the case where R follows the protocol specifications, it results in decreasing the value

$$+\beta_R \cdot (\Pr[fin(T(x_0, x_1), R(i)) = 1] - 1)$$

Sender T :

- T can obtain the highest utility by following the protocol honestly.

The pair of strategies (σ_T, σ_R) is in Nash equilibrium.

The compiled protocol of SOT meets GT-security.

Conclusion

- We have proposed a compiler of two-party protocols in the LUC framework based on [CLOS02].

  - It transforms any two-party protocol secure against semi-honest adversaries into a protocol secure against malicious adversaries.

- We have shown the application of our compiler to an oblivious transfer protocol to achieve a primitive with both UC and GT security.

Thank You!