Concurrently Composable Non-Interactive Secure Computation

ASIACRYPT 2022

Andrew Morgan, Cornell University
Rafael Pass, Cornell Tech

Secure Two-Party Computation [Yao82]

R

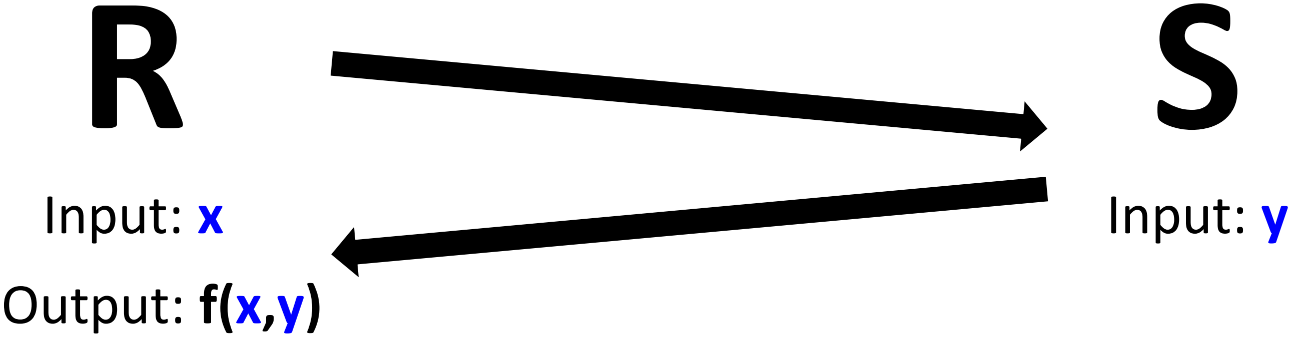
Input: **x**

S

Input: **y**

- Common output: **f(x,y)**
- Adversary **should learn nothing** besides the output.
  - Formally: **simulation-based security.**
  - We’ll consider fully **malicious adversaries** [GMW87].

Non-Interactive Secure Computation (NISC) [Yao82, IKO+11]



R

Input: **x**

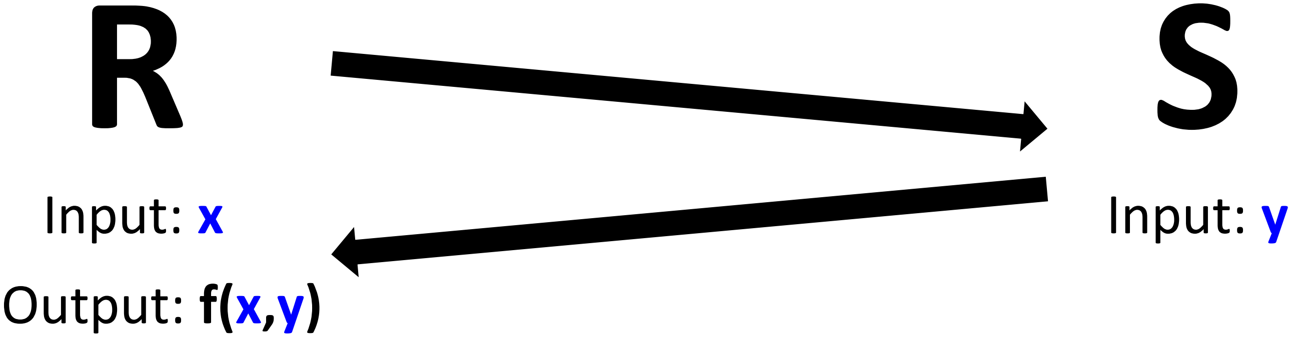
Output: **f(x,y)**

S

Input: **y**

- **Two rounds** with **one-sided functionality** (minimal).
- **Plain model: 4 rounds necessary and sufficient!** [KO04, GMPP16]
  - Need relaxed security definition: **superpolynomial-time simulation** [Pas03, PS04].

Non-Interactive Secure Computation (NISC)
[Yao82, IKO+11]



R

Input: **x**

Output: **f(x,y)**

S

Input: **y**

- **Two rounds** with **one-sided functionality** (minimal).

- [BGI+17]: **malicious SPS security in plain model.**
  - Based on subexponential versions of standard assumptions (DDH, QR, Nth Residuosity, or LWE [BD18]).

A Stronger Definition of Security

- For many applications, the standard definition of simulation-based security **isn't sufficient**.
  - **Concurrency:** security holds with many active instances.
  - **Composability:** security holds when used as a sub-protocol.

Universal Composability

- **Universal composability (UC)** framework [Can01] captures **composability** and **concurrency** guarantees.
- Security needs to hold against a stronger adversary: an **environment** which can run many instances, corrupt parties, and control communication between corrupted parties.
  - **Strictly stronger** than "normal" stand-alone security!
- **"Angel-based" UC security** [PS04, CLP10]**:** UC analogue of SPS
  - **Simulator and environment** both have access to a **superpolynomial-time helper H.**

Universal Composability

- **Problem:** UC security is **strictly stronger** than stand-alone security, making it far more difficult to prove results.

- **Best known round complexity** (without trusted setup):
  - Stand-alone SPS security: **two rounds** [BGI$^+$17]. *(minimal!)*
  - Concurrent SPS security: **two broadcasts** (two-sided/MPC). [AMR21, FJK21]
    - Still requires **three messages** for one-sided 2PC.
  - “Angel-based” (SPS) UC security: **(unspecified) constant rounds** [KMO14, GLP$^+$15, CLP17].

Main Question

Is concurrently composable NISC achievable with malicious SPS security in the plain model?

- **Open for even concurrent security!** *(for general functionalities)*

- **This work**: **NISC with angel-based UC security.**

Main Theorem

Theorem 1. Assuming the existence of:

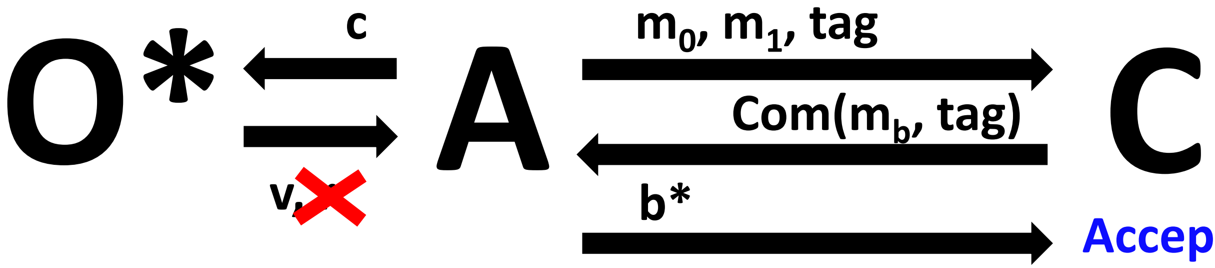
- **subexponentially secure maliciously (stand-alone) SPS-secure NISC**
- **subexponentially secure non-interactive CCA-secure commitments**

then there exists a **maliciously (oracle-aided) UC-secure NISC protocol in the plain model** for any polynomial-time computable functionality.

- **SPS-secure NISC** known from subexp. DDH/QR/LWE. [BGI+17]

Non-Interactive CCA-secure Commitments

- Tag-based commitments with:
  - **Binding**
  - **non-interactivity** (single commitment message)
  - **“chosen-ciphertext” hiding** against adversaries with a decommitment oracle on any tag but the challenge tag.



Accept if b = b* AND O* not queried on tag

  - **Weak CCA security: O*** returns only value.

Non-Interactive CCA-secure Commitments

- Tag-based commitments with:
  - **Binding**
  - **non-interactivity** (single commitment message)
  - **“chosen-ciphertext” hiding** against adversaries with a decommitment oracle on any tag but the challenge tag.

- **Known constructions require more sophisticated assumptions.**
  - [PPV08]: from **adaptive OWPs.**
  - [GKLW21]: from a variety of assumptions [LPS17, BL18, KK19] e.g., subexp. **hinting PRGs** *(can be based on CDH/LWE)*, subexp. **keyless CRHFs,** subexp. **time-lock puzzles.**

Necessity of Assumptions

- **Question:** Can we construct concurrently composable SPS-secure NISC from **weaker assumptions**?

- **Answer: NO**, we show that non-interactive (weakly) CCA-secure commitments are not only **sufficient**, but also **necessary**, for UC-secure NISC.

Necessity of Assumptions

Theorem 1. Assuming the existence of:

- **subexponentially secure maliciously (stand-alone) SPS-secure NISC**
- **subexponentially secure non-interactive CCA-secure commitments**

then there exists a **maliciously (oracle-aided) UC-secure NISC protocol in the plain model** for any polynomial-time computable functionality.

Theorem 2. Assuming the existence of:

- **maliciously (oracle-aided) UC-secure NISC (with perfect correctness) for the equality functionality**

then there exists **a non-interactive weakly CCA-secure commitment scheme.**

Construction in **Thm. 1** is from **nearly minimal assumptions!**

(subexp. security sufficient, poly-time necessary)

Constructing Our Protocol: Summary

- Begin by **leveraging the underlying NISC** to perform the computation.

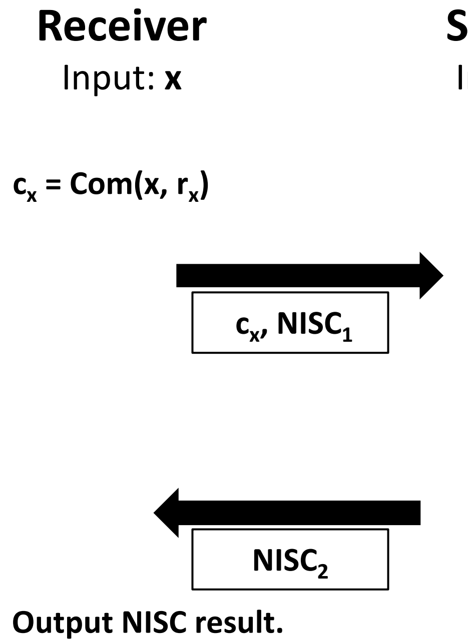
- **UC security:** simulator must **extract** the malicious party's input from their message.
  - **Restricted to poly-time:** **can't extract** using simulator for inner (SPS) NISC!
  - **Solution:** **Superpolynomial-time helper H** implements **CCA decommitment oracle O*** to extract from commitments.
  - **Tag-based security** guarantees that an adversary **cannot break honest parties' commitments.**

Constructing Our Protocol

- Start by considering a **malicious receiver.**

- **R** commits to **x** for extractability.

- **Need to verify commitment c_x:**
    - **Use the NISC for "interactive witness encryption"!**



Receiver
Input: **x**

Sender
Input: **y**

$c_x = Com(x, r_x)$

c_x, $NISC_1$

$NISC_2$

Output NISC result.

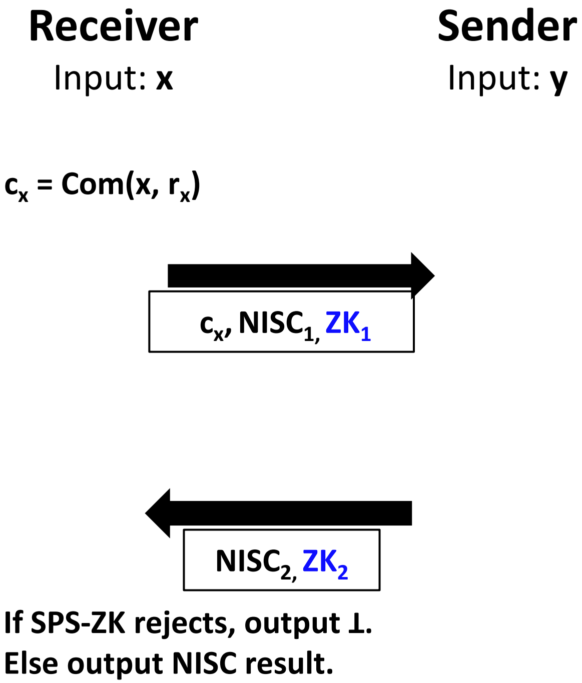
NISC

R's input: **(x, r_x)**
S's input: **(y, c_x)**
Functionality:
If $c_x \neq Com(x, r_x)$ return ⊥
Else return f(x,y).

Constructing Our Protocol

- Start by considering a **malicious receiver.**

- Still need to simulate sender message!
  - Begin by adding **SPS-ZK** (implementable with SPS-NISC) to verify sender message.



Receiver
Input: **x**

Sender
Input: **y**

$c_x = \mathrm{Com}(x, r_x)$

c_x, NISC_1, ZK_1

NISC_2, ZK_2

If SPS-ZK rejects, output ⊥.
Else output NISC result.

NISC

R's input: **(x, r_x)**
S's input: **(y, c_x)**
Functionality:
If $c_x \neq \mathrm{Com}(x, r_x)$ return ⊥
Else return $f(x,y)$.

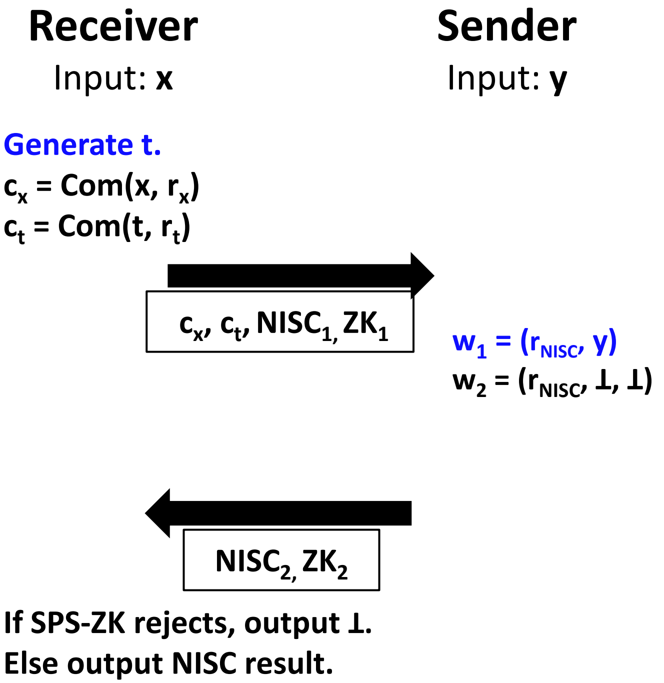
SPS-ZK

Statement: **(NISC_1, NISC_2, c_x)**
Witness: **(r_{NISC}, y)**

Proves that NISC_2 is correctly generated w.r.t. other inputs.

Constructing Our Protocol

- Start by considering a **malicious receiver.**
- Still need to simulate sender message **without knowing y**!
  - Add **two-track functionality.**
  - Sender can "program" NISC output when it uses the correct trapdoor.
  - **Honest sender** uses "real" witness $\mathbf{w_1}$.
  - **Simulator** can extract **t** and complete the ZK using $\mathbf{w_2}$.



Receiver

Input: **x**

Sender

Input: **y**

Generate t.

c_x = Com(x, r_x)

c_t = Com(t, r_t)

c_x, c_t, $NISC_1$, ZK_1

w_1 = (r_{NISC}, y)

w_2 = (r_{NISC}, ⊥, ⊥)

$NISC_2$, ZK_2

If SPS-ZK rejects, output ⊥.

Else output NISC result.

NISC

R's input: **(x, t, r_x, r_t)**

S's input: **(y, t', z, c_x ,c_t)**

Functionality:

If c_x ≠ Com(x, r_x)

or c_t ≠ Com(t, r_t) return ⊥

If t = t' return z

Else return f(x,y).

SPS-ZK

Statement: **($NISC_1$, $NISC_2$, c_x)**

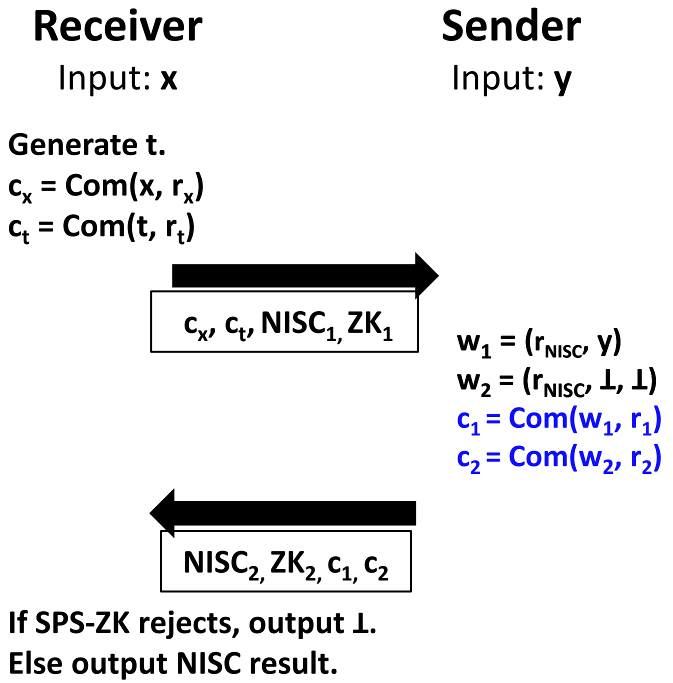
Witness: **w_1 = (r_{NISC}, y)**

AND w_2 = (r_{NISC}, t', z)

Proves that $NISC_2$ is correctly generated w.r.t. either w_1 or w_2.

Constructing Our Protocol

- Finally, consider a **malicious sender.**
- Need **extractability** for input **y**.
    - Add "argument of knowledge" by **committing to ZK witnesses.**



Receiver

Input: **x**

Sender

Input: **y**

Generate t.
c_x = Com(x, r_x)
c_t = Com(t, r_t)

c_x, c_t, $NISC_1$, ZK_1

w_1 = (r_{NISC}, y)
w_2 = (r_{NISC}, ⊥, ⊥)
c_1 = Com(w_1, r_1)
c_2 = Com(w_2, r_2)

$NISC_2$, ZK_2, c_1, c_2

If SPS-ZK rejects, output ⊥.
Else output NISC result.

NISC

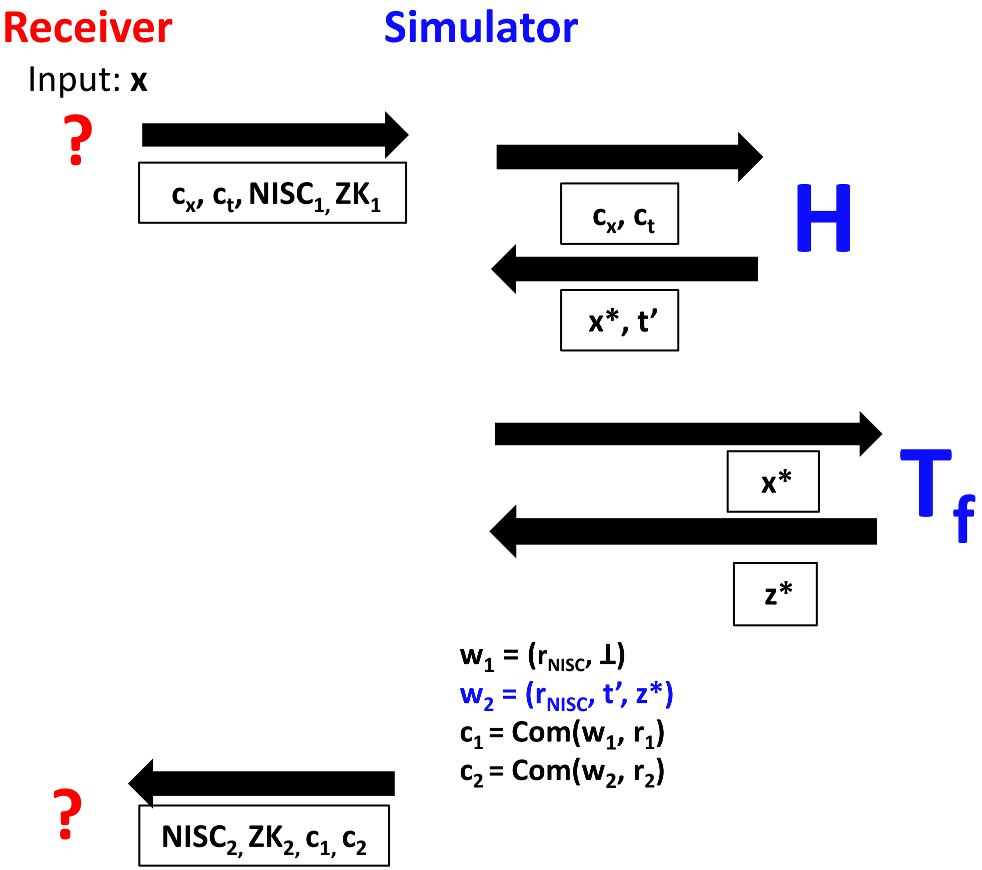
R's input: **(x, t, r_x, r_t)**
S's input: **(y, t', z, c_x ,c_t)**
Functionality:
If c_x ≠ Com(x, r_x)
or c_t ≠ Com(t, r_t) return ⊥
If t = t' return z
Else return f(x,y).

SPS-ZK

Statement: **($NISC_1$, $NISC_2$, c_x, c_1, c_2)**
Witness: **w_1 = (r_{NISC}, y), r_1**
AND w_2 = (r_{NISC}, t', z), r_2

Proves that $NISC_2$ is correctly generated w.r.t. either w_1 or w_2, AND the respective c is correct.

Simulating Our Protocol



Receiver

Input: **x**

Simulator

?

c_x, c_t, NISC$_1$, ZK$_1$

c_x, c_t

H

x^*, t'

x^*

T_f

z^*

$w_1 = (r_{NISC}, \perp)$

$w_2 = (r_{NISC}, t', z^*)$

$c_1 = Com(w_1, r_1)$

$c_2 = Com(w_2, r_2)$

?

NISC$_2$, ZK$_2$, c_1, c_2

NISC

R's input: **(x, t, r_x, r_t)**

S's input: **(y, t', z, c_x, c_t)**

Functionality:

If $c_x \neq Com(x, r_x)$
or $c_t \neq Com(t, r_t)$ return $\perp$
If $t = t'$ return z
Else return $f(x,y)$.

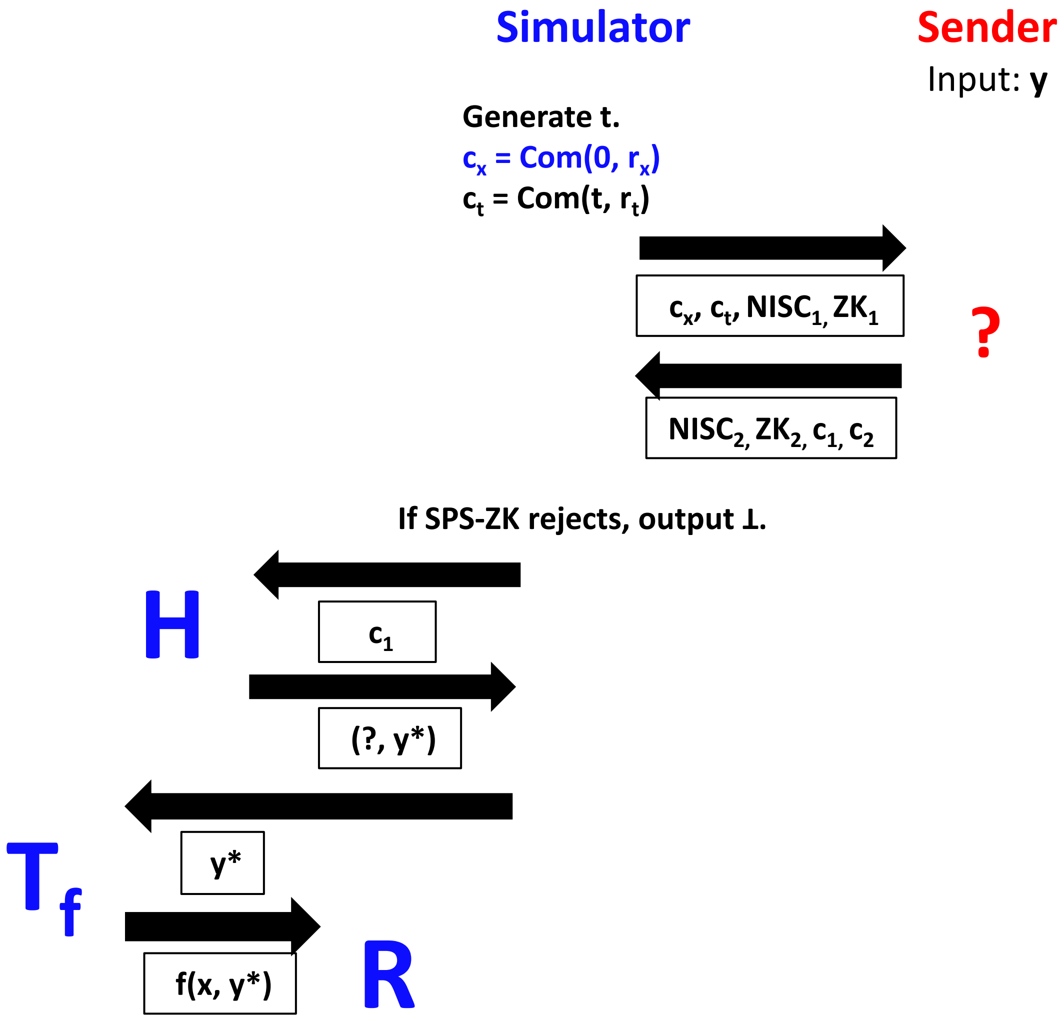
SPS-ZK

Statement: **$(NISC_1, NISC_2, c_x, c_1, c_2)$**

Witness: **$w_1 = (r_{NISC}, y), r_1$**
AND $w_2 = (r_{NISC}, t', z), r_2$

Proves that NISC$_2$ is correctly generated w.r.t. either w_1 or w_2, AND the respective c is correct.

Simulating Our Protocol



NISC

R's input: **(x, t, r_x, r_t)**

S's input: **(y, t', z, c_x ,c_t)**

Functionality:

If c_x ≠ Com(x, r_x)
or c_t ≠ Com(t, r_t) return ⊥
If t = t' return z
Else return f(x,y).

SPS-ZK

Statement: **(NISC$_1$, NISC$_2$, c_x, c_1, c_2)**

Witness: **w_1 = (r_{NISC}, y), r_1**
AND w_2 = (r_{NISC}, t', z), r_2

Proves that NISC$_2$ is correctly generated w.r.t. either w_1 or w_2, AND the respective c is correct.

Summary

- We present the first **concurrently and composably secure** NISC protocol.
    - Satisfies **angel-based UC security** against malicious adversaries in the plain model.
    - Constructed from **stand-alone secure NISC** and **CCA-secure non-interactive commitments.**

- Additionally, we show that the above building blocks are both **sufficient and necessary** for UC-secure NISC.
    - Construction and proof in the paper!