Composition of Cryptographic Protocols - Feasibility

Muthu Venkitasubramaniam
University of Rochester

Some slides borrowed from Manoj, Huijia, Abhishek and Rafael
Secure Multi-party Computation
[Yao, Goldreich-Micali-Wigderson]

**Goal:** Allow a set of distrustful parties to compute any functionality $f$ of their inputs, while preserving:

- **Correctness**
- **Privacy**

Even when no honest majority
Step 1: Specify goal as an functionality $f$ performed by an ideal trusted service

**GOAL = CORRECTNESS + PRIVACY**

Step 2: Security defined via protocol emulation in ideal world (a.k.a simulation)
Examples of Goals / Functionalities

**F\text{comp}**
1. Receive x from A and y from B
2. Output b = (x > y) to A and B

**F\text{ZK}**
1. Receive x, w from A
2. Output b = R(x, w) to B

**F\text{COIN}**
1. Toss coin c
2. Output c to A and B

**F\text{OT}**
1. Receive s_0, s_1 from A and b from B
2. Output s_b to B
Secure Minimum Spanning Tree [BS,sV]

Goal: Securely compute MST over the union of their edges
Secure Minimum Spanning Tree [BS,sV]

Goal: Securely compute MST over the union of their edges

- Suppose, we have secure protocol for $F_{comp}$
- Replace calls $F_{comp}$ to with secure protocol to get protocol for MST
- Does this mean this new protocol is secure?
The Classic Stand-Alone Model

One set of parties executing a single protocol in isolation
But, Life is CONCURRENT

Many parties running many different protocol executions
The Chess-master Problem

8am:
Lose!

8pm:
Lose!
Win at least 1 (or draw both)

What makes it hard?
- Concurrency
- Scheduling
- Unawarness
Same attack on protocols

E.g., real attacks on OpenSSL implementation [B’98]
A fundamental question: Composition

Is security preserved under protocol composition?
Security under composition

“Concurrently Secure” MPC

Why Care?

1. Composition occurs in real life
   ---Need concurrent security
2. Composition occurs in system design
   ---Want modular, simpler, solutions
3. Better understanding of security notions
   ---Various applications

MPC  PKE  Signature  Commitments  ZK  WH  ....
Concurrent Security

Protocol Executions

Trusted party

REAL

IDEAL
Both $A$ and $S$ required to be PPT.

Running the protocol $\pi$ in the concurrent setting is

Computing $f$ using a trusted party

$S$ simulates the view of $A$ &

the outputs of honest parties are the same in the two worlds

The UC Security [C01] Theorem:

If $\pi$ UC-implements $F_{\text{comp}}$ and $\rho^f$ UC-implements MST,

then $\rho^\pi$ UC-implements MST.

UC Composition Theorem:

If $\pi$ UC-implements $F$ comp and $\rho^f$ UC-implements MST,

then $\rho^\pi$ UC-implements MST.
UC Security [C01]

The UC Composition Theorem:

If $\pi$ UC-implements $f$ and $\rho^f$ UC-implements $G$,

then $\rho^\pi$ UC-implements $G$.

The strongest model of composition

1. Concurrent Security
2. Modular analysis

Theorem [CF, CKL, L]: It is impossible to achieve concurrent security for all “non-trivial functionalities”
Self-Composition

An unbounded number of instances of the **same** protocol

**Examples:** Self-Composable MPC ....
Non-Malleable Encryption
Concurrent Non-Malleable (NM) ZK
CMA-secure signature
Password authenticated key exchange (PAKE)
Impossibility Results

Impossibility of General Composition

Impossibility of Self Composition
Chosen Protocol Attack for OT

[BPS06, AGJPS12, GKOV12]

Impossibility of General Composition:
For every $\pi_{OT}$, there exists $\pi'_{OT}$ such that $\pi_{OT} \circ \pi'_{OT}$ breaks security of $\pi_{OT}$

Real Adv can learn honest party’s input, but Simulator cannot
Chosen Protocol Attack: **Real World**

**Attack**: Eve plays man-in-the-middle to learn \((s_0, s_1)\)
Chosen Protocol Attack: **Ideal World**

Attack Fails: With probability $\approx \frac{1}{2}$, Eve will ask for $s_{1-b}$.
From Impossibility of General Composition to Impossibility of Self-Composition

Want: Multiple Executions of $\pi_{OT}$ only (no $\pi'_{OT}$)

Replace with Garbled Circuits computing his Next-Message Functions

Give Garbled Circuits to Eve as Aux. Input
Problem: Who gets the GC Keys?

Eve should have keys to execute GCs on Alice’s messages, but can’t give her ALL keys.

Eve needs to run extra $\pi_{OT}$ executions with Alice to get “necessary” keys.

$S_0, S_1$

$\{GC_i\}$ Keys

$\pi_{OT}$

$GC_1$

$GC_n$
More Details

Concurrent OT Executions

Impossibility extends to all “non-trivial” functions by a reduction (in the concurrent setting) to OT
[AGJPS12,GKO12]

Real World: Eve executes GCs one-by-one to learn $s_0, s_1$

Ideal World: Attack fails as before due to security of GCs
What can we implement with Concurrent Security?

**Theorem [CF, CKL, L]:** It is impossible to achieve concurrent security for all “non-trivial functionalities”

**SOLUTION:** Get some “limited” help from a trusted party
Limited Trusted Help

Common Reference String (CRS)

Tamper Proof Hardware Model
Feasible in weaker models!

- **Honest Majority**
  [DM00,BGW88,BR89]

- **Tamper Proof Hardware**
  [K07,NW07,CGS08,MS08]

- **Common Reference String**
  [BFM88,D00,CLOS02,MY03,GO07,CPS07,DNO10]

- **Timing**
  [DNS98,G06,LKP05]

- **Public-Key Infrastructure**
  [JSI96,DN03,BCNP04,DNO10]

- **Augmented CRS (GUC)**
  [CDPW07]

**Concurrent Security** in a **Generalized UC model**
Intuition of Constructions

- General Composition
- Self Composition
Generalized UC [LPV09]

IDEAL

\[ z = F(x, y) \]

REAL

A framework of models

- Embeds most weaker models
- Close to UC, leverage previous results

1. Augmented Real World
2. Multi-session Ideal/Real World
Concurrent MPC in Generalized UC

Compilation for UC by [GMW87, BMR90, CLOS02, Pas04] assuming Semi-Honest OT

Implement multi-session ZK functionality
Implement multi-session ZK functionality

Design a “special” ZK protocol (P,V), s.t.
Simulate w/o witness (ZK)

Extract witness (AOK)

Concurrent ZKAOK (Concurrent Simulation-Extractability)
Extract witnesses from adv even when receiving simulated proofs
**Concurrent ZKAOK**

Extract witnesses from adv even when receiving simulated proofs.

*Have been studied a LOT!*

in Concurrent ZK [DNS98,RK99,PRS02,...]

Straight-line non-black-box simulation [Bar01,...]
How to get straight-line simulation?

By giving $S$ certain **SUPER-POWER** over $\text{Adv}$

= The ability to get a trapdoor

**UC-puzzle** + **Non-Malleability**
**Concurrent ZKAOK**

Extract witnesses from \text{adv} even when receiving simulated proofs

---

**A weaker notion:** **Fully concurrent ZKA** *(conc. simulation soundness)*

\text{Adv} cannot cheat even when receiving simulated proofs

---

Compilation from ZKA to ZKAOK

[BL02,PR03,Pas04,DNO10,MPR10,LPV13]
A weaker notion: **Fully concurrent ZKA**

Adv cannot cheat even when receiving simulated proofs

Concurrent Simulation

Security against MIM attacks

← UC-puzzles

← Non-Malleable Commitment
How to Cook Up Concurrent Security in Your Favorite Model X (CRS, PKA, SPS…)?

1. Instantiate a UC-puzzle using model X
2. Plug in
Common Reference String

Preprocessing:
Trusted Party samples a distribution D and publishes it

Protocol Execution:
Parties exchange messages

THEOREM [CLOS02]: Every goal can be implemented with concurrent security in the CRS model.
Property 1: Hard to solve with trusted setup
Property 2: Easy to solve by controlling setup in an undetectable way
PUZZLE (in CRS)

Property 1: Hard to solve with trusted setup

Property 2: Easy to solve by controlling setup in an undetectable way

“Impossible assuming factoring is hard”
PUZZLE (in CRS)

Rand. primes $p,q$

$CRS = pq$

"Impossible assuming factoring is hard"

COROLLARY: Any goal can be implemented with concurrent security in the CRS model
The State of UC Security

- **Possible**: with limited “trusted help”
  
  - Trusted set-up models: Honest majority [BGW88, CCD88, BR89, DM00], CRS [BFM, CLOS], PKI [BCNP], Timing model [DNS, KLP], Tamper-proof Hardware [K], ...

**Thm** [LPV09, LPV12] **For static corruption**, UC-Puzzles provide a crisp and tight characterization for any setup
Are we done?
The Classic Static Corruption
corrupt in the beginning

But, Life is NOT STATIC

Adaptive Corruption
corrupt adaptively
during execution
The State of UC Security

• **Possible**: with limited “trusted help”
  
  – Trusted set-up models: Honest majority [BGW88, CCD88, BR89, DM00], CRS [BFM, CLOS], PKI [BCNP], Timing model [DNS, KLP], Tamper-proof Hardware [K], ...

  | Thm | For static corruption, UC-Puzzles provide a crisp and tight characterization for any setup |
  | Thm | For adaptive corruption, (adaptive) UC-Puzzles are sufficient |
Are we done, now?

All the approaches we have seen require some minimal trusted setup
But, in LIFE, Who Can You TRUST?

NO ONE!
**In wonderland:** UC with TRUST

- Honest Majority [DM00,BGW88,BR89]
- Public Key Registration [BCNP04,LPV09,DNO10,LPV12]
- Tamper-Proof Hardware [Kat07,CGS08,LPV09,GISVW10,LPV12]
- CRS [Can01,CLOS02,CPS07,CDPW07,GO07,LPV09,DNO10,LPV12]
- Timing Model [DNS98,KLP05,LPV09,LPV12]
- Physically Uncloneable Functions [BFSK11,OSVW13]

**On earth:** relaxed security notions

- Input Indistinguishable Computation [MPR06,GGJS12]
- Super-Polynomial-time Simulation [Pas03,BS05,LPV09,LPV12,GGJS12]
- Angel-based security [PS04,MMY06,CLP10,LP12,GLPPS13,KMO14]
- Multiple-ideal query security [GJO10,GJ13,GGJ13]
Ideal Goal:

- Fully composable / concurrent (i.e. UC)
- Tolerates adaptive corruptions
- No trusted setup
- Standard (polynomial-time) hardness
- Black-box in the underlying primitives
Super-Poly Time Simulation (SPS) [P’03]

We know, super-poly-time security reduction is impossible.

Still, meaningful in many (most) cases.

Possible!

- **Static** [P03, PS04, BS05, LPV09, GGJS12, LPV12]
- **Adaptive** [BS05, DMRV13, V14]

But, using strong hardness assumptions.
Angel-Based Security [PS04]

Simulator and Adv. receive help from an angel

Angel: A restricted super-poly-time oracle

Possible w/ CRH [PS04, MMY06, BS05]

But, even stronger assumptions

e.g. Adaptively hard CRH
Angel-Based Security [PS04]
Simulator and Adv. receive help from an angel

Possible under polynomial-time assumptions!
[CLP10]
Angel: Decommitment Oracle
New Primitive: CCA-secure Commitments
CCA-Secure Commitments [CLP10]

Chosen-Commitment-Attack (CCA) security:

Either \( A \) copies the left identifier to the right
Or LHS is hiding --- view of \( A \) indistinguishable
Chosen-Commitment-Attack (CCA) security:

**Theorem** [CLP10, LP11, GLPPS14, K14] Assuming OWFs

∃ O(log²n)-round Blackbox CCA Com.

**Theorem** [CLP10, LP11] Assuming CCA Com. and OT

∃ BB construction static (G)UC for any functionality
Can we get Angel-Based Adaptive UC-Security?

- Implies super-polynomial security, i.e. no setup
- Analyze single instance and guarantee composition (GUC [CDPW07])
- Possibility of polynomial-time assumptions by relying on rewinding based techniques

**Bottleneck 1:** [GS12] Rewinding based techniques don’t compose well

**Bottleneck 2:** Adaptive Composable Commitments implies selective opening security IMPOSSIBLE! [ORSV11]

**Our Approach:** Adaptive CCA-Secure Coin-Tossing
Def 1: CCA-Secure Coin-Tossing $\langle I, R \rangle$

Chosen-Coin-Attack (CCA) security:

**Angel:** $O$ is a biasing oracle

**Security?** Simulate a coin with $A^O$
Def 1: CCA-Secure Coin-Tossing \( \langle I, R \rangle \)

Chosen-Coin-Attack (CCA) security:

Angel: \( O \) is a biasing oracle

Security? Simulate a coin with \( A^O \)
Def 1: CCA-Secure Coin-Tossing \( \langle I, R \rangle \)

Chosen-Coin-Attack (CCA) security:
Def 1: CCA-Secure Coin-Tossing $\langle I, R \rangle$

Chosen-Coin-Attack (CCA) security:

Either A copies the left identifier to the right or corrupts
Or LHS is simulatable --- view of A indistinguishable
Theorem 1: Assuming **CCA Coin-Tossing** and sim. PKE, adaptive UC-realize any (well-formed) functionality.

Theorem 2: Assuming **OWFs**, $O(n^\varepsilon)$-round **CCA Coin-Tossing**
Adaptive UC Security without setup [HV16]

✓ Polynomial-time assumptions (OWF+SimPKE)
✓ Fully black-box

``Strongest” definition of concurrent adaptive security realizable without set-up
Open Problems

• General feasibility results are not practical
  – Many number of rounds
  – High communication complexity
  – Often non-black-box in the underlying cryptographic primitive

• [HV16] UC feasibility in the CRS under minimal assumptions in a black-box way (static & adap.)

• [HPV16, HPV17] UC feasibility in the Tamper Proof Hardware model (static & adap.)

Need: A unified “practical” way of getting UC
THANK YOU