

Universally Composable Commitments

Canetti and Fischlin, 2001

Daniel Shahaf

Tel-Aviv University

June 2009

Table of Contents

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

Summary

- 1 The UC Framework
- 2 The Commitment Functionality
- 3 A Better Construction

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

Summary

1 The UC Framework

UC Review

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

Summary

- Back to Ran's presentation
- ideal functionality, public delayed output, \mathcal{F}_{COM} -hybrid, $\mathcal{F}_{\text{AUTH}}$ -hybrid, plain model, securely realizes, ...

2 The Commitment Functionality

- Goal: Defining the Ideal
- Applications: Zero-Knowledge from Commitments
- Realizations in the Plain Model
- Realizations in the CRS Model

Defining The Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Definition (The ideal bit commitment functionality \mathcal{F}_{COM})

Given parties P_1, \dots, P_n and adversary \mathcal{S} :

- 1 Upon receiving input $\langle \text{Commit}, sid, P_i, P_j, b \rangle$ from party P_i , **record** the value b and **send** delayed public output $\langle \text{Receipt}, sid, P_i, P_j \rangle$ to P_j **and to** \mathcal{S} .
Ignore subsequent Commit messages.

Defining The Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Definition (The ideal bit commitment functionality \mathcal{F}_{COM})

Given parties P_1, \dots, P_n and adversary \mathcal{S} :

- 1 Upon receiving input $\langle \text{Commit}, sid, P_i, P_j, b \rangle$ from party P_i , **record** the value b and **send** delayed public output $\langle \text{Receipt}, sid, P_i, P_j \rangle$ to P_j **and** to \mathcal{S} .
Ignore subsequent Commit messages.
- 2 Upon receiving input $\langle \text{Open}, sid, P_i, P_j \rangle$ from P_i : if a value b was previously recorded, then **send** a public delayed output $\langle \text{Open}, sid, P_i, P_j, b \rangle$ to P_j **and** \mathcal{S} .

Defining The Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Definition (The ideal bit commitment functionality \mathcal{F}_{COM})

Given parties P_1, \dots, P_n and adversary \mathcal{S} :

- 1 Upon receiving input $\langle \text{Commit}, sid, P_i, P_j, b \rangle$ from party P_i , **record** the value b and **send** delayed public output $\langle \text{Receipt}, sid, P_i, P_j \rangle$ to P_j **and** to \mathcal{S} .
Ignore subsequent Commit messages.
- 2 Upon receiving input $\langle \text{Open}, sid, P_i, P_j \rangle$ from P_i : if a value b was previously recorded, then **send** a public delayed output $\langle \text{Open}, sid, P_i, P_j, b \rangle$ to P_j **and** \mathcal{S} .
- 3 Upon receiving $\langle \text{Corrupt-committer}, sid \rangle$ from \mathcal{S} , **send** the recorded value b to the adversary.

Defining The Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Definition (The ideal bit commitment functionality \mathcal{F}_{COM})

Given parties P_1, \dots, P_n and adversary \mathcal{S} :

- 1 Upon receiving input $\langle \text{Commit}, sid, P_i, P_j, b \rangle$ from party P_i , **record** the value b and **send** delayed public output $\langle \text{Receipt}, sid, P_i, P_j \rangle$ to P_j **and** to \mathcal{S} .
Ignore subsequent Commit messages.
- 2 Upon receiving input $\langle \text{Open}, sid, P_i, P_j \rangle$ from P_i : if a value b was previously recorded, then **send** a public delayed output $\langle \text{Open}, sid, P_i, P_j, b \rangle$ to P_j **and** \mathcal{S} .
- 3 Upon receiving $\langle \text{Corrupt-committer}, sid \rangle$ from \mathcal{S} , **send** the recorded value b to the adversary. As long as Receipt was not yet written on P_j 's tape, **allow** the adversary to **modify** the recorded value.

Discussion and Variants

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

- Good for a single, one-bit commitment only.
 - But composable...
- Captures adaptive corruptions.
- No error messages.
- No acknowledgements.
- Committing is public.
- Opened values are public.
- Captures many strong notions of security.
 - Unbounded concurrent composition (universal composition).
 - Non-malleable? Perfect hiding? Perfect binding?

Defining Zero-Knowledge

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Let R be a binary relation.

Definition (The ideal zero-knowledge functionality \mathcal{F}_{ZK}^R)

Given parties P_1, \dots, P_n and adversary \mathcal{S} :

Defining Zero-Knowledge

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Let R be a binary relation.

Definition (The ideal zero-knowledge functionality \mathcal{F}_{ZK}^R)

Given parties P_1, \dots, P_n and adversary \mathcal{S} :

- 1 Upon receiving input $\langle \text{verifier}, sid, P_i, P_j, x \rangle$
from party P_i , **send** delayed public output
 $\langle \text{verifier}, sid, P_i, P_j, x \rangle$ **to** \mathcal{S} .
Ignore subsequent verifier messages.

Defining Zero-Knowledge

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Let R be a binary relation.

Definition (The ideal zero-knowledge functionality \mathcal{F}_{ZK}^R)

Given parties P_1, \dots, P_n and adversary \mathcal{S} :

- 1 Upon receiving input $\langle \text{verifier}, sid, P_i, P_j, x \rangle$ from party P_i , **send** delayed public output $\langle \text{verifier}, sid, P_i, P_j, x \rangle$ to \mathcal{S} .
Ignore subsequent verifier messages.
- 2 Upon receiving input $\langle \text{prover}, sid, P_j, P_i, x', w \rangle$ from P_j , **send** a public delayed output $\langle sid, v \rangle$ to P_i **and** \mathcal{S} , where $v = 1$ if and only if $x = x'$ and $R(x, w)$.

Applications: Zero-Knowledge

Universally
Composable
Commitments

Theorem (informal)

UC commitments imply 'strong' UC ZK.

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

**Applications:
Zero-Knowledge
from
Commitments**

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Applications: Zero-Knowledge

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Theorem (informal)

UC commitments imply 'strong' UC ZK.

Theorem (more formal)

\mathcal{F}_{ZK} (for Hamiltonicity) can be realized in the \mathcal{F}_{COM} -hybrid model. Further, the realization:

- does not need any cryptographic assumptions.
- works with respect to unbounded adversaries and environments.

Applications: Zero-Knowledge

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Theorem (informal)

UC commitments imply 'strong' UC ZK.

Theorem (more formal)

\mathcal{F}_{ZK} (for Hamiltonicity) can be realized in the \mathcal{F}_{COM} -hybrid model. Further, the realization:

- does not need any cryptographic assumptions.
- works with respect to unbounded adversaries and environments.

Proof Summary

- (Blum Hamiltonicity Protocol)ⁿ, using \mathcal{F}_{COM} .
- Given \mathcal{A} against \mathcal{F}_{COM} -hybrid BHP, simulator runs \mathcal{A} and acts as its \mathcal{F}_{COM} .

The Plain Model

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

**Realizations in
the Plain Model**

Realizations in
the CRS Model

A Better
Construction

Summary

Theorem

There does not exist a **bilateral**, **terminating** protocol that securely realizes \mathcal{F}_{COM} in the **plain** model.

The Plain Model

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal
Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model
Realizations in
the CRS Model

A Better
Construction

Summary

Theorem

There does not exist a **bilateral**, **terminating** protocol that securely realizes \mathcal{F}_{COM} in the **plain** model.

Proof Idea

Suppose \mathcal{A} merely channels the environment, who corrupts the committer **before** he commits. Due to \mathcal{F}_{COM} 's design, in order to simulate the adversary's commitment, \mathcal{S} has to be able to execute **extraction**. Hence, so can a honest receiver. In contradiction to secrecy.

The Plain Model

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal
Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model
Realizations in
the CRS Model

A Better
Construction

Summary

Theorem

There does not exist a **bilateral**, **terminating** protocol that securely realizes \mathcal{F}_{COM} in the **plain** model.

Proof Idea

Suppose \mathcal{A} merely channels the environment, who corrupts the committer **before** he commits. Due to \mathcal{F}_{COM} 's design, in order to simulate the adversary's commitment, \mathcal{S} has to be able to execute **extraction**. Hence, so can a honest receiver. In contradiction to secrecy.

Note

The proof will be **unconditional** (no computational assumptions).

Formal Proof of Impossibility

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Proof

- Suppose \mathcal{Z} instructs \mathcal{A} to corrupt the committer before the commitment takes place, and $\exists \mathcal{S}$ such that $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\text{COM}}, \mathcal{S}, \mathcal{Z}}$.
 - Note: this is a stronger order of quantifiers.
 - \mathcal{S} expects to see \mathcal{A} 's commitment (from which it will **extract** the committed value and pass it to its \mathcal{F}_{COM}).

Formal Proof of Impossibility

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Proof

- Suppose \mathcal{Z} instructs \mathcal{A} to corrupt the committer before the commitment takes place, and $\exists \mathcal{S}$ such that $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\text{COM}}, \mathcal{S}, \mathcal{Z}}$.
 - Note: this is a stronger order of quantifiers.
 - \mathcal{S} expects to see \mathcal{A} 's commitment (from which it will **extract** the committed value and pass it to its \mathcal{F}_{COM}).
- Construct new environment \mathcal{Z}' and adversary \mathcal{A}' such that no simulator will work.
- \mathcal{A}' channels \mathcal{Z}' , who corrupts **the receiver**.

Formal Proof of Impossibility

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Proof

- Suppose \mathcal{Z} instructs \mathcal{A} to corrupt the committer before the commitment takes place, and $\exists \mathcal{S}$ such that $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\text{COM}}, \mathcal{S}, \mathcal{Z}}$.
 - Note: this is a stronger order of quantifiers.
 - \mathcal{S} expects to see \mathcal{A} 's commitment (from which it will **extract** the committed value and pass it to its \mathcal{F}_{COM}).
- Construct new environment \mathcal{Z}' and adversary \mathcal{A}' such that no simulator will work.
- \mathcal{A}' channels \mathcal{Z}' , who corrupts **the receiver**.
- Upon receiving a commitment c , the receiver **runs \mathcal{S}** and **acts as its \mathcal{F}_{COM}** .
- The simulated \mathcal{S} tells **its \mathcal{F}_{COM}** what c hides.

Formal Proof of Impossibility

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model
Realizations in
the CRS Model

A Better
Construction

Summary

Proof

- Suppose \mathcal{Z} instructs \mathcal{A} to corrupt the committer before the commitment takes place, and $\exists \mathcal{S}$ such that $\text{REAL}_{\pi, \mathcal{A}, \mathcal{Z}} \approx \text{IDEAL}_{\mathcal{F}_{\text{COM}}, \mathcal{S}, \mathcal{Z}}$.
 - Note: this is a stronger order of quantifiers.
 - \mathcal{S} expects to see \mathcal{A} 's commitment (from which it will **extract** the committed value and pass it to its \mathcal{F}_{COM}).
- Construct new environment \mathcal{Z}' and adversary \mathcal{A}' such that no simulator will work.
- \mathcal{A}' channels \mathcal{Z}' , who corrupts **the receiver**.
- Upon receiving a commitment c , the receiver **runs \mathcal{S}** and **acts as its \mathcal{F}_{COM}** .
- The simulated \mathcal{S} tells **its \mathcal{F}_{COM}** what c hides.
- \mathcal{A}' has broken the 'hiding' property. □

Remaining Alternatives

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Remaining options:

- Weaken \mathcal{F}_{COM} .
- Non-bilateral protocols.
- Non-terminating protocols.
- Add **trust assumptions** on top of the 'plain' model.

Remaining Alternatives

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Remaining options:

- Weaken \mathcal{F}_{COM} .
- Non-bilateral protocols.
- Non-terminating protocols.
- Add **trust assumptions** on top of the 'plain' model.
 - "Suppose we had a **common reference string (CRS)**..."
 - Seems useful.
 - How to formalize in the model?

Modelling a Common Reference String

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

**Realizations in
the CRS Model**

A Better
Construction

Summary

- Formal specification?
- How to access the string?

Modelling a Common Reference String

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

- Formal specification?
- How to access the string?
- Answer: define it as an **ideal functionality**.
 - Define an ideal functionality to strengthen the basic model.
 - Other options: synchronous communications, **authenticated** communications, set-up assumptions, ...

Desired Properties of a CRS Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

- Where is it needed?
- Which parties have access?

Desired Properties of a CRS Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

- Where is it needed?
 - Not needed in the ideal process.
 - ⇒ Encapsulate CRS as **black-box** part of the protocol implementation.

- Which parties have access?

Desired Properties of a CRS Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

- Where is it needed?
 - Not needed in the ideal process.
 - ⇒ Encapsulate CRS as **black-box** part of the protocol implementation.
 - Environment **can't** access it.
 - Simulator may **choose** it for the simulated adversary.
 - Simulator can cheat (trapdoors).
- Which parties have access?

Desired Properties of a CRS Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

- Where is it needed?
 - Not needed in the ideal process.
 - ⇒ Encapsulate CRS as **black-box** part of the protocol implementation.
 - Environment **can't** access it.
 - Simulator may **choose** it for the simulated adversary.
 - Simulator can cheat (trapdoors).
- Which parties have access?
 - Want: everyone!
 - But: then \mathcal{Z} knows it — too strong (simulator can't cheat)
 - ⇒ CRS is **private** to **this protocol instance**. (Shared among the parties, but not known to the 'outside world'.)

The CRS Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Definition (The ideal CRS functionality \mathcal{F}_{CRS})

Given a distribution D and a set \mathcal{P} of parties:

- 1 Ignore all messages not coming from a party in \mathcal{P} .
- 2 On input $\langle \text{CRS}, \text{sid} \rangle$ from P : if no value v is recorded, then **choose** $v \leftarrow D$ and **record** it. **Send** v as a public delayed output to P .

A One-Time CRS Protocol

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

**Realizations in
the CRS Model**

A Better
Construction

Summary

Theorem

There exists a protocol that securely realizes \mathcal{F}_{COM} in the $\langle \mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{CRS}} \rangle$ -hybrid model.

A One-Time CRS Protocol

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Theorem

There exists a protocol that securely realizes \mathcal{F}_{COM} in the $\langle \mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{CRS}} \rangle$ -hybrid model.

Practical Considerations

- The composition theorem requires each copy of the protocol to have its own CRS.
- ⇒ Length of CRS (number of \mathcal{F}_{CRS} instances) is **linear** in the number of commitments.

A One-Time CRS Protocol

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

Goal: Defining
the Ideal

Applications:
Zero-Knowledge
from
Commitments

Realizations in
the Plain Model

Realizations in
the CRS Model

A Better
Construction

Summary

Theorem

There exists a protocol that securely realizes \mathcal{F}_{COM} in the $\langle \mathcal{F}_{\text{AUTH}}, \mathcal{F}_{\text{CRS}} \rangle$ -hybrid model.

Practical Considerations

- The composition theorem requires each copy of the protocol to have its own CRS.
- ⇒ Length of CRS (number of \mathcal{F}_{CRS} instances) is **linear** in the number of commitments.
- Want: more efficient construction.
- Problem: construction of **what?**

- 3 A Better Construction
 - The Reusable Commitment Functionality
 - A Realizing Protocol
 - Proof of Security

Allowing CRS Reuse

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- Goal: more efficient CRS reuse.
- Problem: can't reuse **across** applications of the composition theorem.

Allowing CRS Reuse

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality
A Realizing
Protocol
Proof of Security

Summary

- Goal: more efficient CRS reuse.
- Problem: can't reuse **across** applications of the composition theorem.
- Solution: reuse **intra**-application, i.e., intra-functionality.

Definition (The ideal multiple commitments functionality $\mathcal{F}_{\text{MCOM}}$)

Identical to \mathcal{F}_{COM} , but also takes a **commitment id** (*cid*) parameter from the committer (and passes it to the receiver).

- No ordering restrictions.

Defining The Functionality

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality
A Realizing
Protocol
Proof of Security

Summary

Definition (The ideal bit commitments functionality $\mathcal{F}_{\text{MCOM}}$)

Given parties P_1, \dots, P_n and adversary \mathcal{S} :

- 1 Upon receiving input $\langle \text{Commit}, sid, cid, P_i, P_j, b \rangle$ from party P_i , **record** the value b and **send** delayed public output $\langle \text{Receipt}, sid, cid, P_i, P_j \rangle$ to P_j and to \mathcal{S} .
Ignore subsequent Commit messages.
- 2 Upon receiving input $\langle \text{Open}, sid, cid, P_i, P_j \rangle$ from P_i : if a value b was previously recorded, then **send** a public delayed output $\langle \text{Open}, sid, cid, P_i, P_j, b \rangle$ to P_j and \mathcal{S} .
- 3 Upon receiving $\langle \text{Corrupt-committer}, sid, cid \rangle$ from \mathcal{S} , **send** the recorded value b to the adversary. As long as Receipt was not yet written on P_j 's tape, **allow** the adversary to **modify** the recorded value.

Observations

Realizations of $\mathcal{F}_{\text{MCOM}}$ must:

- be **extractible** (to simulate corrupted/relay committer).
- be **equivocable** (to simulate corrupted/relay receiver).
- ensure **intra-instance** (inter-*cid*) security ‘manually’ (composition theorem doesn’t apply).

Note

$\mathcal{F}_{\text{MCOM}}$ **cannot** be realized in the **plain** model.

Extractibility and Equivocability: Requirements

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- The scheme should be both **extractible** and **equivocable** by the simulator \mathcal{S} .
- But **not** by the parties!

Extractibility and Equivocability: Requirements

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- The scheme should be both extractible and equivocable **by the simulator \mathcal{S}** .
 - But not by the parties!
- ⇒ Simulator should have some **trapdoor information** that the parties don't know:
Parties will encrypt, simulator will decrypt.

Extractibility and Equivocability: Requirements

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- The scheme should be both extractible and equivocal by the simulator \mathcal{S} .
 - But not by the parties!
- ⇒ Simulator should have some **trapdoor information** that the parties don't know:
Parties will encrypt, simulator will decrypt.
- Q: How is encryption useful without ability to decrypt?
A: Show the encryption coins.

Extractibility and Equivocability: Requirements

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- The scheme should be both extractible and equivocable **by the simulator \mathcal{S}** .
 - But not by the parties!
- ⇒ Simulator should have some **trapdoor information** that the parties don't know:
Parties will encrypt, simulator will decrypt.
- Q: How is encryption useful without ability to decrypt?
A: Show the encryption coins.
- Q: How to give \mathcal{S} that extra information?
A: Put the public keys in the CRS (which \mathcal{S} chooses).

Extractibility and Equivocability: Implementation

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

**A Realizing
Protocol**
Proof of Security

Summary

- Use claw-free trapdoor permutations (CFTDP).
 - The trapdoor allows **inversion**.
 - Parties **won't** know the trapdoor.

Extractibility and Equivocability: Implementation

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- Use claw-free trapdoor permutations (CFTDP).
 - The trapdoor allows **inversion**.
 - Parties **won't** know the trapdoor.
- Commit to b : send $f_b(x)$ for random x .
Decommit: send x and b .

Extractibility and Equivocability: Implementation

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- Use claw-free trapdoor permutations (CFTDP).
 - The trapdoor allows **inversion**.
 - Parties **won't** know the trapdoor.
- Commit to b : send $f_b(x)$ for random x .
Decommit: send x and b .
- Equivocable (**given** the trapdoor), but not extractible.

Extractibility and Equivocability: Implementation

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- Use claw-free trapdoor permutations (CFTDP).
 - The trapdoor allows **inversion**.
 - Parties **won't** know the trapdoor or the secret key.
- Commit to b : send $f_b(x)$ for random x .
Decommit: send x and b .
- Equivocable (**given** the trapdoor), but not extractible.
- Commit to b : send $f_b(x)$ and $c = \text{ENC}(x)$ for random x .
Decommit: send x , encryption coins, and b .

Extractibility and Equivocability: Implementation

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality
A Realizing
Protocol
Proof of Security

Summary

- Use claw-free trapdoor permutations (CFTDP).
 - The trapdoor allows **inversion**.
 - Parties **won't** know the trapdoor or the secret key.
- Commit to b : send $f_b(x)$ for random x .
Decommit: send x and b .
- Equivocable (**given** the trapdoor), but not extractible.
- Commit to b : send $f_b(x)$ and $c = \text{ENC}(x)$ for random x .
Decommit: send x , encryption coins, and b .
- Extractible (**given** the sk), but not equivocable.

Extractibility and Equivocability: Implementation

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality
A Realizing
Protocol
Proof of Security

Summary

- Use claw-free trapdoor permutations (CFTDP).
 - The trapdoor allows **inversion**.
 - Parties **won't** know the trapdoor or the secret key.
- Commit to b : send $f_b(x)$ for random x .
Decommit: send x and b .
- Equivocable (**given** the trapdoor), but not extractible.
- Commit to b : send $f_b(x)$ and $c = \text{ENC}(x)$ for random x .
Decommit: send x , encryption coins, and b .
- Extractible (**given** the sk), but not equivocable.
- Commit to b : send $f_b(x)$, $c_b = \text{ENC}(x)$,
and $c_{1-b} = \text{ENC}(0)$ for random x .
Decommit: send x , encryption coins **of** x , and b .

Extractibility and Equivocability: Implementation

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality
A Realizing
Protocol
Proof of Security

Summary

- Use claw-free trapdoor permutations (CFTDP).
 - The trapdoor allows **inversion**.
 - Parties **won't** know the trapdoor or the secret key.
- Commit to b : send $f_b(x)$ for random x .
Decommit: send x and b .
- Equivocable (**given** the trapdoor), but not extractible.
- Commit to b : send $f_b(x)$ and $c = \text{ENC}(x)$ for random x .
Decommit: send x , encryption coins, and b .
- Extractible (**given** the sk), but not equivocable.
- Commit to b : send $f_b(x)$, $c_b = \text{ENC}(x)$,
and $c_{1-b} = \text{ENC}(0)$ for random x .
Decommit: send x , encryption coins **of** x , and b .
 - Equivocate: encrypt claws and open **one** set of coins.
 - Extract: decrypt and apply f_b .

More Problems

Universally
Composable
Commitments

How to prevent simple forwarding attacks?

The UC
Framework

How to prevent mauling the encrypted party's name?

The
Commitment
Functionality

A Better
Construction

What to do with the $\text{ENC}(0)$ set of coins?

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

More Problems

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

How to prevent simple forwarding attacks?

- Encrypt party's name along with the preimage.

How to prevent mauling the encrypted party's name?

What to do with the $\text{ENC}(0)$ set of coins?

More Problems

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

How to prevent simple forwarding attacks?

- Encrypt party's name along with the preimage.

How to prevent mauling the encrypted party's name?

- Use LR-CCA (non-malleable) encryption.

What to do with the $\text{ENC}(0)$ set of coins?

More Problems

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality
A Realizing
Protocol
Proof of Security

Summary

How to prevent simple forwarding attacks?

- Encrypt party's name along with the preimage.

How to prevent mauling the encrypted party's name?

- Use LR-CCA (non-malleable) encryption.

What to do with the $\text{ENC}(0)$ set of coins?

- If kept, \mathcal{A} will expect them upon corrupting, and \mathcal{S} will have to manufacture them for its equivocations.

More Problems

Universally
Composable
Commitments

How to prevent simple forwarding attacks?

- Encrypt party's name along with the preimage.

The UC
Framework

How to prevent mauling the encrypted party's name?

- Use LR-CCA (non-malleable) encryption.

The
Commitment
Functionality

A Better
Construction

What to do with the $\text{ENC}(0)$ set of coins?

- If kept, \mathcal{A} will expect them upon corrupting, and \mathcal{S} will have to manufacture them for its equivocations.
- Solutions:
 - **Erase** those coins.
 - Sample 'ciphertexts' **without** running ENC , in a way that allows \mathcal{S} to find the oblivious-sampling coins for its (non-dummy) ciphertext.

The Reusable
Commitment
Functionality
A Realizing
Protocol
Proof of Security

Summary

The Protocol $\text{UCC}_{\text{ReUse}}$

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

**A Realizing
Protocol**
Proof of Security

Summary

Common reference string:

- public key for a claw-free TDP pair f_0, f_1
- public key for an LR-CCA-secure encryption scheme ENC

The Protocol $\text{UCC}_{\text{ReUse}}$

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality
A Realizing
Protocol
Proof of Security

Summary

Common reference string:

- public key for a claw-free TDP pair f_0, f_1
- public key for an LR-CCA-secure encryption scheme ENC

Code for party P_i to commit to b :

- $y = f_b(x)$ for random x ;
- $c_b = \text{ENC}(\langle x, P_i \rangle)$ with coins r_b ;
- $c_{1-b} = \text{ENC}(\langle 0, P_i \rangle)$ with coins r_{1-b} ;
- Erase r_{1-b} ;
- Send $\langle y, c_0, c_1 \rangle$.

The Protocol $\text{UCC}_{\text{ReUse}}$

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality
A Realizing
Protocol
Proof of Security

Summary

Common reference string:

- public key for a claw-free TDP pair f_0, f_1
- public key for an LR-CCA-secure encryption scheme ENC

Code for party P_i to commit to b :

- $y = f_b(x)$ for random x ;
- $c_b = \text{ENC}(\langle x, P_i \rangle)$ with coins r_b ;
- $c_{1-b} = \text{ENC}(\langle 0, P_i \rangle)$ with coins r_{1-b} ;
- Erase r_{1-b} ;
- Send $\langle y, c_0, c_1 \rangle$.

Code for party P_j to verify an $\text{open}(b, x, r_b)$:

- Fail unless $\langle x, P_i \rangle$ encrypts to c_b under coins r_b .
- Fail unless $y = f_b(x)$.
- Declare that b is the opened value.

Overview

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol

Proof of Security

Summary

Theorem

$\text{UCC}_{\text{ReUse}}$, with LR-CPA-secure encryption,
securely realizes \mathcal{F}_{COM} .

Theorem

$\text{UCC}_{\text{ReUse}}$, with LR-CCA (non-malleable) encryption,
securely realizes $\mathcal{F}_{\text{MCOM}}$.

Overview

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

Theorem

$\text{UCC}_{\text{ReUse}}$, with LR-CPA-secure encryption,
securely realizes \mathcal{F}_{COM} .

Theorem

$\text{UCC}_{\text{ReUse}}$, with LR-CCA (non-malleable) encryption,
securely realizes $\mathcal{F}_{\text{MCOM}}$.

Proof Plan

Define hybrids for \mathcal{Z} 's view:

$H_{\text{real/genuine}}$ Parties running $\text{UCC}_{\text{ReUse}}$ with adversary \mathcal{A} in the
real-life model.

$H_{\text{ideal/fake}}$ Parties running $\mathcal{F}_{\text{MCOM}}$ with adversary \mathcal{S} in the
ideal model.

Goal: show that $H_{\text{r/g}}$ and $H_{\text{i/f}}$ are indistinguishable.

The Simulator

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol

Proof of Security

Summary

The simulator \mathcal{S} :

- 1 **selects** a CRS whose trapdoor and sk it knows.

The Simulator

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol

Proof of Security

Summary

The simulator \mathcal{S} :

- 1 **selects** a CRS whose trapdoor and sk it knows.
- 2 when told (by $\mathcal{F}_{\text{MCOM}}$) of a honest commitment, commits **equivocally** to \mathcal{A} .
- 3 when told (by $\mathcal{F}_{\text{MCOM}}$) of a honest open(b), opens (equivocally) to \mathcal{A} .

The Simulator

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

The simulator \mathcal{S} :

- 1 **selects** a CRS whose trapdoor and sk it knows.
- 2 when told (by $\mathcal{F}_{\text{MCOM}}$) of a honest commitment, commits **equivocally** to \mathcal{A} .
- 3 when told (by $\mathcal{F}_{\text{MCOM}}$) of a honest open(b), opens (equivocally) to \mathcal{A} .
- 4 when told (by \mathcal{A}) of a **correct** corrupted commitment, **extracts** the bit and commits accordingly to $\mathcal{F}_{\text{MCOM}}$.
- 5 when told (by \mathcal{A}) of a **correct** corrupted open(b^*):
 - If the extracted bit b is not the opened bit b^* , abort.
 - Else, instructs $\mathcal{F}_{\text{MCOM}}$ to open.

The Simulator

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

The simulator \mathcal{S} :

- 1 **selects** a CRS whose trapdoor and sk it knows.
- 2 when told (by $\mathcal{F}_{\text{MCOM}}$) of a honest commitment, commits **equivocally** to \mathcal{A} .
- 3 when told (by $\mathcal{F}_{\text{MCOM}}$) of a honest open(b), opens (equivocally) to \mathcal{A} .
- 4 when told (by \mathcal{A}) of a **correct** corrupted commitment, **extracts** the bit and commits accordingly to $\mathcal{F}_{\text{MCOM}}$.
- 5 when told (by \mathcal{A}) of a **correct** corrupted open(b^*):
 - If the extracted bit b is not the opened bit b^* , abort.
 - Else, instructs $\mathcal{F}_{\text{MCOM}}$ to open.
- 6 when asked (by \mathcal{A}) to corrupt a party, **learns** from $\mathcal{F}_{\text{MCOM}}$ the true committed value and hands \mathcal{A} the appropriate preimage and coins.

Helper Functions

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

Hybrids

$H_{\text{real/fake}}$ Same as $H_{\text{real/genuine}}$, but encrypt the **claw** instead of a dummy encryption. (The coins are still **erased**.)

$H_{\text{ideal/genuine}}$ Same as $H_{\text{ideal/fake}}$, but \mathcal{S} magically knows the committed-to bits of honests and generates appropriate **dummy** encryptions.

Helper Functions

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

Hybrids

$H_{\text{real/fake}}$ Same as $H_{\text{real/genuine}}$, but encrypt the **claw** instead of a dummy encryption. (The coins are still **erased**.)

$H_{\text{ideal/genuine}}$ Same as $H_{\text{ideal/fake}}$, but \mathcal{S} magically knows the committed-to bits of honests and generates appropriate **dummy** encryptions.

Events

\mathcal{B} The adversary \mathcal{A} generates a commitment, which \mathcal{S} extracts as b , but later is opened (by \mathcal{A}) **correctly** to $1 - b$.

\mathcal{C} The adversary \mathcal{A} commits with ciphertexts whose preimages (decryptions) are claws.

Note: $\mathcal{B} \subset \mathcal{C}$.

Lemmas

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- 1 $H_{r/g} \approx H_{r/f}$ (by LR-CPA-security of the encryption).
- 2 $\text{Prob} [\mathcal{B} \mid H_{r/f}] < \nu$ (because coins **are** erased).
- 3 $\text{Prob} [\mathcal{C} \mid H_{i/g}] < \nu$ (by claw-freeness).
- 4 $\left| \text{Prob} [\mathcal{C} \mid H_{i/f}] - \text{Prob} [\mathcal{C} \mid H_{i/g}] \right| < \nu$
(by LR-CPA/LR-CCA-security of the encryption).

Lemmas

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- 1 $H_{r/g} \approx H_{r/f}$ (by LR-CPA-security of the encryption).
- 2 $\text{Prob} [\mathcal{B} \mid H_{r/f}] < \nu$ (because coins **are** erased).
- 3 $\text{Prob} [\mathcal{C} \mid H_{i/g}] < \nu$ (by claw-freeness).
- 4 $\left| \text{Prob} [\mathcal{C} \mid H_{i/f}] - \text{Prob} [\mathcal{C} \mid H_{i/g}] \right| < \nu$
(by LR-CPA/LR-CCA-security of the encryption).
- 5 $\text{Prob} [\mathcal{B} \mid H_{i/f}] < \nu$
(from 3 and 4).

Lemmas

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

① $H_{r/g} \approx H_{r/f}$ (by LR-CPA-security of the encryption).

② $\text{Prob} [\mathcal{B} \mid H_{r/f}] < \nu$ (because coins **are** erased).

③ $\text{Prob} [\mathcal{C} \mid H_{i/g}] < \nu$ (by claw-freeness).

④ $\left| \text{Prob} [\mathcal{C} \mid H_{i/f}] - \text{Prob} [\mathcal{C} \mid H_{i/g}] \right| < \nu$
(by LR-CPA/LR-CCA-security of the encryption).

⑤ $\text{Prob} [\mathcal{B} \mid H_{i/f}] < \nu$
(from 3 and 4).

⑥ $\left| \text{Prob} [\mathcal{B} \mid H_{r/f}] - \text{Prob} [\mathcal{B} \mid H_{i/f}] \right| < \nu$
(from 2 and 5).

Lemmas

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

The Reusable
Commitment
Functionality

A Realizing
Protocol
Proof of Security

Summary

- 1 $H_{r/g} \approx H_{r/f}$ (by LR-CPA-security of the encryption).
- 2 $\text{Prob} [\mathcal{B} \mid H_{r/f}] < \nu$ (because coins **are** erased).
- 3 $\text{Prob} [\mathcal{C} \mid H_{i/g}] < \nu$ (by claw-freeness).
- 4 $\left| \text{Prob} [\mathcal{C} \mid H_{i/f}] - \text{Prob} [\mathcal{C} \mid H_{i/g}] \right| < \nu$
(by LR-CPA/LR-CCA-security of the encryption).
- 5 $\text{Prob} [\mathcal{B} \mid H_{i/f}] < \nu$
(from 3 and 4).
- 6 $\left| \text{Prob} [\mathcal{B} \mid H_{r/f}] - \text{Prob} [\mathcal{B} \mid H_{i/f}] \right| < \nu$
(from 2 and 5).
- 7 $H_{r/f} \approx H_{i/f}$
(from 6, since \mathcal{B} is the only distinction).

Summary

Universally
Composable
Commitments

The UC
Framework

The
Commitment
Functionality

A Better
Construction

Summary

- 1 The UC Framework
- 2 The Commitment Functionality
- 3 A Better Construction

A young man who is unable to commit a folly
is already an old man.

—Paul Gauguin

The End.