

An Overview on Universal Composability

M. Choudary Gorantla

Information Security Institute
Queensland University of Technology

June 5, 2008

Outline

- 1 Introduction
- 2 UC framework
- 3 Formal Definitions
- 4 UC formulation of key exchange

All the figures are intentionally copied from Canetti's papers
and presentations :)

Protocol Analysis

- 1 Provide a problem definition
 - what the adversary is allowed to do
 - what it means for the primitive to be secure
- 2 Prove the protocol secure under the definition
 - Show that if the primitive is not secure then some known hard problem can be solved (proof by reduction)
- 3 This “standalone” approach does not guarantee composability

A simple example

1 Sharing Keying Material

- Let π be a secure protocol that uses an n -bit secret key k
- Define π_1 that uses a $2n$ bit key $k = k_1 || k_2$, publicizes k_1 and runs π using k_2
- Define π_2 that uses a $2n$ bit key $k = k_1 || k_2$, publicizes k_2 and runs π using k_1
- When run alone π_1 and π_2 are as secure as π , but become completely insecure when composed

More examples on key exchange and bit commitment in Canetti's "Tutorial Paper"

Ideal/Real model paradigm

- 1 Ideal Model
 - (Dummy) parties send their inputs to a (incorruptible) trusted party, called an ideal functionality \mathcal{F}
 - \mathcal{F} computes a specified function and sends the output to parties
- 2 Real Model
 - Real parties execute the protocol with no trusted help
- 3 A protocol is secure if any attack in the real model can be simulated in the ideal model
- 4 Since there can be no such attack in the ideal model, the real protocol is secure

Universal Composition

- Let ρ be a protocol that realizes an ideal functionality \mathcal{F}
- Let π be an arbitrary protocol in which the parties make ideal calls to different instances of \mathcal{F}
- π is called a \mathcal{F} -hybrid protocol that uses both real model communication and instances of ideal process for \mathcal{F}
- A composed protocol π^ρ is constructed by executing π and replacing ideal calls to \mathcal{F} with invocations of ρ

Theorem

running the protocol π^ρ , with no access to \mathcal{F} , has essentially the same effect as running the \mathcal{F} -hybrid protocol π

Universal composition with joint state (JUC)

- 1 Ideal functionality of a cryptographic task is specified for single session execution
 - By UC theorem, multiple concurrent sessions of a protocol realize multiple concurrent instances of ideal functionality
 - security is guaranteed for multiple sessions of a protocol that have mutually disjoint states
- 2 JUC guarantees security for multi-session extensions
- 3 Analyse protocol for a single session execution and apply the JUC theorem to achieve security of multi-session extension

JUC

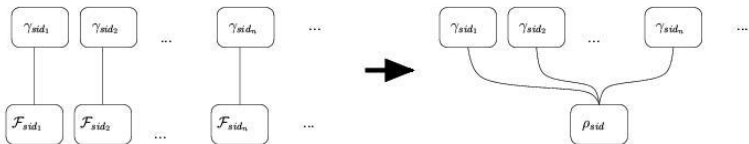


Figure: JUC (Canetti'01, Page 64)

- 1 γ is a protocol that UC-realizes \mathcal{G} making ideal calls to \mathcal{F}
- 2 ρ is a protocol that UC-realizes multiple instances (ssid) of \mathcal{F} within a single instance (sid)
- 3 the protocol on the right hand side UC-realizes \mathcal{G}

Definition of security

Protocol Emulation

A protocol π securely realizes an ideal functionality \mathcal{F} if for any real-life adversary \mathcal{A} , there exists an ideal-process adversary \mathcal{S} such that no environment \mathcal{Z} , on any input, can tell with non-negligible probability whether it is interacting with \mathcal{A} and parties running π in the real-life process, or it is interacting with \mathcal{S} and \mathcal{F} in the ideal process

What does this mean?

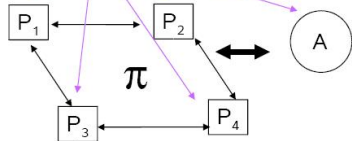
- 1 π should be as good as $\text{IDEAL}_{\mathcal{F}}$
- 2 \mathcal{Z} can be any arbitrary environment (e.g. higher level protocol calling π)

Protocol Emulation

UC security:



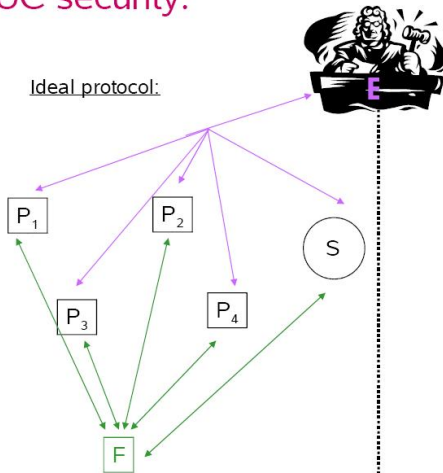
Protocol execution:



Protocol Emulation

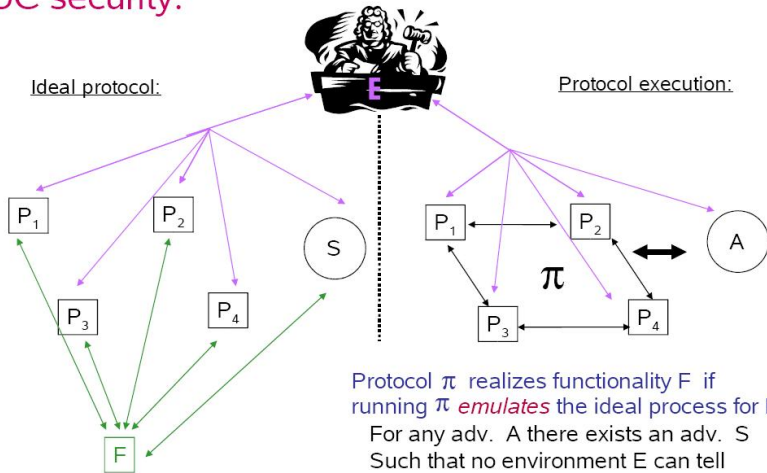
UC security:

Ideal protocol:



Protocol Emulation

UC security:



Protocol π realizes functionality F if running π *emulates* the ideal process for F :
For any adv. A there exists an adv. S
Such that no environment E can tell whether it's interacting with:

“Universal” composition

- subroutine calls
- Sequential, parallel, concurrent, executions
- Executions by same party, by unrelated parties
- Executions on same/related inputs, on unrelated inputs
- Unbounded number of executions
- Dynamic and adversarial code generation (“chosen protocol attacks”)

A functionality for key exchange (\mathcal{F}_{KE}): Attempt 1

Functionality \mathcal{F}_{KE}

Wait to receive:

- (sid, P_i, P_j) from party P_i
- (sid, P_j, P_i) from party P_j

Then:

- 1 Choose $\kappa \in_R \{0, 1\}^k$
- 2 Output $(\text{sid}, P_i, P_j, \kappa)$ to P_i and P_j
- 3 Send (sid, P_i, P_j) to the adv.
- 4 Halt.

Too Ideal

\mathcal{F}_{KE} : Attempt 2

Functionality \mathcal{F}_{KE}

Wait to receive:

- (sid, P_i, P_j) from party P_i
- (sid, P_j, P_i) from party P_j

Then:

- 1 If one of the parties is corrupted then obtain a value a from the adv. Else choose $\kappa \in_R \{0, 1\}^k$
- 2 Output $(\text{sid}, P_i, P_j, \kappa)$ to P_i and P_j
- 3 Send (sid, P_i, P_j) to the adv.
- 4 Halt.

Too Strong: requires mutual authentication

\mathcal{F}_{KE} : Attempt 3 [CK01]

Functionality \mathcal{F}_{KE}

\mathcal{F}_{KE} proceeds as follows, running on security parameter k , with parties P_1, \dots, P_n and an adversary S .

- 1 Upon receiving a value (Establish — session, $sid, P_i, P_j, role$) from some party P_i , record the tuple $(sid, P_i, P_j, role)$ and send this tuple to S . In addition, if there already is a recorded tuple $(sid, P_j, P_i, role')$ (either with $role' \neq role$ or $role' = role$) then proceed as follows:
 - 1 If both P_i and P_j are uncorrupted then choose $\kappa \in_R \{0, 1\}^k$, send (key, sid, κ) to P_i and P_j , send (key, sid, P_i, P_j) to S and halt.
 - 2 If either P_i or P_j is corrupted, then send a message (Choose — value, sid, P_i, P_j) to S ; receive a value κ from S , send (key, sid, κ) to P_i and P_j , and halt.
- 2 Upon corruption of either P_i or P_j , proceed as follows: If the session key is not yet sent (i.e. it was not yet written on the outgoing communication tape), then provide S with the session key. Otherwise provide no information to S .

Cannot be realizable if adaptive corruptions are allowed
[HMS03]

\mathcal{F}_{KE} : Attempt 4 [HMS03]

Functionality \mathcal{F}_{KE}

\mathcal{F}_{KE} proceeds as follows, running on security parameter k , with parties P_1, \dots, P_n and an adversary S .

- 1 Wait to receive values $(ready, sid)$ from the parties P_i and P_j and from S . When receiving $(ready, sid)$ from either P_i or P_j , forward this message (including the sender identity) to S .
- 2 After having received values $(ready, sid)$ from P_i, P_j , and S (in any order), proceed as follows:
 - 1 If both P_i and P_j are uncorrupted, choose $\kappa \in_R \{0, 1\}^k$.
 - 2 If at least one of the parties P_i and P_j is corrupted, send a message (choose — key, sid) to S . Upon receiving an answer (key, sid, κ) from S , extract the value of κ from it.
- 3 Once κ is set, send (key, sid, κ) to P_i and P_j , and send (key, sid) to S . Then halt.

Caveats

- 1 Composability does not guarantee security
 - the protocol is only as good as the ideal functionality it realizes
- 2 There can be ideal functionalities which may not be realizable at all
 - need to put some relaxations
 - define an ideal functionality that would be realizable by simple protocols (with milder assumptions) and at the same time would guarantee reasonable security and composability
- 3 Each cryptographic task should have an ideal functionality (?)

Thank You!