A Novel Framework for Protocol Analysis[*]

Kristian Gjøsteen[†] George Petrides[†] Asgeir Steine[†]
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
{kristiag,petrides,asgeirs}@math.ntnu.no

Abstract

We describe a novel reformulation of Canetti's Universal Composability (UC) framework for the analysis of cryptographic protocols. Our framework is different mainly in that it is (a) based on systems of interactive Turing machines with a fixed communication graph and (b) augmented with a global message queue that allows the sending of multiple messages per activation. The first feature significantly simplifies the proofs of some framework results, such as the UC theorem, while the second can lead to more natural descriptions of protocols and ideal functionalities. We illustrate how the theory can be used with several examples.

Keywords: Protocol Security, Universal Composability

1 Introduction

Protocol analysis is arguing whether a given protocol has desirable functional and security properties or not. Canetti [2] introduced the Universal Composability framework, which has since become the most common model for analysis of cryptographic protocols (although there are other more or less equivalent models [7]).

In Canetti's framework, analysis begins by defining a so-called ideal functionality that encapsulates some desired functionality and security properties in the form of a trusted third party. The idea is to compare a cryptographic protocol with this ideal functionality. If the protocol is in a specific sense indistinguishable from the functionality, we say that the protocol realises the functionality. If the functionality has the desired functional and security properties, then so will the protocol.

However, Canetti's definitions have more powerful properties. The most interesting property is that of composability, where any subprotocol can be replaced by an ideal functionality that is realised by the subprotocol. This is interesting from a practical point of view, since ideal functionalities are typically much easier to work with than cryptographic protocols. Replacing subprotocols by ideal functionalities can therefore simplify analysis.

We note that ideal functionalities are also used to provide ideal models for the facilities underlying protocols, such as communication networks, common reference strings, random oracles, out-of-band key agreement, etc.

One interesting feature of Canetti's approach is a strong preference for studying the single-session case, since this significantly simplifies analysis. The multi-session analysis follows from the single-session case by composing multiple instances. For many protocols, this approach is quite simply not feasible (see [4] for one example). In other cases, multi-session analysis may result in tighter concrete results (the key agreement functionality in Sect. 5.3 is one example).

Some drawbacks of Canetti's framework are more apparent in the setting of anonymous communications. One such drawback is that all protocol machines must agree on a unique session identifier before

[*]An extended abstract of this paper has been published in ProvSec 2011, LNCS 6890, Springer, 2011.

[†]Funded by the Norwegian Research Council's VERDIKT programme, project 183195.

the execution. This is awkward because the anonymity requirements explicitly forbid most obvious approaches for agreeing on session identifiers.

Another drawback is the way the activation of interactive Turing machines (ITMs) is handled. When an ITM wants to send a message to another ITM, it has to write the message to the appropriate communication tape and stop its execution. The message recipient is activated, but the sender cannot control when itself will be reactivated.

Our contribution. Based on our ongoing work with practical protocols (among others with anonymous communications and electronic elections), we have developed a reformulation of Canetti's framework in terms of systems of ITMs where communication is regulated by a fixed communication graph.

In this paper we formally describe this reformulation, for which we also define the notions of emulation and realisation, and prove the usual composition theorem from [2]. Additionally, we provide several examples that demonstrate various framework features.

There are two main differences between our formulation and that of Canetti:

- Canetti uses a very dynamic setting, where instances of ITMs come into existence as needed. Instead, we consider a fixed system of ITMs with communication regulated by a fixed communication graph.
- We augment Canetti's system of activation with a message queue, where ITMs can submit multiple messages into the queue for later delivery. Moreover, ITMs can send messages to themselves and, therefore, also self-reactivate once all previously enqueued messages have been processed.

Since we use a fixed system of ITMs and a fixed communication graph, we shall usually let our protocol machines handle multiple sessions. As we have already argued, this is already necessary for some protocols, and may have other advantages as well.

The main advantage of adding a message queue is to get simpler and more natural protocol descriptions. In the field of anonymous communications, this also gives us quite natural solutions to certain problems that seem difficult to solve in Canetti's framework.

A final advantage of our formulation is that it significantly simplifies proofs of framework results. For instance, the proof of our composition theorem is essentially contained in the drawing given in Fig. 2.

We follow the concrete security approach of [1], which we believe is essential for practical applications. Canetti uses local resource bounds on the parts of the system that imply global resource bounds for the entire system. Instead, we consider systems that "usually terminate", without exceeding global resource bounds. In our experience, this simplifies many arguments significantly.

Overview. Section 2 provides the basic definitions for systems and partial systems of interactive Turing machines. Once we have a notion of executions of systems, in Sect. 3 we introduce the notion of indistinguishable systems and partial systems, and prove some basic results about indistinguishability. Section 4 defines the notion of emulation and proves our composition theorem. Finally, our framework is illustrated via examples in Sect. 5, before our conclusions appear in Sect. 6.

1.1 Definitions

An *unordered pair* is a set with one or two elements.

Our graphs will be undirected multigraphs with loops. A *graph* is a tuple (V, E, λ), where V is a set of vertices, E is a set of edges and λ is a function that assigns an unordered pair of vertices to every edge.

An *interactive Turing machine* (ITM) is a Turing machine with one read-and-write working tape and five other tapes: random, input, incoming communication, output, and outgoing communication. The first three are read-only, the latter two are write-only. The ITM may enter a special *wait state*, in which processing temporarily stops, but may be resumed later from the same state. We refer the reader to [5] for more information on ITMs.

2 Systems of Interactive Turing Machines

In this section we introduce the notion of a system, essentially an assignment of ITMs to vertices in an undirected multigraph where the edges in the graph decide which ITMs can communicate with each other. An execution of such a system begins by activating the ITM instance attached to the initial vertex. Each active instance may add any number of messages to the message queue. When done, the active instance may choose to deliver a message immediately to another instance, thereby activating it, or deliver the first message in the queue, activating its recipient.

We also define partial systems, essentially systems with "loose edges" attached to vertices in the system. Partial systems can be composed by composing the communication graphs and replacing identical "loose edges" with real edges.

2.1 Definitions

A *system* $\Sigma = (V, E, \lambda, v_0, M)$ consists of the following:

- A *communication graph* (V, E, λ).
- A distinguished *initial vertex* $v_0 \in V$.
- A function M that assigns an ITM to each $v \in V$. We denote this ITM by M_v.

The communication graph can have parallel edges, and we shall assume that every vertex has a loop.

A *partial system* $\Pi = (W, F, \lambda', N, X, \kappa)$ consists of a graph (W, F, λ'), a function N that assigns an ITM to each $v \in W$, a set X of *external* edges (disjoint from the *internal* edges F) and a function $\kappa : X \to W$ that assigns a vertex to every external edge.

A partial system is *initial* if a vertex is designated as initial. Two partial systems are *interchangeable* if they have identical external edges, and they are either both initial or both not initial.

Two partial systems $\Pi_1 = (W_1, F_1, \lambda'_1, N_1, X_1, \kappa_1)$ and $\Pi_2 = (W_2, F_2, \lambda'_2, N_2, X_2, \kappa_2)$ are *composable* if they are not both initial and have disjoint vertex and edge sets. Their *composition*, denoted by $\Pi_1 \odot \Pi_2$, is the partial system $\Pi_3 = (W_3, F_3, \lambda'_3, N_3, X_3, \kappa_3)$, where:

- $W_3 = W_1 \cup W_2$,
- $F_3 = F_1 \cup F_2 \cup (X_1 \cap X_2)$,
- $\lambda'_3(e) = \begin{cases} \lambda'_1(e) & \text{if } e \in F_1, \\ \lambda'_2(e) & \text{if } e \in F_2, \\ \{\kappa_1(e), \kappa_2(e)\} & \text{if } e \in X_1 \cap X_2, \end{cases}$
- $N_{3,v} = \begin{cases} N_{1,v} & \text{if } v \in W_1, \\ N_{2,v} & \text{if } v \in W_2, \end{cases}$
- $X_3 = (X_1 \cup X_2) \setminus (X_1 \cap X_2)$, and
- $\kappa_3(e) = \begin{cases} \kappa_1(e) & \text{if } e \in X_1, \\ \kappa_2(e) & \text{if } e \in X_2. \end{cases}$

If either Π_1 or Π_2 is initial, then so is $\Pi_1 \odot \Pi_2$. If $\Pi_1 \odot \Pi_2$ is initial and has no external edges then it is a system and we say that Π_1 *completes* Π_2 (and vice versa).

Remark 1. Given two interchangeable partial systems, a third partial system may be composable with only one of them due to conflicts with vertices or internal edges in the other. For the same reason, several other natural and desirable operations, such as the composition of a partial system with itself, are forbidden by the above constructions.

The simple solution to this technical problem is to rename edges. We shall assume that such renaming can be done without cost. Also, trivial renaming will typically go unmentioned.

2.2 Execution Model

An *execution* of a system $\Sigma = (V, E, \lambda, v_0, M)$ works with the following state:

- For each vertex $v \in V$, one instance of the ITM M_v.
- A message queue Q containing tuples $V \times E \times \{0,1\}^*$.

When the execution starts, any input to the system is written to the input tape of the M_{v_0} instance, which is then activated. Whenever an instance of an ITM M_v enters its wait state, the execution proceeds as follows:

- *Queuing a message.* If M_v has written a tuple $(\mathtt{queue}, e, m)$ to its communication tape and $\lambda(e) = \{v, v'\}$ for some $v' \in V$ then add (v, e, m) to Q and reactivate M_v.
- *Handing over.* If M_v has written $(\mathtt{hand{-}over}, e, m)$ to its communication tape and $\lambda(e) = \{v, v'\}$ then write (e, m) to the incoming communication tape of $M_{v'}$ and activate $M_{v'}$.
- *Delivering a message.* If M_v did not write to its communication tape and (v', e, m) is the first entry in Q with $\lambda(e) = \{v', v''\}$ then remove the entry from the queue, write (e, m) to the incoming communication tape of $M_{v''}$ and activate $M_{v''}$.
- *Default activation.* If M_v did not write to its communication tape, $v \neq v_0$ and Q is empty then activate M_{v_0}.
- *Termination.* If $v = v_0$ and M_v has halted after writing a (possibly empty) string x to its output tape then copy x to the system's output tape and stop.

Remark 2. Note that queuing messages via loop edges allows for some degree of controlled self-reactivation of ITMs.

2.2.1 Resource Bounds

There is no requirement that an execution must terminate. However, we shall mainly be interested in systems that with high probability terminate, in which case it is interesting to consider the time required before termination. We define the total execution time of a system to be the sum of the execution time of all the ITMs plus the cost of managing activation and the message queue.

We say that a system is (t, δ)-*bounded* if it terminates after time at most t, except with probability at most δ.

Apart from execution time, there are other interesting resource measures in protocol analysis, such as the number of sessions run or the number of corrupted parties. In particular, the size of the ITM description is relevant in preventing trading execution time for description size.

In general, for some tuple of resource bounds $R = (r_1, \dots, r_n)$, we shall say that a system is (R, δ)-*bounded* or $(r_1, \dots, r_n, \delta)$-*bounded* if the probability that the system terminates without exceeding any of the resource bounds in R is at least $1 - \delta$.

2.3 Protocols

The notion of a system of ITMs described above is very general. Our next aim is to show how protocols fit into this general system. For this, we shall define two notions, those of *protocol machine* and *ideal functionality*.

Protocol machines and *ideal functionalities* are ITMs that expect to be attached to vertices in a communication graph with a loop and specific number of incident non-loop edges. An ideal functionality will distinguish a special *attacker edge* and consider the remaining edges as *input/output* (*i/o*) *edges*. A protocol machine will distinguish up to three classes of edges: one or more *i/o edges*, zero or more *subprotocol edges* and zero or one *corruption edge*.

Remark 3. The interpretation of this is as follows. A protocol machine is attached to a vertex in a communication graph and receives instructions over its i/o edges directing its operation. If the protocol produces output, this is sent out via the i/o edges.

An attacker may be able to corrupt the protocol machine in various ways. In this case, the attacker will send special messages to the protocol machine over its corruption edge, and the protocol machine will respond according to its programming.

The protocol machine may delegate work to subprotocols. If so, the protocol machine controls the subprotocol by sending messages over its subprotocol edges and possibly receiving subprotocol output via the same subprotocol edges.

An ideal functionality either models an ideal subprotocol or some idealisation of some feature a protocol relies on, such as a physical communications network. The i/o edges play the same role as for protocol machines.

Protocol machines and ideal functionalities are organised in a layered hierarchical fashion with protocol machines above their subprotocol machines and ideal functionalities at the bottom. This is made formal by the notion of *protocol system*, which is a non-initial partial system satisfying:

1. All the ITMs are either protocol machines or ideal functionalities.
2. Every corruption and attacker edge is external.
3. Any non-external edge is considered an i/o edge by one of its incident ITMs and a subprotocol edge by the other.
4. If every edge in the graph is considered to be directed from the ITM that considers it as a subprotocol edge to the ITM that considers it as an i/o edge, then the communication graph has no cycles except for trivial cycles made up of loop edges.

If no ITM considers an external edge to be a subprotocol edge, we say that the protocol system is *closed*. Two protocol systems are *composable* if they are composable as partial systems and the composition is again a protocol system.

Remark 4. It is possible to have a closed protocol system consisting of a single ideal functionality. As we will see later (Sect. 5), this is an important class of protocol systems.

Remark 5. Traditionally, the equivalent of a closed protocol system is assumed to interact with a partial system composed of an environment and an attacker. In our formulation, a closed protocol will be composed with an initial partial system (the environment) so that it forms a system. Informally, we may consider part of the environment to be the attacker.

Remark 6. It will be useful to assume that every protocol machine is given an "identity", and every ideal functionality associates an "identity" with each i/o edge. These identities must be consistent, in the sense that subprotocol and parent protocol machines have the same identities, and the identity the ideal functionality associates to an i/o edge corresponds to the identity of any protocol machine attached via that edge.

3 Indistinguishability

The notion of indistinguishability is central in cryptography, and we shall define two notions of indistinguishability for systems and partial systems.

A distinguisher is an algorithm that provides input to an unknown system and then tries to say something about the system based on its output. Formally, a *distinguisher* $D = (D_1, D_2)$ is a pair of algorithms such that D_1 outputs a string and a state, while D_2 takes a string and a state as input and outputs 0 or 1. An execution of D with a system Σ proceeds as follows:

1. D_1 is run and outputs x and a state.
2. The system Σ gets x as input to an execution and outputs y.
3. D_2 is given y and the state as input and outputs 0 or 1.

Remark 7. Since systems are not guaranteed to terminate, the same holds for distinguisher executions. However, if the system execution terminates, the distinguisher execution will also terminate.

We say that a distinguisher interacting with a system Σ is (R, δ)-bounded with respect Σ if the above execution does not exceed the resource bounds in R except with probability δ. Note that we consider the time used to be not just the time used for the system execution, but also the time used by the distinguisher algorithms.

3.1 Definition of Indistinguishability of Systems

Let Σ_1 and Σ_2 be two systems, D a distinguisher and R_1 and R_2 resource bounds. Let E_i be the event that the execution of D with Σ_i outputs 1, and let G_i be the event that the execution of D with Σ_i terminates without exceeding the resource bound R_i. Note that D interacting with Σ_i is (R_i, δ_i)-bounded if $\Pr[\neg G_i] \leq \delta_i$.

The *distinguishing advantage* of D with respect to the resource bounds R_1 and R_2 is defined to be

$$\mathrm{Adv}(D, \Sigma_1, \Sigma_2; R_1, R_2) = |\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2]| \ .$$

A system Σ_1 is $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$*-indistinguishable* from system Σ_2 if for any D the following holds:

1. If (D, Σ_1) is (R_1, δ_1)-bounded then (D, Σ_2) is (R_2, δ_2)-bounded ($\Pr[\neg G_1] \leq \delta_1 \Rightarrow \Pr[\neg G_2] \leq \delta_2$).
2. $\mathrm{Adv}(D, \Sigma_1, \Sigma_2; R_1, R_2) \leq \varepsilon$.

Remark 8. Note that this notion of indistinguishability need not be symmetric.

We have now defined a notion of bounded indistinguishability. Next, we define a similar notion of unbounded indistinguishability. The *unbounded distinguishing advantage* of D is defined to be

$$\mathrm{Adv}(D, \Sigma_1, \Sigma_2) = |\Pr[E_1] - \Pr[E_2]| \ .$$

We say that Σ_1 and Σ_2 are ε*-indistinguishable* if any distinguisher D has unbounded distinguishing advantage at most ε.

Remark 9. If two systems are ε-indistinguishable, it is easy to show that for any input distribution, the probability that the executions terminate may differ by at most ε.

These notions of indistinguishability for systems carry over in a natural way to partial systems. Let Π_1 and Π_2 be interchangeable partial systems. We say that Π_1 is $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$-*indistinguishable* (respectively ε-*indistinguishable*) from Π_2 if for any partial system Π_3 that completes Π_1, we have that $\Pi_1 \odot \Pi_3$ is $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$-indistinguishable (respectively ε-indistinguishable) from $\Pi_2 \odot \Pi_3$ as systems.

3.2 Basic Results

3.2.1 Transitivity

We have a limited form of transitivity for indistinguishability. This follows from the fact that if two systems are distinguishable then one or both must be distinguishable from any third system.

Theorem 10. *Suppose* Π_1, Π_2 *and* Π_3 *are interchangeable partial systems such that* Π_1 *is* $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$*-indistinguishable from* Π_2 *and* Π_2 *is* $(R_2, \delta_2, R_3, \delta_3, \varepsilon')$*-indistinguishable from* Π_3. *Then,* Π_1 *is* $(R_1, \delta_1, R_3, \delta_3, \varepsilon + \varepsilon')$*-indistinguishable from* Π_3.

Proof. Let Π_4 be any partial system that completes Π_1 and D be a distinguisher. We define E_i to be the event that the execution of D with $\Pi_i \odot \Pi_4$ outputs 1 and G_i the event that this execution terminates before exceeding the bound R_i, $1 \leq i \leq 3$.

The first condition of $(R_1, \delta_1, R_3, \delta_3, \varepsilon + \varepsilon')$-indistinguishability is given by the first condition of the indistinguishability assumptions:

$$\Pr[\neg G_1] \leq \delta_1 \Rightarrow \Pr[\neg G_2] \leq \delta_2 \Rightarrow \Pr[\neg G_3] \leq \delta_3.$$

The second condition can be verified by the following computation using the second condition of the indistinguishability assumptions:

$$\begin{aligned} |\Pr[E_1 \wedge G_1] - \Pr[E_3 \wedge G_3]| &\leq |\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2]| + |\Pr[E_2 \wedge G_2] - \Pr[E_3 \wedge G_3]| \\ &\leq \varepsilon + \varepsilon' \ , \end{aligned}$$

which concludes the proof. □

3.2.2 Composition

It is clear that if we have two indistinguishable partial systems and compose them with the same partial system, the resulting (partial) systems are indistinguishable.

Theorem 11. *Suppose* Π_1 *and* Π_2 *are interchangeable partial systems such that* Π_1 *is* $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$*-indistinguishable from* Π_2. *Then, for any partial system* Π_3 *composable with* Π_1 *we have* $\Pi_1 \odot \Pi_3$ *is* $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$*-indistinguishable from* $\Pi_2 \odot \Pi_3$.

Proof. This follows immediately from the definition of indistinguishability since either Π_3 completes Π_i or for any partial system Π_4 that completes $\Pi_i \odot \Pi_3$, the partial system $\Pi_3 \odot \Pi_4$ completes Π_i. □

3.2.3 Adding Resource Bounds

Suppose we have two ε-indistinguishable partial systems. What happens if we impose resource bounds on them ? It is clear that it depends on the resource bounds since a bounded distinguisher is considered successful if, with high probability, it keeps the bound in the execution with one of the partial systems but not with the other. Theorem 12 gives us a way to convert unbounded indistinguishability to bounded indistinguishability.

Theorem 12. *Let Π_1 and Π_2 be 0-indistinguishable partial systems, and let (R_1,δ_1) and (R_2,δ_2) be resource bounds. Suppose that for any distinguisher D and partial system Π_3 that completes Π_1, if $(D,\Pi_1 \odot \Pi_3)$ is (R_1,δ_1)-bounded then $(D,\Pi_2 \odot \Pi_3)$ is (R_2,δ_2)-bounded. Then, Π_1 is $(R_1,\delta_1,R_2,\delta_2,\max\{\delta_1,\delta_2\})$-indistinguishable from Π_2.*

Proof. The first condition of indistinguishability is given by the assumption, hence we only need to show that for any distinguisher D and partial system Π_3 that completes Π_1 we have $\mathrm{Adv}(D,\Pi_1 \odot \Pi_3,\Pi_2 \odot \Pi_3;R_1,R_2) \leq \max\{\delta_1,\delta_2\}$.

For a distinguisher D and partial system Π_3 as above, let E_i be the event that an execution of D with $\Pi_i \odot \Pi_3$ outputs 1 and G_i the event that this execution terminates without exceeding the bound R_i. Then, the distinguishing advantage is $|\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2]|$. First note that, trivially,

$$\Pr[E_i \wedge G_i] \leq \Pr[E_i] \ . \tag{1}$$

Also, since

$$\Pr[E_i] = \Pr[E_i \wedge G_i] + \Pr[E_i \wedge \neg G_i] \leq \Pr[E_i \wedge G_i] + \Pr[\neg G_i] \ ,$$

it follows that

$$\Pr[E_i \wedge G_i] \geq \Pr[E_i] - \Pr[\neg G_i] \ . \tag{2}$$

We now need to show that the advantage is bounded by $\max\{\delta_1,\delta_2\}$. There are two cases to consider:

Case 1: If $\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2] > 0$ then by inequalities (1) and (2), and the definition of 0-indistinguishability we have

$$\begin{aligned}\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2] &\leq \Pr[E_1] - (\Pr[E_2] - \Pr[\neg G_2]) \\ &= (\Pr[E_1] - \Pr[E_2]) + \Pr[\neg G_2] \\ &\leq \delta_2.\end{aligned}$$

Case 2: Similarly, if $\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2] < 0$ we have

$$\begin{aligned}\Pr[E_2 \wedge G_2] - \Pr[E_1 \wedge G_1] &\leq \Pr[E_2] - (\Pr[E_1] - \Pr[\neg G_1]) \\ &= (\Pr[E_2] - \Pr[E_1]) + \Pr[\neg G_1] \\ &\leq \delta_1.\end{aligned}$$

In either case the distinguishing advantage is bounded by $\max\{\delta_1,\delta_2\}$ as required. □

The same arguments can be extended to ε-indistinguishable partial systems which with added resource bound gives $(R_1,\delta_1,R_2,\delta_2,\varepsilon + \max\{\delta_1,\delta_2\})$-indistinguishability.

3.2.4 Subsystem Simulator

Any partial system can be simulated by a single "simulator" vertex. The "simulator" simply runs the partial system as described in the execution model. However, the simulating machine will most usually need some extra time for accounting. It is possible to prove the following theorem.

Theorem 13. *For any partial system Π there is an interchangeable single-vertex partial system Π_S that is 0-indistinguishable from Π.*

```
Algorithm $A$ with input $z$:
  1. Let $D$ interact with the system $\Pi_1 \odot \Pi_3$ with
     the exception that the message $\Pi_1$ sends over
     its external edge is substituted by
     $(\mathsf{sampled}, z)$.
  2. If $D$ stops without reaching the resource
     bound $R$, output 0.
  3. If $D$ reaches the resource bound $R$, stop the
     interaction and output 1.
```

```
Algorithm $B$ with input $z$:
  1. Let $D$ interact with the system $\Pi_1 \odot \Pi_3$ with
     the exception that the message $\Pi_1$ sends over
     its external edge is substituted by
     $(\mathsf{sampled}, z)$.
  2. If $D$ outputs $b$ without reaching the resource
     bound $R$, output $b$.
  3. If $D$ reaches the resource bound $R$, stop the
     interaction and output 0.
```

Figure 1: Algorithms for the proofs of Lemma 15 and Lemma 16.

3.3 Techniques

How do we prove that two partial systems are indistinguishable? One approach is to build on the existing work for indistinguishable probability spaces. Essentially, two probability spaces are indistinguishable if it is hard to say from which space an element was sampled. Such indistinguishable spaces can be used to build indistinguishable partial systems.

Another approach for constructing indistinguishable partial systems is to identify an "exceptional" event, prove that as long as that event does not occur the partial systems are indistinguishable, and finally bound the probability of that event occurring. This bound will also bound the distinguishing advantage.

3.3.1 Indistinguishable Probability Spaces

Let Z_1 and Z_2 be two probability spaces and assume that sampling from Z_1 and Z_2 requires identical time. A t-distinguisher for Z_1 and Z_2 is an algorithm that on input z sampled either from Z_1 or Z_2 requires time at most t (including sampling time) before outputting either 0 or 1.

We say that Z_1 and Z_2 are (t, ε)-*indistinguishable* if for any t-distinguisher it holds that

$$|\Pr[E_1] - \Pr[E_2]| \leq \varepsilon \ ,$$

where E_i is the event that the t-distinguisher outputs 1 on input sampled from Z_i.

Define two partial systems Π_1 and Π_2 having one vertex and one external edge as follows: The first time it receives (sample) via the external edge, the machine attached to the corresponding vertex samples z from Z_i and sends $(\mathsf{sampled}, z)$ over its external edge, and nothing else happens.

Theorem 14. *Let Z_1 and Z_2 be (t, ε)-indistinguishable probability spaces, Π_1 and Π_2 be as above, and (R, δ) be any resource bound that has t as its time bound. Then Π_1 is $(R, \delta, R, \delta + \varepsilon, \varepsilon)$-indistinguishable from Π_2.*

The proof proceeds in two steps. First, Lemma 15 proves the resource bound. Then Lemma 16 bounds the distinguishing advantage.

Lemma 15. *Let Z_1, Z_2, Π_1 and Π_2 be as above, Π_3 be any partial system that completes Π_1 and Π_2, and D be a distinguisher such that D interacting with $\Pi_1 \odot \Pi_3$ is (R, δ)-bounded. Then D interacting with $\Pi_2 \odot \Pi_3$ is $(R, \delta + \varepsilon)$-bounded.*

Proof. Let δ_i be the probability that D interacting with $\Pi_i \odot \Pi_3$ exceeds the resource bound R. Note that $\delta_1 \leq \delta$.

The algorithm A given in Fig. 1 is a t-distinguisher for Z_1 and Z_2. We see that if its input has been sampled from Z_i, it simulates the system $\Pi_i \odot \Pi_3$ perfectly until cut-off, and the probability that it outputs 1 is δ_i.

Therefore, A is a $(t, |\delta_1 - \delta_2|)$-distinguisher for Z_1 and Z_2. This means that $|\delta_1 - \delta_2| \leq \varepsilon$, by assumption, which implies that $\delta_2 \leq \delta_1 + \varepsilon \leq \delta + \varepsilon$.

Hence, the probability that D interacting with $\Pi_2 \odot \Pi_3$ exceeds the resource bound R is at most $\delta + \varepsilon$. □

Lemma 16. *Let Π_3 be a partial system that completes Π_1 and D a distinguisher such that D interacting with $\Pi_1 \odot \Pi_3$ is (R, δ)-bounded. Then the distinguishing advantage of D is at most ε.*

Proof. The algorithm B given in Fig. 1 is a t-distinguisher for Z_1 and Z_2. For a distinguisher D we define the events E_i that D interacting with $\Pi_i \odot \Pi_3$ outputs 1 and G_i that the interaction is time bounded by R. We see that if B's input has been sampled from Z_i, it simulates the system $\Pi_i \odot \Pi_3$ perfectly until cut-off, and the probability that it outputs 1 equals $\Pr[E_i \wedge G_i]$.

The distinguishing advantage of B is therefore equal to $|\Pr[E_1 \wedge G_1] - \Pr[E_2 \wedge G_2]|$, which, by assumption, must be at most ε. □

This completes the proof of Theorem 14.

3.3.2 Exceptional Events

The technique of *exceptional events* [8] is central in modern cryptography. The idea is to isolate the conditions under which two systems can deviate, and then somehow bound the probability of this exceptional event occurring. This bound then becomes a bound on how often the systems can deviate, and therefore be distinguished.

Theorem 17. *Let Π_1 and Π_2 be interchangeable partial systems, and let F_1, F_2 be ("exceptional") events for the respective partial systems. For some resource bound R, any partial system Π_3 that completes Π_1 and any distinguisher D such that $(D, \Pi_1 \odot \Pi_3)$ is (R, δ)-bounded, we have that Π_1 is $(R, \delta, R, \delta + \varepsilon, \varepsilon)$-indistinguishable from Π_2 if the following hold:*

1. *The probability of F_1 and F_2 occurring is ε.*
2. *If F_1 and F_2 do not occur then the probabilities that*
   - *(i) the resource bound R is exceeded, and*
   - *(ii) the distinguisher outputs 1 without exceeding the resource bound R,*

   are each identical for the two systems.

Proof. For a distinguisher D and a partial system Π_3 that completes Π_1 and Π_2 we define the events E_i that D outputs 1 when interacting with $\Pi_i \odot \Pi_3$ and G_i that the interaction does not exceed the resource bound R. The conditions for the theorem are then:

1. $Pr[F_1] = Pr[F_2] = \varepsilon$,
2. $Pr[\neg G_1 | \neg F_1] = Pr[\neg G_2 | \neg F_2]$,
3. $Pr[E_1 \wedge G_1 | \neg F_1] = Pr[E_2 \wedge G_2 | \neg F_2]$.

We first assume that $Pr[\neg G_1] \leq \delta$ and prove that it implies $Pr[\neg G_2] \leq \varepsilon + \delta$. Using Conditions 1 and 2 above, we have that

$$\begin{aligned}
Pr[\neg G_2] &= Pr[\neg G_2 \wedge F_2] + Pr[\neg G_2 \wedge \neg F_2] \\
&\leq Pr[F_2] + Pr[\neg G_2 | \neg F_2] Pr[\neg F_2] \\
&\leq Pr[F_2] + Pr[\neg G_1 \wedge \neg F_1] \\
&\leq Pr[F_2] + Pr[\neg G_1] \\
&\leq \varepsilon + \delta \ .
\end{aligned}$$

Next we need to show that $|Pr[E_1 \wedge G_1] - Pr[E_2 \wedge G_2]| \leq \varepsilon$. Firstly, notice that

$$Pr[E_i \wedge G_i] = Pr[E_i \wedge G_i \wedge F_i] + Pr[E_i \wedge G_i \wedge \neg F_i] \ .$$

Secondly, by Conditions 1 and 3 above, we have that

$$\begin{aligned}
Pr[E_1 \wedge G_1 \wedge \neg F_1] - Pr[E_2 \wedge G_2 \wedge \neg F_2] &= Pr[E_1 \wedge G_1 | \neg F_1] Pr[\neg F_1] - Pr[E_2 \wedge G_2 | \neg F_2] Pr[\neg F_2] \\
&= (1 - \varepsilon)(Pr[E_1 \wedge G_1 | \neg F_1] - Pr[E_2 \wedge G_2 | \neg F_2]) \\
&= 0 \ .
\end{aligned}$$

It follows that

$$\begin{aligned}
|Pr[E_1 \wedge G_1] - Pr[E_2 \wedge G_2]| &= |Pr[E_1 \wedge G_1 \wedge F_1] - Pr[E_2 \wedge G_2 \wedge F_2] + 0| \\
&\leq \max\{Pr[F_1], Pr[F_2]\} \\
&= \varepsilon \ ,
\end{aligned}$$

which concludes the proof. □

4 Composition Theorem

One very useful notion is that of composability. A protocol is said to be secure if it is in some sense indistinguishable from an appropriate ideal functionality. When this secure protocol is used as a sub-protocol, the following theorem proves that analysis of the parent protocol can be done using the ideal functionality abstraction, possibly a major simplification.

Let Π_1 and Π_2 be protocol systems with the same external i/o edges. We say that Π_1 $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$-*emulates* Π_2 if there exists a partial system $\mathscr{S}$ such that Π_1 and $\Pi_2 \odot \mathscr{S}$ are interchangeable, and Π_1 is $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$-indistinguishable from $\Pi_2 \odot \mathscr{S}$. In such case, we call $\mathscr{S}$ a *simulator*.

If the partial system Π_2 consists of a single ideal functionality $\mathscr{F}$, we say that Π_1 $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$-*realises* Π_2 and, informally, that it is $\mathscr{F}$*-secure*.

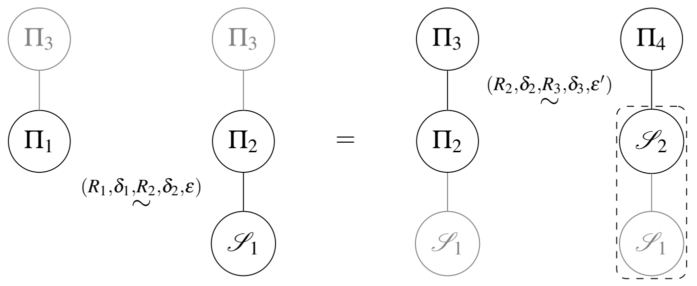
Theorem 18 (Composition). *Let* Π_1, Π_2 *and* Π_4 *be closed protocol systems and let* Π_3 *be a protocol system composable with* Π_1 *and* Π_2. *If* Π_1 $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$*-emulates* Π_2 *and* $\Pi_2 \odot \Pi_3$ $(R_2, \delta_2, R_3, \delta_3, \varepsilon')$*-emulates* Π_4 *then* $\Pi_1 \odot \Pi_3$ $(R_1, \delta_1, R_3, \delta_3, \varepsilon + \varepsilon')$*-emulates* Π_4.

Proof. Let $\mathscr{S}_1$ and $\mathscr{S}_2$ be simulators such that the partial system Π_1 is $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$-indistinguishable from $\Pi_2 \odot \mathscr{S}_1$ and $\Pi_2 \odot \Pi_3$ is $(R_2, \delta_2, R_3, \delta_3, \varepsilon')$-indistinguishable from $\Pi_4 \odot \mathscr{S}_2$.

Firstly, by Theorem 11, the system $\Pi_1 \odot \Pi_3$ is $(R_1, \delta_1, R_2, \delta_2, \varepsilon)$-indistinguishable from $\mathscr{S}_1 \odot \Pi_2 \odot \Pi_3$, and $\mathscr{S}_1 \odot \Pi_2 \odot \Pi_3$ is $(R_2, \delta_2, R_3, \delta_3, \varepsilon')$-indistinguishable from $\mathscr{S}_1 \odot \mathscr{S}_2 \odot \Pi_4$.

Secondly, by Theorem 10, $\Pi_1 \odot \Pi_3$ is $(R_1, \delta_1, R_3, \delta_3, \varepsilon + \varepsilon')$-indistinguishable from $\mathscr{S}_1 \odot \mathscr{S}_2 \odot \Pi_4$.

Finally, by letting $\mathscr{S}_3 = \mathscr{S}_1 \odot \mathscr{S}_2$, it follows that $\Pi_1 \odot \Pi_3$ is $(R_1, \delta_1, R_3, \delta_3, \varepsilon + \varepsilon')$-indistinguishable from $\mathscr{S}_3 \odot \Pi_4$, as required. □



Π_3
Π_3
Π_3
Π_4
$(R_2,\delta_2,\underset{\sim}{R_3},\delta_3,\varepsilon')$
Π_1
Π_2
$=$
Π_2
$\mathscr{S}_2$
$(R_1,\delta_1,\underset{\sim}{R_2},\delta_2,\varepsilon)$
$\mathscr{S}_1$
$\mathscr{S}_1$
$\mathscr{S}_1$

Figure 2: Proof of Theorem 18. The bold parts of the diagram contain the indistinguishability assumptions. Theorem 11 says that adding the faded parts does not affect indistinguishability. The result follows from transitivity and taking the subsystem enclosed by the dashed lines as the required simulator.

```
On input (keygen):
  1. Stop if $s$ has been recorded.
  2. $(s,v) \leftarrow \mathscr{K}$.
  3. Record $s$. Output $(\mathsf{key}, v)$.

On input $(\mathsf{sign}, m)$:
  1. $\sigma \leftarrow s(m)$.
  2. Output $(\mathsf{signature}, \sigma)$.
On input $(\mathsf{verify}, m, \sigma, v')$:
  1. Output $(\mathsf{verify}, v'(m,\sigma))$.
```

Figure 3: Signature protocol from signature scheme Υ.

Note that the technicalities encountered in the corresponding proof of [2] are avoided due to our use of a fixed communication graph and global resource bounds. A visual version of the proof is given in Fig. 2.

5 Examples

We shall provide three examples of protocol analysis in our framework. The first is the now standard proof that secure digital signature schemes realise a natural signature functionality. The second example shows how a protocol can be used to simulate multiple, independent communication networks, even if only one physical network is available. Similar techniques are very useful in protocol design. The final example shows how we can prove that Diffie-Hellman is a secure key agreement protocol against passive attackers.

5.1 Signatures

A *signature scheme* Υ consists of a probabilistic key generation algorithm $\mathscr{K}$ that outputs descriptions of two algorithms, a signing algorithm s and a deterministic verification algorithm v. The signing algorithm takes a message from some message space as input, and outputs a *signature*. The verification algorithm takes a message and a signature as input, and outputs 0 or 1. We require that for any signature σ output by the signature algorithm for message m, $v(m,\sigma) = 1$.

We say that the signature scheme Υ is (t,n,ε)-secure if for any pair (s,v) generated by $\mathscr{K}$ and any algorithm $\mathscr{A}$ that (i) is given v as input, (ii) is allowed to get signatures on at most n messages and (iii) requires at most time t before producing some output (m,σ), the probability that $v(m,\sigma) = 1$ for a σ that was not given to $\mathscr{A}$ as a signature on m is at most ε.

We get a "protocol" from the signature scheme Υ as described in Fig. 3. It realises the functionality given in Fig. 4 under static corruptions.

```
On $(\mathsf{keygen})$ from player $P$:
  1. Stop if $(P)$ has been recorded.
  2. Record $(P)$.
  3. Hand over $(\mathsf{keygen}, P)$ to $\mathscr{A}$.
  4. Wait for $(\mathsf{keygen}, P, s, v)$ from $\mathscr{A}$.
  5. Record $(P, s, v)$.
  6. Send $(\mathsf{key}, v)$ to player $P$.
On $(\mathsf{sign}, m)$ from player $P$:
  1. Stop if $(P, s, v)$ is not recorded.
  2. Compute $\sigma \leftarrow s(m)$.
  3. Record $(m, \sigma, v)$.
  4. Send $(\mathsf{signature}, \sigma)$ to $P$.
On $(\mathsf{verify}, m, \sigma, v)$ from player $P$:
  1. If $(m, \sigma, v)$ is recorded, send $(\mathsf{verify}, 1)$ to $P$
     and stop.
  2. If $(P', s, v)$ is recorded for some $s$ and honest
     $P'$, send $(\mathsf{verify}, 0)$ to $P$ and stop.
  3. If $v(m, \sigma) = 1$, record $(m, \sigma, v)$.
  4. Send $(\mathsf{verify}, v(m, \sigma))$ to $P$.
```

Figure 4: Signature functionality. The adversary-supplied s and v are descriptions of algorithms, where s is probabilistic and v deterministic. We require that for any allowed m, if $\sigma \leftarrow s(m)$ then $v(m, \sigma) = 1$.

```
On $(\mathsf{keygen}, P)$ from the functionality:
  1. $(s, v) \leftarrow \mathscr{K}$.
  2. Output $(\mathsf{keygen}, P, s, v)$.
```

Figure 5: Simulator for the proof of Theorem 19.

The proof proceeds as follows. First, we apply Theorem 13 to gather all honest protocol machines into a single ITM.

Next, we reorganise the functionality and add bookkeeping work (that is, recording public keys and generated signatures, and looking up keys and message/signature pairs), as well as outsource key generation to a new ITM, namely the simulator given in Fig. 5, so that the new system is identical to the ideal functionality composed with the simulator except that all valid signatures are accepted.

It is a tedious exercise to verify that the resulting system is 0-indistinguishable from the first system. We need extra resources for bookkeeping and sending messages to the simulator, and the amount of work depends on the exact implementation of the bookkeeping, as well as the number N of players, the number n of requests for signing and verification, and the total length l of those requests. Let us fix some resource bound R_1 and δ and increase the resource bound R_1 to R_2 by adding sufficient resources for the bookkeeping work. Now R_2 depends on R_1, N, n and l. Increasing the resource bound to (R_2, δ) and applying Theorem 12 costs us a distinguishing advantage of δ.

Next, we begin rejecting as invalid any technically valid signatures that have not been recorded as honestly generated. Unless a forgery is generated within the time bound, the system is indistinguishable.

Assume that the time bound implied by the resource bound R_2 is t. To bound the probability of the exceptional event, namely the generation of a forgery within time bound t, we construct an attacker against the signature scheme as follows: It chooses one honest user at random from the N honest users. For that user, it uses the verification key it gets as an attacker against the signature scheme. Instead of signing messages, it queries its signing oracle. When the forgery is detected, it is a forgery for the chosen user with probability $1/N$, which means that if our time bound is t and the signature is (t, n, ε)-secure, then the probability of the exceptional event is upper-bounded by $N\varepsilon$. Application of Theorem 17

On (enter, pos) from U:
1. Record (U, pos).

On (leave, pos) from U:
1. Erase any record (U, pos).

On (listen, pos) from $\mathscr{A}$:
1. Record (listen, pos), then hand over $(\mathsf{listen-ok})$ to $\mathscr{A}$.

On $(\mathsf{send}, pos, sid, m, N)$ from U:
1. Stop if (U, pos) is not recorded.
2. If (listen, pos) is recorded, hand over $(\mathsf{send}, pos, sid, m, N)$ to $\mathscr{A}$.
3. Else send $(\mathsf{recv}, pos, sid, m)$ to N.

On $(\mathsf{send}, pos, sid, m)$ from N:
1. If (listen, pos) is recorded, hand over $(\mathsf{send}, pos, sid, m, N)$ to $\mathscr{A}$.
2. Else, for any (U, pos) recorded, send $(\mathsf{recv}, pos, sid, m, N)$ to U.

On $(\mathsf{send}, pos, sid, m, N)$ from $\mathscr{A}$:
1. Stop if (listen, pos) is not recorded.
2. Send $(\mathsf{recv}, pos, sid, m)$ to N.

On $(\mathsf{send}, pos, sid, m)$ from $\mathscr{A}$:
1. Stop if (listen, pos) is not recorded.
2. For any (U, pos) recorded, send $(\mathsf{recv}, pos, sid, m)$ to U.

Figure 6: The radio link functionality. User players are denoted by U, network players by N.

concludes the proof sketch of the following theorem:

Theorem 19. *Let Υ be a (t, n, ε)-secure signature scheme. Let R_1 and R_2 be the resource bounds discussed above. Then the protocol in Fig. 3 $(R_1, \delta, R_2, \delta + N\varepsilon, 2\delta + N\varepsilon)$-realises the functionality in Fig. 4.*

5.2 Radio Link

Sometimes, it is easier to construct complex protocols if we can assume multiple independent communication networks, one for each subprotocol. In the real world, we usually make do with one network, and instead attach some prefix to messages identifying which subprotocol it belongs to.

As an example, we shall describe a functionality for modelling a certain form of radio communication between a set of users and a collection of network operators. Each network consists of many base stations connected by a secure network. Radio communication always happens between users and base stations. Users may communicate with multiple base stations at any one time. Which base stations they communicate with may change over time.

Each network base station is assigned a *position*. Once a user has entered the vicinity, he will receive messages broadcast by the network and he will be able to send messages to the network. The adversary may choose to listen to a certain position, in which case we model the adversary as unrealistically powerful and let the radio link around this base station degenerate into a normal adversary-controlled network. The functionality is described in Fig. 6.

Now we shall describe a multiplexing protocol that essentially pretends to provide n independent radio networks while only using one network. The protocol machine has n input edges, and communicates with one radio link functionality. Every user and network player runs one copy of the protocol machine. The protocol is described in Fig. 7.

Note that the command (Leave, pos) has to be treated with a bit of care. Any message enqueued from $\mathscr{F}_{RL}$ to ϕ_n before (Leave, pos) is received by ϕ_n has to be treated as if (Leave, pos) was never received. This makes the protocol a bit non-intuitive.

If the protocol ϕ_n is properly composed with $\mathscr{F}_{RL}$, it realises n copies of $\mathscr{F}_{RL}$ composed in parallel. Verifying that ϕ_n composed with $\mathscr{F}_{RL}$ and n copies of $\mathscr{F}_{RL}$ composed with the simulator given in Fig. 8 are 0-indistinguishable is as usual a tedious affair.

All that is left is to realise that resource bounds may change, which calls for an application of Theorem 12. We fix a resource bound R_1 and increase R_1 to R_2 by adding the potential extra cost of running

Part 1: User

On (enter, pos) from input i:

1. Record $(pos, i, 0)$ and hand over (enter, pos) to $\mathscr{F}_{RL}$.

On (leave, pos) from input i:

1. Stop if $(pos, i, 0)$ is not recorded.
2. Change the record to $(pos, i, 1)$ and send (leave, pos, i) to self.
3. If no record $(pos, j, 0)$ is left then hand over (leave, pos) to $\mathscr{F}_{RL}$.

On (leave, pos, i) from self:

1. Remove the record $(pos, i, 1)$.

On $(\mathsf{send}, pos, sid, m, N)$ from input i:

1. Stop if $(pos, i, 0)$ is not recorded.
2. Hand over $(\mathsf{send}, pos, i||sid, m, N)$ to $\mathscr{F}_{RL}$.

On $(\mathsf{recv}, pos, i||sid, m, N)$ from $\mathscr{F}_{RL}$:

1. Stop if (pos, i, b) is not recorded.
2. Hand over $(\mathsf{recv}, pos, sid, m, N)$ on input i.

Part 2: Network

On $(\mathsf{send}, pos, sid, m)$ from input i:

1. Hand over $(\mathsf{send}, pos, i||sid, m)$ to $\mathscr{F}_{RL}$.

On $(\mathsf{recv}, pos, i||sid, m)$ from $\mathscr{F}_{RL}$:

1. Hand over $(\mathsf{recv}, pos, sid, m)$ on input i.

Figure 7: The multiplexing protocol ϕ_n. Each instance has n input/output edges, and one connection to an $\mathscr{F}_{RL}$ functionality. Its behaviour depends on whether it is acting for a user player or for a network player.

On (listen, pos) from $\mathscr{A}$:

1. For i from 1 to n, do: hand over (listen, pos) to $\mathscr{F}_{RLi}$, then wait for $(\mathsf{listen{-}ok})$ from $\mathscr{F}_{RLi}$.

On $(\mathsf{listen{-}ok})$ from $\mathscr{F}_{RLn}$:

1. Hand over $(\mathsf{Listen{-}ok})$ to $\mathscr{A}$.

On $(\mathsf{send}, pos, sid||i, m, N)$ from $\mathscr{A}$:

1. Hand over $(\mathsf{send}, pos, sid, m, N)$ to $\mathscr{F}_{RLi}$.

On $(\mathsf{send}, pos, sid||i, m)$ from $\mathscr{A}$:

1. Hand over $(\mathsf{send}, pos, sid, m)$ to $\mathscr{F}_{RLi}$.

On $(\mathsf{send}, pos, sid, m)$ from $\mathscr{F}_{RLi}$:

1. Hand over $(\mathsf{send}, pos, sid||i, m)$ to $\mathscr{A}$.

On $(\mathsf{send}, pos, sid, m, N)$ from $\mathscr{F}_{RLi}$:

1. Hand over $(\mathsf{send}, pos, sid||i, m, N)$ to $\mathscr{A}$.

Figure 8: The simulator from the proof of Theorem 20.

$\mathscr{F}_{RL1} \odot \ldots \odot \mathscr{F}_{RLn}$ instead of $\phi_n \odot \mathscr{F}_{RL}$ and arrive at the following theorem.

Theorem 20. *The protocol $\phi_n \odot \mathscr{F}_{RL}$ $(R_1, \delta, R_2, \delta, \delta)$-realises $\mathscr{F}_{RL1} \odot \ldots \odot \mathscr{F}_{RLn}$.*

5.3 Key Agreement

In this section we illustrate how passive security of the Diffie-Hellman key agreement can be proven within our framework. The section also serves as an example where multi-session analysis gives a tighter security reduction than basic application of the UC-theorem. For this section we shall assume that $\mathbb{G} = \langle g \rangle$ is a cyclic group of prime order p for which the distribution of random triples of the form (g^a, g^b, g^{ab}) and the distribution of completely random triples from $\mathbb{G}$ are (t, ε)-indistinguishable. For simplicity we assume that the protocol players communicate through an authenticated eavesdropping channel modeled as the functionality $\mathscr{F}_N$.

Figure 11 gives a protocol description of the standard Diffie-Hellman key agreement (Π_{KA}) and Fig. 10 describes the functionality that captures its security properties ($\mathscr{F}_{KA}$).

We prove that the protocol Π_{KA} realises $\mathscr{F}_{KA}$ by a sequence of steps where we gradually alter the protocol system Π_{KA}. As the first step we gather all protocol parties into one simulating machine M_1 using Theorem 13. Next we facilitate the use of random self reducibility [6] by altering M_1 to a machine M_2 and introducing a new machine M_{DDH}. At the beginning of the execution, M_2 queries M_{DDH} to receive

On (m,V) from a party U:

1. Send (m,U) to V and hand over $(\mathsf{m},\mathsf{U},\mathsf{V})$ to $\mathscr{A}$.

Figure 9: The authenticated insecure network functionality $\mathscr{F}_N$.

On $(\mathsf{Establish},V)$ from a party U:

1. Generate a new random identifier id.
2. Record (U,V,id) and (V,U,id).
3. Send $(\mathsf{Establish},U,V,id)$ to $\mathscr{A}$.

On $(\mathsf{Output},U,\tilde{k},id)$ from $\mathscr{A}$:

1. Stop if (U,V,id) is not recorded for any V.
2. If V is corrupted then send $(\mathsf{Key},\tilde{k},V)$ to U.
3. Else generate a random $k \in \mathbb{G}$ and send (Key,k,V) to U.
4. In either case, remove the record (U,V,id).

Figure 10: The key agreement functionality $\mathscr{F}_{KA}$.

a random DDH-triple (g_1,g_2,g_k). After this, every h_i from an honest session is generated by raising g_i to a random power and multiplying with a random power of g, $i=0,1$:

$$h_i \leftarrow g_i^{r_i} g^{s_i} \quad .$$

The r_i and s_i are recorded so that the key can be generated as $k \leftarrow g_k^{r_1 r_2} g_1^{r_1 s_2} g_2^{r_2 s_1} g^{s_1 s_2}$. Apart from this M_2 behaves exactly as M_1.

By a bit of calculation, it can be verified that the new system is 0-indistinguishable from the previous one as the triples (h_1,h_2,k) are random DDH-triples. Adding resource bounds by Theorem 12 we get that Π_{KA} is $(R_1,\delta,R_2,\delta,\delta)$-indistinguishable from $M_2 \odot M_{DDH}$, where R_1 is a bound for interactions with Π_{KA} and R_2 is obtained from R_1 by adding the resources needed to generate DDH-triples less efficiently, and simulate the protocol in one vertex.

We can now replace the machine M_{DDH} by a machine M_{RAND} that generates a random triple instead of a DDH-triple. By Theorem 14 the previous partial system is $(R_2,\delta,R_2,\delta+\varepsilon,\varepsilon)$-indistinguishable from the new one if the time component of R_2 is t or less. At this point each key k (with honest players) is a random group element completely independent of the corresponding h_1 and h_2. It follows that this partial system is 0-indistinguishable from a partial system $\mathscr{F}_{KA} \odot \mathscr{S}_{KA}$ where $\mathscr{F}_{KA}$ handles key generation and a machine $\mathscr{S}_{KA}$ handles the simulation of the messages with h_1 and h_2 and every corrupted key agreement session. Adding the potential extra resources needed for running $\mathscr{F}_{KA} \odot \mathscr{S}_{KA}$ instead of $M_2 \odot M_{RAND}$ and applying Theorem 12 one final time we get that $M_2 \odot M_{RAND}$ is $(R_2,\delta+\varepsilon,R_3,\delta+\varepsilon,\delta+\varepsilon)$-indistinguishable from $\mathscr{F}_{KA} \odot \mathscr{S}_{KA}$. By the transitivity of Theorem 10 we get the following theorem.

Theorem 21. *The partial system Π_{KA} $(R_1,\delta,R_3,\delta+\varepsilon,2\delta+2\varepsilon)$-realises $\mathscr{F}_{KA}$ under the assumptions above.*

6 Concluding Remarks

We have defined a novel framework for doing protocol analysis. The framework is based on the ideas embodied in Canetti's Universal Composability [2] and Pfitzmann–Waidner's reactive simulatability [7], but the exact formalisation is novel.

The new framework is based on our ongoing work of analysing practical protocols (for anonymous communication and electronic voting to name a few). It is our opinion that our framework is highly suitable for analysing practical protocols.

```
On input $(\mathsf{Establish}, V)$:
  1. Generate a random $x \in \mathbb{Z}_p$ and set $h_1 \leftarrow g^x$.
  2. Record $(V, x, h_1)$.
  3. Send $((\mathsf{Establish}, h_1), V)$ to $\mathscr{F}_N$.
On $((\mathsf{Establish}, h_1), V)$ from $\mathscr{F}_N$:
  1. Generate a random $y \in \mathbb{Z}_p$ and set $h_2 \leftarrow g^y$.
  2. Send $((\mathsf{Respond}, h_1, h_2), V)$ to $\mathscr{F}_N$.
  3. Output $(\mathsf{Key}, h_1^y, V)$.
On $((\mathsf{Respond}, h_1, h_2), V)$ from $\mathscr{F}_N$
  1. Stop if $(V, x, h_1)$ is not recorded.
  2. Output $(\mathsf{Key}, h_2^x, V)$.
  3. Remove record $(V, x, h_1)$.
```

Figure 11: The key agreement protocol Π_{KA} for party U.

We have provided three simple examples of theorems proven in our framework. The first example is the now standard proof that a natural digital signature functionality can be realised using secure digital signatures. The second example is less standard, but shows how our framework naturally encapsulates the ideas concerning joint state composability [3]. The last example shows how passive security of the Diffie-Hellman key agreement can be proven in our framework.

References

[1] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Proc. of the 38th Annual Symposium on Foundations of Computer Science (FOCS'97), Miami Beach, Florida, USA*, pp. 394–403. IEEE, October 1997.

[2] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. http://eprint.iacr.org/2000/067, December 2005.

[3] R. Canetti and T. Rabin. Universal Composition with Joint State. In *Proc. of the 23rd Annual International Cryptology Conference (CRYPTO'03), Santa Barbara, California, USA, LNCS*, volume 2729, pp. 265–281. Springer-Verlag, August 2003.

[4] K. Gjøsteen and L. Kråkmo. Universally Composable Signcryption. In *Proc. of the 4th European PKI Workshop on Theory and Practice (EuroPKI'07), Mallorca, Balearic Islands, Spain, LNCS*, volume 4582, pp. 346–353. Springer-Verlag, June 2007.

[5] O. Goldreich. *Foundations of Cryptography*. Cambridge University Press, 2001.

[6] M. Naor and O. Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Journal of the ACM*, 51(2) pp. 231–262, March 2004.

[7] B. Pfitzmann and M. Waidner. A Model for Asynchronous Reactive Systems and its Application to Secure Message Transmission. In *Proc. of the 22nd IEEE Symposium on Security and Privacy (S&P'01), Oakland, California, USA*, pp. 184–200. IEEE, May 2001.

[8] V. Shoup. Sequences of Games: A Tool for Taming Complexity in Security Proofs. `http://eprint.iacr.org/2004/332`, November 2004.

Kristian Gjøsteen is an associate professor at the Department of Mathematical Sciences, Norwegian University of Science and Technology (NTNU), working on the analysis of cryptographic protocols. He received his PhD from NTNU in 2004.

George Petrides is a Postdoctoral fellow at the Department of Mathematical Sciences, Norwegian University of Science and Technology, since 2009. He received his PhD from the University of Manchester, UK, in 2006. His research interests include FSR sequences, privacy enhancing technologies, formal proof techniques and group theoretic cryptography.

Asgeir Steine is a Ph.D. student at the Department of Mathematical Sciences, Norwegian University of Science and Technology. His research topics include privacy enhancing protocols and cryptographic protocol analysis. He received his M.E from the same department in 2007.