IST-2002-507932

ECRYPT

European Network of Excellence in Cryptology

Network of Excellence
Information Society Technologies

D.PROVI.8

Summary Report on Models and Definitions for Cryptographic Protocols

Due date of deliverable: 31. January 2008
Actual submission date: 07. February 2008

Start date of project: 1. February 2004
Duration: 4 years

Lead contractor: University of Salerno (UNISA)

Revision 1.0

<table>
<thead>
<tr>
<th>Dissemination Level</th>
<th>PU Public</th>
<th>PP Restricted to other programme participants (including the Commission services)</th>
<th>RE Restricted to a group specified by the consortium (including the Commission services)</th>
<th>CO Confidential, only for members of the consortium (including the Commission services)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Summary Report on Models and Definitions for Cryptographic Protocols

Editors
Giuseppe Persiano (UNISA), Ivan Visconti (UNISA)

Contributors
Christian Cachin (IBM),
Sebastian Gajek (RUB),
Thomas Gross (IBM),
Tibor Jager (RUB),
Olivier Pereira (UCL),
Ahmad Sadeghi (RUB),
Berry Schoenmakers (TUE),
Julien Stern (CRY),
Carmine Ventre (UNISA)

07. February 2008
Revision 1.0

The work described in this report has in part been supported by the Commission of the European Communities through the IST program under contract IST-2002-507932. The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.
Abstract

This report focuses on models and definitions for cryptographic protocols. We consider different notions of security and discuss some of the most widely used models and set-up assumptions (e.g., random oracle, common reference string, bare public keys, trusted computing base).

We consider as examples some two-party games (e.g., zero-knowledge proof systems, commitment schemes, ), present the current state-of-the-art and stress the most important open problems.

Finally we overview the implications on practical protocols.
# Contents

1 Introduction .................................................. 1

2 Security Models ............................................. 3
   2.1 The Simulation Paradigm .................................. 3
   2.2 Composability Security Frameworks ......................... 4
      2.2.1 Universal Composability ............................... 5
      2.2.2 Reactive Simulatability ............................... 6
   2.3 Formalizing Composable Security ......................... 7
      2.3.1 Modeling Concurrent Behaviors in a Cryptographic Setting ............................... 7
      2.3.2 Modeling Bounded Computations in a Distributed Setting ......................... 9
   2.4 Generic Group Model .................................. 10
   2.5 Relaxed Security .................................. 12

3 Set-Up Assumptions ........................................ 13
   3.1 The Plain Model .................................. 13
   3.2 Models with Public Keys .................................. 13
   3.3 Reference String and Random Oracle Models ................. 14
   3.4 Broadcast Channel .................................. 15
   3.5 Trusted Computing Base .................................. 16

4 State-of-the-Art and Open Problems .......................... 17
   4.1 Zero Knowledge Proof Systems .......................... 17
   4.2 Commitment Schemes .................................. 23
      4.2.1 Other Notions of Commitments ......................... 24
   4.3 Zero Knowledge Sets .................................. 27
   4.4 Implications on Practical Protocols ....................... 28
Chapter 1

Introduction

This document presents an overview of models and definitions for cryptographic protocols.

A cryptographic protocol is a procedure executed by parties in order to perform a given task while satisfying some security requirements.

The provable security framework is currently adopted as the standard approach to formally assess the security of a cryptographic protocol, in contrast to the use of heuristic arguments. In the provable security framework it is first specified the role of the honest parties and of the adversaries in an abstract model. Then it is claimed that, under some assumptions, there is no succeeding adversary for the protocol. The proof of security is referred to as reduction and produces and adversary that violates the assumptions on top of a succeeding adversary of the protocol.

Security models. In this framework a central role is therefore played by the models that are considered and that define the capabilities of both the honest parties and the adversaries.

This document therefore starts with an overview of some of the most accepted security models, focusing in particular on the recently proposed composability frameworks. These frameworks are a strong implementation of the well known simulation paradigm that is the de-facto standard model for claiming the security of a cryptographic protocol. We will explore some of these frameworks and consider in particular the notion of composable security. A protocol in this framework has to preserve its security even in case the adversary can play at the same time other protocols. This line of research is now central in Cryptography since many protocols are executed in networks of computers that run many tasks simultaneously.

Set-up assumptions. An important ingredient for the security of a protocol is given by the assumptions on which its security is based. Following the complexity-theoretic approach, one can specify the existence of secure primitives that are then used in the protocol and that guarantee its security. Another class of assumptions is instead represented by the set-up assumptions, i.e., the existence of constraints and features that affect the honest parties and the adversaries that participate to the protocols.

In this document we will overview some of the most widely used set-up assumptions and stress their impact on allowing the design of cryptographic protocols, in particular for overcoming impossibility results and obtaining efficient constructions.

The last part of this document focuses on some specific cryptographic protocols as secure proof systems and commitment schemes. We will consider the current state-of-the-art and
discuss the combination of security models, definitions, complexity-theoretic and set-up assumptions for obtaining cryptographic protocols for these tasks. We will stress the current open problems giving therefore evidence that this is an active and challenging area in the foundations of Cryptography.
Chapter 2

Security Models

The notion of a secure stand-alone protocol assumes that a protocol is run by the parties only once, in isolation with respect to other executions of the same protocol and of other protocols. This is quite a strong and unrealistic assumption. Early results showed that essentially any efficient functionality can be securely computed (see [124] for the two-party case and [15, 38, 71] for the multi-party case) and these can be considered as starting points for investigation in more realistic scenarios.

2.1 The Simulation Paradigm

Security of the protocols is proved by using the so-called simulation paradigm consisting in showing an efficient algorithm that simulates the view of an adversary. In the stand-alone scenario, the adversary plays the protocol impersonating one (or more) players, and his view includes all data exchanged during the execution of the protocol and the private random coins of the players he is impersonating.

More specifically, the standard notion of security of a protocol considers the notion of securely computing a functionality $f$. The assumed adversary for the protocol is used to design an ideal-setting adversary. In the ideal setting there exists a trusted third party that simply gets the inputs of each party, evaluates the function and sends to each party the corresponding output. Informally, a protocol $\rho$ securely computes a functionality if for any real-setting adversary $A$ there exists an ideal-setting adversary $S$ such that the behavior of $A$ in the real-life setting is simulated by $S$ in the ideal setting. Since no adversary can be successful in the ideal setting, this implies security in the real setting.

Considering concurrent executions of protocols is a more realistic assumption. For example, at any given point in time, several protocols are being concurrently executed on the Internet and it is conceivable that an adversary tries to break (and he might even succeed) the security of the protocol by coordinating his actions in several executions. In this case the adversary does not simply take part in the protocol but can also start new concurrent executions of the protocols. The work of the simulator is much more complex in this setting since in general a simulator for a stand-alone protocol is not a good simulator anymore when the protocol is concurrently executed with other protocols.

Recently the research is focusing on these more realistic scenarios where protocols have to be secure even when they are concurrently composed with themselves or with other protocols. More precisely, the following two main types of composition have been investigated.
1. Self composition: in this case a single protocol is concurrently executed any polynomial number of times. This type of composition has been, for instance, used in the notion of concurrent zero knowledge [60].

2. General composition: in this case a protocol is concurrently executed along with other protocols. In particular the other protocols may even have been designed maliciously in order to attack the security of the given protocol. This type of composition has been, for instance, used in the notion of universal composability [22].

Both types of compositions can be relaxed by considering the following two limitations of the power of the adversary.

1. Bounded concurrency: in this case there exists a bound on the concurrency of the adversary. This is in general achieved by assuming a bound on the number of protocols that the adversary can concurrently run or on the communication complexity that the adversary can generate in concurrent sessions.

2. Timing/Synchronicity assumptions: in this case the adversary has only a partial control of the communication channel. More specifically, he can delay the messages up to a given bound.

**Man-in-the-middle Adversary.** A man-in-the-middle is an adversary that simultaneously participates to two executions of a two-party protocol and acts as left party in one and as right party in the other. The adversary has complete control over the scheduling of the messages. Informally, a protocol is said *non-malleable* if the power of the man-in-the-middle adversary during the real game is not greater of the one in the simulated game. A protocol that is secure against man-in-the-middle attacks is referred to as non-malleable. Several known constructions for non-malleable protocols in the plain model do not preserve non-malleability in case the man-in-the-middle can concurrently play a polynomial number of sessions as both left and right party. A protocol that is secure in this model is referred to as concurrent non-malleable.

The need of managing concurrent executions of protocols and sophisticated man-in-the-middle attacks motivated the introduction of new frameworks that focus on preserving the security of the protocols even in case their executed under asynchronous concurrent composition.

### 2.2 Composability Security Frameworks

Essentially, two frameworks have proliferated that are based on the ideal-world/real-world paradigm: The framework of Universal Composability (UC) presented in [22] and the framework of Reactive Simulatability (RSIM) introduced in [109]. The analysis within these frameworks yield in particular the following advantages over previous cryptographic models: Firstly, the security of protocols holds under general composition and with arbitrary sets of parties. This is a strong security guarantee that simplifies the design of cryptographic systems. One can handle protocols as components and simply plug them into arbitrary higher-level protocols while the complexity of analyzing the composite protocol is considerably reduced. Secondly, the ideal functionality is free from any probabilism by abstracting away cryptography. Thus, it can be expressed in abstract term algebra following the *Dolev-Yao model* [59] such that machine-assisted proofs are feasible using, e.g., theorem provers or model checkers. Since
hand-made proofs are prone to errors, tool-supported analyses are an option to affirm the results.

2.2.1 Universal Composability

In the UC framework, Interactive Turing Machines interact in two worlds (see Fig. 2.2.1). The real-world model comprises honest parties and the adversary $A$. The parties run a protocol $\pi$ in order to compute a cryptographic task. $A$ controls the communication and potentially corrupts the parties. The ideal world includes “dummy” parties who interact with an ideal functionality $F$, running the ideal protocol $\phi$. The functionality $F$ represents a trusted party that carries out the same cryptographic task. It simply obtains the inputs of the players and provides them with the desired outputs. The ideal-world adversary $S$ (dubbed the simulator) is allowed to delay messages, however it is unable to gain knowledge of any inputs/outputs except the functionality $F$ is willing to do that. Intuitively, the ideal functionality captures the security requirements of a given cryptographic task we expect from the real-world protocol $\pi$ and defines the adversarial corruption model we consider in that setting. In the UC framework there exists an additional entity called the environment $Z$ who has to distinguish between the two worlds. Therefore, the it feeds all parties with input, retrieves their outputs, and interacts with the adversary in an arbitrary way throughout the computation.

Figure 2.1: The ideal-world/real-world paradigm in the UC framework. Parties I and R execute protocol $\pi$ in the real world while dummy parties I’ and R’ use help of $F$ to compute the same task. (Arrowed lines denote input/output communication; straight lines denote interactive communication.)

The ideal-world adversary $S$ does not perceive the message exchange between the real-world parties and has to simulate the interaction in order to mimic the behavior of $A$. Then, security of protocol $\pi$ is captured by the fact that every attack $A$ mounts in the real world, $S$ carries out in the ideal world. The protocol security is implied, since the ideal world is such that no attacks can be carried out. We have then that the outputs $Z$ retrieves from the execution of $\phi$ with the dummy player and $S$ and the execution of $\pi$ with the real-world players and $A$ are indistinguishably distributed. Indistinguishability means in this case computational indistinguishability ("$\approx"$). Informally, a protocol $\pi$ is then said to securely emulate an ideal-world protocol $\phi$. In addition, a protocol $\pi$ is said to securely realize a cryptographic task, if for any real-world adversary $A$ that interacts with $Z$ and real players running $\pi$, there exists an ideal-world simulator $S$ that interacts with $Z$, the ideal functionality $F$, and the
dummy players running the ideal protocol $\phi$, so that no probabilistic polynomial time-bounded environment $Z$ is able to distinguish whether it is interacting with the real-world $A$ or the ideal-world adversary $S$. A more general definition is:

**Definition 2.2.1** A protocol $\pi$ UC-emulates protocol $\phi$ if for any adversary $A$ there exists an adversary $S$ such that for all environments $Z$ that output only one bit:

$$\text{UC-EXEC}_{\phi,S,Z} \approx \text{UC-EXEC}_{\pi,A,Z}$$

A protocol $\pi$ UC-realizes an ideal functionality $F$ if $\pi$ UC-emulates the ideal protocol for $F$.

It is easy to see that ideal protocol $\phi$ is the protocol that defines the communication between $F$ and the dummy players that simply forward their inputs and outputs. This is equivalent to the fact that $F$ bypasses the dummy players.

A key point of the UC framework is the composition theorem. It guarantees composition with arbitrary sets of parties. Consider a protocol $\rho$ that operates in the $F$-hybrid model, meaning that parties interact in the normal way and in addition can invoke an arbitrary number of copies of the functionality $F$. We call the invocation of $F$ subroutine-respecting, if only $\rho$ is permitted to receive the inputs and outputs of the ideal functionality. Then, the following holds.

**Theorem 2.2.2 (Universal Composition)** Let $\pi$ and $\phi$ be two subroutine-respecting probabilistic polynomial time-bounded protocols such that $\pi$ UC-emulates $\phi$. Then the subroutine-invoking protocol $\rho^{\pi/\phi}$ UC-emulates $\rho$ for any probabilistic polynomial time-bounded protocol $\rho$.

If $\pi$ UC-emulates $\phi$, we have that there is no $Z$ that can distinguish with non-negligible probability between the players running $\pi$ and players running $\phi$ in the presence of the adversary. The subroutine-respecting invocation ensures that the surrounding protocol $\rho$ feeds $\pi$ and $\phi$ in the same way so that the outputs are identically distributed. The composition theorem prevails that replacing the instance of $\pi$ with an instance of $\phi$ does not change the behavior of $\rho$ with respect to any probabilistic polynomial time-bounded adversary; we have a symmetry between the case that $\rho$ interacts with $\pi$ and $\phi$ in the presence of the adversary.

The main attraction of the composition theorem follows from the fact that if $\phi$ UC-realizes $F$ then the real-world protocol $\rho$ can replace the invocation of subroutine $\pi$ by calling the ideal functionality.

### 2.2.2 Reactive Simulatability

In [109] a notion of security-preserving refinement, called reactive simulatability has been introduced. Consider the real and an ideal systems. Reactive simulatability requires that “everything that can happen to users” of the real system in the presence of an arbitrary adversary $A$ can also happen to the same users with the ideal system and another adversary $A'$. Moreover, the notion of “everything that can happen to users”, includes the inputs and outputs to the system and also the communication with the adversary. This includes the fact that the adversary can guess partial information about users. The most important considered variations of simulatability are universal and black-box simulatability. Universal simulatability, states that $A'$ has to be independent of the actual users of the protocol. Black-box simulatability states that $A'$ consists of the original adversary $A$ and a simulator that may only depend on the protocol itself.
In [109] a composition theorem is proved. Specifically, if a larger system is designed based on a specification of a subsystem, and the implementation of the subsystem is later plugged in, then the entire implementation of the larger system is as secure as its design in the same sense of reactive simulatability. This theorem holds for all variants of simulatability (general, universal, and black box), but it is restricted to replacing one system. Recently in [5], a more general composition theorem for black-box simulatability has been proposed by showing that a polynomial number of arbitrary systems can be composed still preserving the simulatability relation.

2.3 Formalizing Composable Security

A central motivation for composable security definition is the ability to capture security in large distributed systems, where the execution environment is not known in advance. This transition from a closed to an open and distributed environment however raises important modeling challenges. Notably, one may wonder:

1. How to model the concurrent execution of the various system components?
2. How to model bounded computational power in a distributed setting?

In the rest of this section, we describe the challenges associated to these questions and review the most characteristic proposed solutions.

2.3.1 Modeling Concurrent Behaviors in a Cryptographic Setting

In a distributed setting, the uncertainties on the timing of the concurrent execution of the different system components are traditionally captured through nondeterministic choices [116].

In order to be able to compare probabilistic executions of distributed systems, and to talk about their computational indistinguishability, it is necessary to define a mechanism to resolve this nondeterminism, leaving only probabilistic choices unspecified. This is achieved by introducing schedulers, that resolve the nondeterministic choices.

Classical perfect information schedulers, that is, schedulers that have full information about the past execution and current state of the system, and can use this information to resolve nondeterministic choices, are however not suitable in a security context: those schedulers could disclose secret information of protocol parties to adversarial components through the way they resolve nondeterministic choices (e.g., by scheduling a transition \( a_0 \) if the first bit of a key is 0 and a transition \( a_1 \) if that bit is set to 1).

Sequential scheduling. In order to avoid such unrealistic behaviors, several cryptographers, including Canetti [22], Pfitzmann and Waidner [109] decided to adopt, in their communication model for composable security, a fixed scheduling strategy that avoids the need of using an explicit scheduler: sequential scheduling.

With sequential scheduling, a special component, called the master scheduler, which is usually the environment (also called user in the terminology of [109]), is activated first. Each active component has then the possibility to perform transitions, and to decide which component will be activated next (by sending him a message in [22], or by activating a special type of port in [109]). If an active component halts without activating any other system
component, the master scheduler becomes active. (The impact of the component chosen to assume the role of master scheduler is analyzed in [49].)

An attractive feature of this scheduling mechanism, which is the most prominent in the current cryptographic literature, is that it makes any closed system (that is, any system that cannot receive any input) purely probabilistic, ruling out all difficulties associated to the use of nondeterminism. On the other hand, it is hard to provide any natural interpretation of sequential activation in a distributed system in which most components can become active and send messages at any time.

**Using explicit schedulers.** Another approach was adopted by Canetti et al. in [23, 24], in which nondeterministic choices are resolved by an oblivious scheduler, that is, a scheduler that has no information at all about the current state of the system or about the transitions that are enabled at any time. In order to avoid the need to consider schedulers that have a length exponential in the length of the messages to be transmitted, the set of actions that can be performed by the different components is partitioned into tasks, with the constraint that tasks must be defined in such a way that, in any state of the system, at most one action is enabled per task. When a task partition is defined, a scheduler can simply take the form of an arbitrary sequence of tasks. This allows, for instance, to gather all $2^n$ actions corresponding to the sending of a nonce of length $n$ in one single task, and to include that task in the scheduler instead of requiring the scheduling of $2^n$ actions in order to make sure that a chosen nonce had a chance to be transmitted.

In this context, indistinguishability of two systems can be defined as an implementation relation requiring that every probabilistic execution of the first system controlled by some task scheduler can be matched by a probabilistic execution of the second compared system when it is controlled by some other scheduler.

Including nondeterminism in the communication model used for cryptographic protocol analysis appears to provide a more realistic abstraction of the behavior of a distributed system than sequential scheduling (in a distributed system, components are typically not able to predict which will be the next active component of the network). Furthermore, it appears that such a modeling choice has a strong impact on security definitions, as it was shown in [25] that the indistinguishability definitions, which underly all security definitions, are actually incomparable according to the adoption of sequential scheduling or explicit schedulers.

Other communication models including nondeterministic choices were proposed recently, including [20, 37, 40], in which schedulers gain partial information about the state of the system and are computationally bounded processes instead of mere sequences of tasks.

In a related way, a hybrid approach was proposed much earlier in the context of the PPT calculus [85, 92]. In this process calculus, internal (invisible) transitions are prioritized over external (visible) transitions and the selection of such a transition is performed according to a fixed strategy: the next action is selected among all enabled actions by performing a random choice according to the uniform distribution. On the other hand, visible actions are scheduled by means of an explicit scheduler, which also is a PPT process that has only partial information on the enabled actions.

The relation between all these different communication models, and their practical impact on security notions is however not clear, but would be of fundamental importance for our understanding of the scope of the security proofs we build.

**Open Problem No. 1:** Investigate communication models that can be adopted
for describing the execution of cryptographic protocol in distributed systems, understand their impact on security notions, and the extent to which they provide a sound abstraction of the actual concurrent behaviors.

2.3.2 Modeling Bounded Computations in a Distributed Setting

Security of cryptographic protocols usually holds only in the presence of computationally bounded parties. This is traditionally captured by considering adversaries that are limited to perform a polynomial number of steps in the length of their inputs (see, e.g., [72]).

This notion, which provides a fairly smooth and consistent treatment in non-concurrent settings (e.g., for defining stand-alone interactive proof systems), becomes much more cumbersome to define in a distributed computation context.

A first class of definition candidates, adopted by Canetti in the initial version of [22] and Pfitzmann and Waidner in [109] for instance, consists in requiring that the number of computational steps taken by each system component (that is, ITM or PIOA according to the model) must be bounded by some polynomial in the security parameter. Such a condition however raises unnatural behaviors: for instance, an encryption service that comes with a bound that is only a function of the security parameter may stop answering requests at some point, which would provide the adversary a way to count the number of messages that the service encrypted for honest users. In order to avoid such exhaustion phenomena, mechanisms such as length functions [4] or guards [49] have been adopted, which provide a way of pre-processing messages and, in many cases, avoid exhaustion.

A more general approach was adopted by Canetti (in the 2005 versions of [22]) and by Hofheinz et al. [80], in which the running time of components is allowed to depend on the length of input messages and security parameter. In these works, further refined by Küsters [83] and then by Hofheinz et al. [79], definitions are tailored in such a way that, as long as the environment/user remains polynomial time-bounded, so will be the behavior of the other system components. Important challenges addressed in those works include obtaining definitions that would be recognizable, capture meaningful behaviors and attacks, and provide robust composable security definitions. In particular, Küsters investigates conditions that make collapse various composable security notions that one would plausibly expect to be equivalent, including Strong Simulatability [49], Black-Box Simulatability [109], and Universal Composability/Simulatability [22, 109], ensuring the robustness of security properties to slight security definition variants.

All those works use a sequential scheduling mechanism providing pure probabilistic execution of closed systems, avoiding the use of schedulers. Canetti et al. [24] however make use of the explicit schedulers included in their model to define polynomial-time bounded computation: they require the number of steps taken by the various system components to be polynomially bounded during each transition, then require the scheduler to enable only a polynomial number of transitions, all the polynomials being computed in the security parameter alone. However, those definitions are not known to make collapse the various composable security notions.

1Note that the immediate idea that consists in requiring the total computation performed by a component to be bounded by a polynomial in the total input length cannot be adopted: two components could just “play ping-pong”, sending each other messages that have a length twice of the last received message, which would result in exchanging messages of exponential length in the number of rounds.
Open Problem No. 2: Investigate and relate protocol execution models that allow bounding the computational power of (some) components in a simple and natural way, while capturing the natural behavior of parties, allowing to detect meaningful attacks, and providing robust security definitions.

2.4 Generic Group Model

The security of modern asymmetric cryptographic systems depends on the conjectured intractability of certain computational problems. Popular examples are the ElGamal encryption and signature scheme [61] and the Diffie-Hellman protocol [57], which rely on the hardness of the discrete logarithm problem in a certain group $G$. An algorithm which is able to compute discrete logarithms can break cryptosystems which depend on the discrete logarithm problem. In the case of the Cramer-Shoup encryption scheme [41] it is even known that the converse is true: An algorithm which is able to break the encryption scheme must be able to compute discrete logarithms.

Since asymmetric cryptosystems depend on computational problems that are only conjectured to be “hard” (i.e., computationally infeasible for sufficiently large parameters), it is interesting to consider the computational complexity of these problems. Proving the “hardness” of a problem means to prove a lower bound on the computational complexity. For a general model of computation, such as the Turing machine model, no applicable lower bound proofs are known. Therefore more restricted models of computation have to be considered.

One such restricted model generic group model. The validity of many important, but in a general model of computation unproven, cryptographic assumptions has been proven in the generic group model [94, 119, 48]. Generic group algorithms are a class of algorithms which operate on the structure of an algebraic group $^2$. However, the algorithms may only exploit the defined properties of an abstract group, not specific properties of a given group representation. This means that the algorithms have only access to trivial informations about the manipulated group elements, such as the equality relation. Therefore the algorithms can not benefit from specific properties of a particular group representation. This implies that the algorithm works for arbitrary groups in a similar way, regardless of the representation of the group elements. The Pollard’s Rho algorithm [113] for computing discrete logarithms works in arbitrary groups, and hence is a generic algorithm. In contrast, index calculus algorithms [88] for computing discrete logarithms in $\mathbb{Z}_p^*$ are much more efficient, but not generic, because special properties of the group elements are exploited, namely the fact that group elements may be treated as factorizable integers and that the factorization is compatible with the group operation.

A generic lower bound on the complexity of a problem implies that there are no algorithms which solve the given problem faster in arbitrary groups. Hence, customized algorithms must be devised for any particular representation of the group in order to solve the problem more efficiently.

Additionally, a lower bound proof in the generic group model may serve as an indicator that a newly suggested (and therefore not well-investigated) problem might indeed be useful for cryptographic applications. Some recently proposed variants of well-known and well-studied cryptographic assumptions offer new possibilities to design cryptosystems and

\[ ^2 \text{The generic group model is not limited to the algebraic structure of a group, but can be extended to a generic ring model by adding an adequate operation, as done in [84] and [19], for example.} \]
cryptographic protocols. The generic group model is an important tool to gain confidence in using such newly suggested, and therefore not well-studied, cryptographic assumptions [18]. However, it is important to note that the generic group model does usually not provide a proof that the new assumption is valid. Instead, it may only act as an indicator that a cryptographic scheme which relies on some new assumption is not totally flawed.

We model a generic group in the following way, as proposed by Shoup in [119]: Let $G$ be a group of order $n$ and let $S_n \subseteq \{0, 1\}^{\lceil \log_2 |G| \rceil}$ be a set of $n$ distinct bit strings. Let

\[ \sigma : G \rightarrow S_n \]

a bijective, random encoding function which encodes the group elements as random, but unique binary strings. The random encoding ensures that the representation of the group $G$ has only the defined properties of an abstract group.

**Definition 2.4.1** A generic group algorithm $A$ is a (probabilistic) algorithm which takes as input a $r$-tuple of encoded group elements $(\sigma(x_1), \ldots, \sigma(x_r))$, $x_i \in G$, $1 \leq i \leq r$. The oracle may query a generic group oracle $O$ to perform arithmetic operations on encoded group elements. The output of $A$ is denoted $A(\sigma; x_1, \ldots, x_r)$.

The input to a generic group algorithm is usually the public part of a problem instance which should be solved by the algorithm. An algorithm $A$ makes a query to the oracle $O$ by specifying encoded elements $\sigma(x_i), \sigma(x_j)$ and an operation $\circ$. $O$ computes $x_i \circ x_j$ and returns $\sigma(x_i \circ x_j)$ (cf. Figure 2.2). The oracle $O$ provides at least the group operation and, possibly, inversion of elements. The generic group can be extended to a generic ring by adding an adequate multiplication operation. Of course additional operations, depending on the respectively given problem, may be provided by adding adequate oracles.

Note that the oracle may use an arbitrary group $H$ which is isomorphic to $G$ for internal representation of the group elements, because the internal representation of the group is hidden by the random encoding function. For example, if $G$ is a cyclic group of order $n$, then $O$ may use the additive group $\mathbb{Z}_n$ for internal representation, since $G \cong \mathbb{Z}_n$ holds for arbitrary cyclic groups $G$ of order $n$.

Usually it is assumed that $A$ has unbounded memory capacity, the time complexity of $A$ is measured by the number of oracle queries.
Example: The discrete logarithm problem in a cyclic group of order \( n \) in Shoup’s generic group model can be stated as: Given a generating element \( \sigma(1) \), an encoded group element \( \sigma(x) \) and the group order \( n \), determine \( x \) with the help of an oracle which computes the group law.

2.5 Relaxed Security

Recently, a relaxed notion of simulation has been studied and actually used for achieving new results that are impossible with the original notion. The new notion allows the simulator to run in super-polynomial time. The basic idea is that starting by assuming the existence of cryptographic primitives that are hard with respect to adversaries running in time \( t \), the simulator is allowed to run in time \( t' < t \) and the existence of a distinguisher between the real and ideal games, can be used to break in time \( t' < t \) the security of the cryptographic primitives thus reaching a contradiction. This technique is referred to as complexity leveraging and has been originally used for proving the soundness of some resettable zero knowledge [28, 90, 54, 55] arguments in the public-key model. Then in [102] super-polynomial-time simulation has been used for achieving round efficient concurrent zero knowledge and in [103] for achieving round-efficient bounded-concurrent secure two-party computation. In [11] general concurrent composition has been achieved in the plain-model by considering this relaxed notion of security. Moreover, in [10] complexity leveraging has been used for obtaining constant-round argument systems that are secure under bounded reset attacks, while in [43] it has been used to obtain efficient non-interactive zero-knowledge in the key registration model.
Chapter 3

Set-Up Assumptions

In this section we overview some of the most attractive set-up assumptions that have been studied so far.

3.1 The Plain Model

The plain model assumes no existing infrastructure (i.e., no set-up assumption). For secure two-party computation, the only additional assumption is the existence of authenticated channels. A message sent using an authenticated channel is guaranteed to be delivered uncorrupted.

3.2 Models with Public Keys

Two main variations of public-key models have been considered in the literature.

**Key registration model.** The strongest one is a trusted public-key model where each user possesses a public key and there exist trusted third parties that certify such ownerships. This model is referred to as the *Key Registration* model and has been used in [9] in order to obtain universally composable protocols. As additional requirement, in [9] the secret key of each user has to be known by a trusted third party (it is only required that it is trusted for the user) or the user has to give a proof of knowledge in isolation. This model has been more recently used for achieving efficient non-interactive zero knowledge for many interesting languages [43].

A variation of the key registration model, referred to as ”key registration with knowledge” has been proposed for achieving universal composability with global setup [26], a stronger form of universal composability that also achieves deniability (thus outperforming the CRS model where deniability can not be achieved for many functionalities).

**Bare public-key model.** The weakest one is the untrusted public-key model where each user possesses a public key but there exists no trusted third party. A very weak variation of this model, namely the *Bare* public-key (BPK, for short) model, has been proposed in [28] to achieve round-efficient concurrent and resettable zero knowledge\(^1\). In this model there

\(^1\)The notion of resettable zero knowledge extends the notion of concurrent zero knowledge by allowing the adversarial verifier to play concurrent sessions with cloned provers (i.e., the provers can be forced to use the same randomness in different sessions).
exists a public (not necessarily trusted) file that contains the public keys of the verifiers. Among the different proposed set-up assumptions, the bare public-key model has the following advantages: 1) it is not based on any trusted third party; 2) no timing assumption is made beyond the separation between set-up stage and proof stage; 3) the set-up stage is non-interactively performed only by the verifiers. Consequently, the public-key model is, from among the currently proposed models, the one that makes the weakest set-up assumptions and, in particular, it is weaker than the widely accepted Public Key Infrastructure model.

3.3 Reference String and Random Oracle Models

**The common reference string model.** The common reference string (CRS, for short) model assumes that a trusted third party generates a string according to a given distribution. The parties that run a protocol receive as additional input such a string.

An interesting restriction of the reference string model is the shared random string model [17] (SRS, for short). Here, the string generated by the trusted third party has the uniform distribution. This model is interesting since in contrast to the common reference string model, here there is no trapdoor information that corresponds to the CRS. Moreover, a trusted source of randomness is not necessarily a trusted third party but can be a natural process. Finally, shared random strings can be obtained by means of coin-tossing protocols.

The CRS and SRS model have been used for implementing secure protocols under minimal complexity-theoretic assumptions and with efficient instantiations, indeed universally composable multi-party computation for any well-formed functionality has been achieved in these models in [30].

**The random oracle model.** A random oracle behaves as a random function of type \( \{0,1\}^* \rightarrow \{0,1\}^k \), say, for some fixed integer \( k \geq 0 \). This means that when the random oracle is queried at an input value, the output value is a uniformly random bit string of length \( k \) (independently distributed from all other output values), and whenever a query is repeated (i.e., the same input value is used), the oracle will yield the same output.

The relation to the shared random string model is that given a random oracle one may easily create a shared random string by querying the oracle at a sequence of inputs \( 0,1,10,11,100, \ldots \) say, and concatenating the output strings to obtain a random string. Hence, assuming a random oracle is a stronger assumption than assuming a shared random string.

A random oracle models the use of a cryptographic hash function as a building block in cryptographic schemes. Clearly, a random function \( f : \{0,1\}^* \rightarrow \{0,1\}^k \) fulfills the requirements of a cryptographic hash function, if \( k \) is sufficiently large. Namely, it is one-way, second-preimage resistant and collision-resistant. However, being a random function it is also excluded to enjoy homomorphic properties such as \( f(x \oplus y) = f(x) \oplus f(y) \), which is useful because such properties could make schemes relying on \( f \) insecure.

In practice, the random oracle is implemented using a concrete hash-function such SHA-1 or SHA-256, with \( k = 160 \), and \( k = 256 \), respectively. The security of the resulting schemes is only heuristic, since clearly a function like SHA-1 is not a random function. Thus, the practical upshot of the random oracle model is that a scheme proved secure in it can only be broken if the attacker takes into account specific properties of the concrete hash function used. For example, the Fiat-Shamir heuristic [64] is provably secure in the random oracle model.
This approach was popularized by Bellare and Rogaway, starting with [14]. Limitations of this approach were emphasized in [31], where it is shown that there exist protocols which are secure in the random oracle model, but become insecure as soon as the random oracle is instantiated with a concrete function.

Open Problem No. 3: It interesting to extend the result of [31] to a class of ‘real-life’ protocols which would actually be used in practice.

A step into this direction was recently made by Bellare et al. [12], who showed that a variant of ElGamal used in combination with a symmetric cipher yields an insecure method for hybrid encryption (yet, the symmetric cipher is constructed in a specific way, depending on the ElGamal variant). Also, Nielsen [96] already showed that a distinction needs to be made in the model whether the random oracle is programmable or non-programmable by showing that these models can be separated (for non-interactive non-committing encryption).

3.4 Broadcast Channel

Whenever more than two parties are concerned by a cryptographic protocol, there is a fundamental distinction regarding their communication medium: whether it supports broadcast or not. The basic case is that parties are connected by pairwise point-to-point channels, which may additionally be authenticated or private, i.e., support integrity protection or confidentiality protection. When communication uses only pairwise channels among a group of potentially faulty or malicious parties, however, not all parties necessarily observe the same messages. In this situation, merely reaching agreement on a single bit among the non-faulty parties represents already the difficult problem of Byzantine agreement [106]. The difficulty is caused by potentially faulty parties who might send conflicting messages over the pairwise channels.

A broadcast channel in the synchronous model (see below) is an abstraction which guarantees that all non-faulty parties receive the same set of messages through the channel and that they receive them in the same order. It can be implemented using a protocol for Byzantine agreement.

Another distinction is the assumption of synchronized clocks. In a synchronous network, there are known upper bounds on the delay of messages on any channel and on the relative clock speeds of the parties. For simplicity and without loss of generality, one can imagine that all communication in synchronous network occurs over in a sequence of global rounds using either the point-to-point channels or the broadcast channel. In every round, every party sends a message on all point-to-point channels and at the end of the round, every non-faulty party receives a message on all point-to-point channels or a special symbol \( \perp \) indicating that no message was sent. Similarly for the broadcast channel, in every round one distinguished party sends a message on the broadcast channel (or is at least supposed to send one), and all parties receive that message or a special symbol \( \perp \) indicating that no message was sent. The sequence of senders is predefined.

Many cryptographic protocols for multiparty computation assume this idealized model. In an asynchronous network, no common clock exists and the delay of all messages on the point-to-point channels is unbounded. Typically an adversarial scheduler is assumed to control the delivery of messages over the network. The counterpart of a synchronous broadcast channel is an asynchronous atomic broadcast facility. Implementing asynchronous atomic
broadcast and asynchronous Byzantine agreement requires randomized protocols and all such protocols are subject to a negligible probability of failure. This is because of the “FLP” impossibility result which shows that all deterministic agreement protocols in asynchronous networks with even a single faulty party have non-terminating executions [65].

3.5 Trusted Computing Base

A trusted computing base (TCB) is a part of a platform that is assumed to perform its tasks correctly, even if the platform itself is corrupted. The assumption that a platform is equipped with a TCB can greatly improve the efficiency of multiparty protocols, or allow otherwise unsolvable problems to be addressed. The literature knows a variety of models for TCBs ranging from a fully capable computer with its own communication network down to a secure signing device. The most practical implementations are the IBM 4758 Cryptoprocessor (essentially a separate, tamperproof computer) and the Trusted Computing Groups’ TPM (Trusted Platform Module), which provides little functionality itself, but whose “remote attestation” ability allows to build a larger software TCB on top of the tamperproof hardware.
Chapter 4

State-of-the-Art and Open Problems

We now discuss the current state-of-the-art for several popular two-party protocols and stress the most important open problems and future directions.

4.1 Zero Knowledge Proof Systems

The classical notion of a zero-knowledge proof has been introduced in [72]. Roughly speaking, in a zero-knowledge proof a prover can prove to a verifier the validity of a statement without releasing any additional information. In order to prove that a protocol does not leak information it is required to show the existence of a probabilistic polynomial-time algorithm, referred to as simulator, whose output is indistinguishable from the output of the interaction between the prover and the verifier. Since its introduction, the concept of a zero-knowledge proof system and the simulation paradigm have been widely used to prove the security of many protocols.

**Notation.** Let $A$ and $B$ be two interactive algorithms, we define by $\langle A, B \rangle(x)$ as the local output of $B$ after an interactive execution with $A$.

**Definition 4.1.1** Let $L$ be an NP-language, $P, V$ be probabilistic polynomial-time algorithms. We say that $\Pi = (P, V)$ is a zero-knowledge proof system if:

- **completeness:** for every $x \in L$ where $|x| = \text{POLY}(k)$, all valid NP witnesses $w$ there exists a negligible function $\nu$ such that
  $$\text{Prob}(\langle P(w), V \rangle(x) = 0) \leq \nu(k);$$

- **soundness:** for any unbounded algorithm $P^*$ there exists a negligible function $\nu$ such that for any $x \notin L$,
  $$\text{Prob}(\langle P^*, V \rangle(x) = 1) \leq \nu(k);$$

- **zero knowledge:** for any polynomial-time adversarial verifier $V^*$, there exists a probabilistic polynomial-time algorithm $S$ such that for any auxiliary input $\text{aux}$, the following two distributions are computationally indistinguishable:
  $$\{(P, V^*(\text{aux}))(x)\}_{x \in L} \quad \{S(\text{aux})\}_{x \in L}$$
The soundness requirement can be relaxed by considering polynomial-time adversaries $P^*$ only, thus obtaining argument systems. The zero-knowledge requirement can be restricted to the black-box notion if there exists one simulator $S$ that works by black-box accessing any adversarial verifier $V^*$.

A different notion of soundness has been recently given in [76].

**Definition 4.1.2 (Co-soundness [76])** A pair of interactive Turing machines $\langle P, V \rangle$ satisfies the co-soundness property for the language $L$ if the following holds. Let $R^\omega_L$ be a binary relation consisting of theorems $x$ and witnesses $w^\omega$ for $x \notin L$. Then, for every $x \notin L$ and for every interactive Turing machines $P^*$ there exists a negligible function $\nu(\cdot)$ such that

$$\text{Prob} \left[ (w^\omega, 1) \leftarrow \langle P^*, V \rangle(x) \text{ and } (x, w^\omega) \in R^\omega_L \right] < \nu(|x|).$$

If a pair of interactive Turing machines $\langle P, V \rangle$ satisfies the completeness and the co-soundness properties then $\langle P, V \rangle$ is called a co-sound interactive proof system. If the co-soundness condition holds only with respect to probabilistic polynomial-time interactive Turing machines $P^*$ then a complete pair $\langle P, V \rangle$ is called a co-sound argument.

**Concurrent and resettable zero knowledge and soundness.** Motivated by considerations regarding the existence of asynchronous networks as the Internet and of specific devices as smart cards, the notions of concurrent and resettable zero knowledge (cZK, rZK, for short) have been introduced in [60, 28]. In a cZK proof system the verifier can open any polynomial number of concurrent sessions with a prover. In a rZK proof system the verifier is allowed to tamper with the prover and to reset the prover in the middle of a proof to any previous state, then asking different questions. It is easy to see that concurrent zero knowledge is a special case of resettable zero knowledge and, currently, rZK is the strongest notion of zero knowledge that has been studied when security against malicious verifiers is considered.

When the prover can instead rewind the verifier then a notion of resettable soundness should be considered. This notion has been investigated and achieved in [10] where the open problem of simultaneous resettability has been given. This problem has been partially solved in [51] where resettably-sound resettable zero knowledge has been achieved under some restrictions.

**Non-malleable zero knowledge.** Non-malleable zero knowledge deals with an adversary (called man-in-the-middle) that simultaneously participates to two executions of a proof systems and acts as a prover in one and as a verifier in the other. The adversary has complete control over the scheduling of the messages in the two executions of the protocol and, while acting as a verifier, can request to see the proof of any theorem of his choice. Informally, a zero-knowledge proof system is said non-malleable if the information that a man-in-the-middle adversary collects as a verifier in one interaction does not help him to prove another statement in the other interaction.

**Simulation-sound zero knowledge.** Simulation-sound zero knowledge has been introduced in [118] for the purpose of constructing cryptosystems secure against adaptive chosen-ciphertext attacks. This concept is related to the concept of non-malleability introduced in [58]. Indeed, both notions deal with an adversary (called the man-in-the-middle) that
simultaneously participates in many executions of two proof systems and acts as a prover in the former and as a verifier in the latter. The adversary has complete control over the scheduling of the messages in the executions of the protocols. Informally, two zero-knowledge proof systems are said mutually concurrent simulation sound if the information that the man-in-the-middle adversary collects as a verifier from concurrent sessions played with a simulated prover of the former proof system does not help him to prove a false statement in the latter proof system and vice versa. Here the man-in-the-middle can choose to see simulated proofs of true and false statements.

Simulation-sound zero knowledge plays an important role for proving the security of protocols. Indeed, when the simulation paradigm is used to prove the security of a protocol, the simulator could, in some cases, need to simulate the proof of a false statement. Here simulation soundness is crucial since the adversary could gain knowledge from such a proof in order to prove a false statement in another protocol.

**State-of-the-art.** Unfortunately, in the plain model, constant-round black-box concurrent zero-knowledge is impossible for non-trivial languages [29] while both constant-round black-box zero knowledge and even constant-round non-malleable zero-knowledge and non-malleable commitments are possible [70, 7]. A concurrent zero-knowledge protocol with logarithmic round complexity has been presented in [114]. By using non-black-box techniques, Barak achieved in [6] a constant-round bounded-concurrent zero-knowledge argument system and in [7] a constant-round non-malleable zero-knowledge argument system.

**Open Problem No. 4:** A very important open question is therefore the existence of a constant-round (or even $O(\alpha(n))$-round for all non-constant functions $\alpha$) concurrent zero knowledge proof or argument system in the plain model (this requires non-black-box techniques) or in new models.

Very partially this open problem has been solved in [107] where (almost) constant-round concurrent zero-knowledge argument systems are achieved in the single-prover model;

The impossibility of constant-round black-box concurrent zero-knowledge for non-trivial languages [29] in the plain model uses the fact that $V^*$ can abort its interactions with the provers. Instead, in [117], Rosen showed that without aborts there still exists a constant-round lower bound but the possibility of obtaining a constant round protocol (with a larger constant) is still an open problem.

In the bare public-key model constant-round concurrent and resettable zero-knowledge arguments for all $NP$ were shown to exist in [28]. Micali and Reyzin [90] noticed that, unlike in the standard model for interactive zero knowledge, distinct notions of soundness arise depending on whether the verifier’s public key is used for once (one-time soundness), for polynomially many sequential arguments (sequential soundness), for polynomially many concurrently interleaved arguments (concurrent soundness), or whether the prover is allowed to reset the verifier to a given state during the interaction (resettable soundness). However, they showed that resettably sound zero knowledge cannot be achieved in the black-box model for non-trivial languages. Consequently, for black-box zero knowledge, the strongest possible notion is that of a concurrently sound resettable zero-knowledge argument. In [90], Micali and Reyzin showed that in the BPK model, concurrent soundness cannot be achieved in less than four rounds. Moreover they showed that the argument system of Canetti et
al. presented in [28] is only sequentially sound and the same holds for the four-round resettable zero-knowledge argument presented in [90]. Recently, the existence of a constant-round concurrently-sound resettable zero-knowledge argument in the BPK model has been proved by [54] where a 4-round (optimal) concurrently-sound resettable zero-knowledge argument in the BPK model has been given for all \( \mathcal{NP} \) languages. When stateful verifiers are considered, then 3-round resettable zero-knowledge is possible as recently shown in [55]. Very recently, other results on concurrent zero-knowledge with sequential and concurrent soundness in the bare public-key model have been presented respectively in [1] and in [56].

We give now the definitions of zero-knowledge and soundness in the BPK model.

**The BPK model.** The Bare Public-Key (BPK, in short) model assumes that:

1. there exists a public file \( F \) that is a collection of records, each containing a public key;
2. an (honest) prover is an interactive deterministic polynomial-time algorithm that takes as input a security parameter \( 1^n \), \( F \), an \( n \)-bit string \( x \), such that \( x \in L \) and \( L \) is an \( \mathcal{NP} \)-language, an auxiliary input \( y \), a reference to an entry of \( F \) and a random tape;
3. an (honest) verifier \( V \) is an interactive deterministic polynomial-time algorithm that works in the following two stages: 1) in a first stage on input a security parameter \( 1^n \) and a random tape, \( V \) generates a key pair \((pk, sk)\) and stores \( pk \) in one entry of the file \( F \); 2) in the second stage, \( V \) takes as input \( sk \), a statement \( x \in L \) and a random string, \( V \) performs an interactive protocol with a prover, and outputs "accept" or "reject";
4. the first interaction of each prover starts after that all verifiers have completed their first stage.

**Definition 4.1.3** Given an \( \mathcal{NP} \)-language \( L \) and its corresponding relation \( R_L \), we say that a pair \( \langle P, V \rangle \) is complete for \( L \), if for all \( n \)-bit strings \( x \in L \) and any witness \( y \) such that \((x, y) \in R_L\), the probability that \( V \) interacting with \( P \) on input \( y \), outputs "reject" is negligible in \( n \).

**Malicious provers in the BPK model.** Let \( s \) be a positive polynomial and \( P^* \) be a probabilistic polynomial-time algorithm that takes as first input \( 1^n \).

\( P^* \) is an \( s \)-sequential malicious prover if it runs in at most \( s(n) \) stages in the following way: in stage 1, \( P^* \) receives a public key \( pk \) and outputs an \( n \)-bit string \( x_1 \). In every even stage, \( P^* \) starts from the final configuration of the previous stage, sends and receives messages of a single interactive protocol on input \( pk \) and can decide to abort the stage in any moment and to start the next one. In every odd stage \( i > 1 \), \( P^* \) starts from the final configuration of the previous stage and outputs an \( n \)-bit string \( x_i \).

\( P^* \) is an \( s \)-concurrent malicious prover if on input a public key \( pk \) of \( V \), can perform the following \( s(n) \) interactive protocols with \( V \): 1) if \( P^* \) is already running \( i \) protocols \( 0 \leq i < s(n) \) he can start a new protocol with \( V \) choosing the new statement to be proved; 2) he can output a message for any running protocol, receive immediately the response from \( V \) and continue.

**Attacks in the BPK model.** Given an \( s \)-sequential malicious prover \( P^* \) and an honest verifier \( V \), a sequential attack is performed in the following way: 1) the first stage of \( V \) is run on input \( 1^n \) and a random string so that a pair \((pk, sk)\) is obtained; 2) the first stage of \( P^* \)
is run on input 1^n and pk and x_1 is obtained; 3) for 1 \leq i \leq s(n)/2 the 2i-th stage of P^* is run letting it interact with V that receives as input sk, x_i and a random string r_i, while the (2i + 1)-th stage of P^* is run to obtain x_i.

Given an s-concurrent malicious prover P^* and an honest verifier V, a concurrent attack is performed in the following way: 1) the first stage of V is run on input 1^n and a random string so that a pair (pk, sk) is obtained; 2) P^* is run on input 1^n and pk; 3) whenever P^* starts a new protocol choosing a statement, V is run on inputs the new statement, a new random string and sk.

Definition 4.1.4 Given a complete pair \langle P, V \rangle for an NP-language L in the BPK model, then \langle P, V \rangle is a concurrently (resp., sequentially) sound interactive argument system for L if for all positive polynomial s, for all s-concurrent (resp., s-sequential) malicious prover P^*, for any false statement “x \in L” the probability that in an execution of a concurrent (resp., sequential) attack V outputs “accept” for such a statement is negligible in n.

We now give the formal definition of a black-box resettable zero-knowledge argument system for \mathcal{NP} in the bare public-key model.

Definition 4.1.5 An interactive argument system \langle P, V \rangle in the BPK model is black-box resettable zero-knowledge if there exists a probabilistic polynomial-time algorithm S such that for any probabilistic polynomial time V^*, for any polynomials s, t, for any x_i \in L, |x_i| = n, i = 1, \ldots, s(n), V^* runs in at most t steps and the following two distributions are indistinguishable:

1. the output of V^* that generates F with s(n) entries and interacts (even concurrently) a polynomial number of times with each P(x_i, y_i, j, r_k, F) where y_i is a witness for x_i \in L, |x_i| = n and r_k is a random tape for 1 \leq i, j, k \leq s(n);

2. the output of S interacting with V^* on input x_1, \ldots, x_{s(n)}.

Moreover we define such an adversarial verifier V^* as an \langle s, t \rangle-resetting malicious verifier.

Open Problem No. 5: Is it possible to construct in the bare public-key model protocols that achieve notions of security that are stronger than concurrent and resettable zero knowledge?

Solved Problem No. 1 (solves in part Open Problem No. 5): In [98], it has been showed that in the bare public-key model there exists a constant-round concurrent non-malleable zero-knowledge argument of knowledge.

Open Problem No. 6: Is it possible to construct in the bare public-key model more practical and secure protocols?

Solved Problem No. 2 (solves in part Open Problem No. 6): In [122], it has been showed that in the bare public-key model there exists an efficient constant-round concurrent zero-knowledge argument of knowledge for any language admitting an efficient \Sigma-protocol.
Solved Problem No. 3 (solves in part Open Problem No. 6): In [125], it has been showed that in the bare public-key model there exists an efficient constant-round resettable zero-knowledge argument system for any language admitting an efficient $\Sigma$-protocol. Moreover they showed that there exist constant-round resettable zero-knowledge argument systems under minimal complexity-theoretic assumptions.

The definition of zero-knowledge in the shared random string model is different. We now show the definition for non-interactive zero knowledge since this was the first notion of zero knowledge that has been explored in this model.

Non-malleable zero knowledge. The study of non-malleable zero-knowledge proof systems has been initiated by Dolev, Dwork and Naor [58] that gave formal definitions and a $O(\log n)$-round non-malleable zero-knowledge proof system. More recently, Barak [7] gave a constant-round non-malleable zero-knowledge argument using non-black box techniques. It is not know whether non-black box techniques are necessary for constant-round non-malleable zero-knowledge. Moreover the known constructions for non-malleable zero-knowledge in the plain model do not preserve non-malleability in case the man-in-the-middle can concurrently play a polynomial number of sessions as both verifier and prover. A protocol that is secure in this model is referred to as concurrent non-malleable.

In the shared-random string model (in which concurrent compositions comes for free), concurrent non-malleable non-interactive zero-knowledge proof systems have been given by Sahai [118] for the single-theorem case and by De Santis et al. [50] for the multi-theorem case. By using interaction, some efficient constructions have been recently given in [69, 87].

In the reference string model, efficient constructions of universally composable zero-knowledge arguments have been achieved in [46, 44, 68, 87].

Recently, in [34], Catalano and Visconti presented 3-round concurrent zero-knowledge proof systems, in the shared random string model, for all $NP$ languages. They gave a construction based on the existence of any one-way function and an efficient construction that is based on the DDH assumption. These results improve the computational soundness achieved in a previous result by Damgård [42] in the sense that theirs are actually zero-knowledge proofs rather than zero-knowledge arguments. Moreover, they showed how to construct an unbounded simulation-sound zero-knowledge proof system in the common reference string model.

This improves the recent results of [87, 69] where similar results were presented for unbounded simulation-sound zero-knowledge arguments (rather than proofs).

In the common reference string (resp., random oracle) model, since the simulator works by generating a new common reference string (resp., by simulating a random oracle), a zero-knowledge protocol is not deniable [102], this is in contrast with respect to the plain model where deniability is enjoyed by zero-knowledge protocols.

Many impossibility results of the plain model do not hold in the previously discussed augmented models.

Open Problem No. 7: It is interesting to find weak set-up assumptions that allow to overcome the impossibility results of the plain model.
Solved Problem No. 4 (solves in part Open Problem no. 7): In [82], Katz showed that it is possible to achieve universally composable multi-party computation using tamper-proof hardware.

4.2 Commitment Schemes

Commitment schemes are arguably among the most important and useful primitives in cryptography. Intuitively a commitment scheme can be seen as the digital equivalent of a sealed envelope. If a party $A$ wants to commit to some message $m$ she just puts it into the sealed envelope, so that whenever $A$ wants to reveal the message, she opens the envelope. Clearly, such a mechanism can be useful only if it meets some basic requirements. First of all the digital envelope should hide the message: no party other than $A$ should be able to learn $m$ from the commitment (this is often referred in the literature as the hiding property). Second, the digital envelope should be binding, meaning with this that $A$ can not change her mind about $m$, and by checking the opening of the commitment one can verify that the obtained value is actually the one $A$ had in mind originally (this is often referred to as the binding property). These two properties make commitments very useful in a wide range of cryptographic applications such as zero-knowledge protocols, multi-party computation, digital auctions and electronic commerce.

A commitment scheme is a primitive to generate and open commitments. More precisely a commitment scheme is a two-phase protocol between two probabilistic polynomial time algorithms sender and receiver. In a first stage (called the commitment phase) sender commits to a bit $b$ using some appropriate commitment function which takes as input $b$ and some auxiliary value $r$ and produces as output a value $y$. The value $y$ is sent to receiver as a commitment on $b$. In the second stage (called the decommitment phase) sender “convinces” receiver that $y$ is actually a valid commitment on $b$ (if receiver is not convinced, it just outputs some special string). The requirements that we make on a commitment scheme are the following ones. First, if both sender and receiver behave honestly, then at the end of the decommitment phase receiver is convinced that sender had committed to bit $b$ with probability 1. This is often referred to as the correctness requirement. Second a cheating receiver can not guess $b$ with probability significantly better than $1/2$. This is the so-called hiding property. Finally, a cheating sender should be able to open a commitment (i.e., to decommit) with both $b$ and $1 - b$ only with very small (i.e., negligible) probability (this is the binding property). Each of the last two properties (i.e., hiding and binding) can be satisfied unconditionally or relatively to a computational assumption. In our context (i.e., where only two parties are involved) this immediately implies that one can not hope to build a commitment scheme where both the hiding and the binding properties hold unconditionally. Unconditionally binding commitment schemes have been constructed under the sole assumption that one-way functions exist [93] and in such a construction an initial message of the receiver is required. It is known how to construct non-interactive unconditionally binding commitment schemes by using any one-to-one one-way function [16]. Constant-round unconditionally hiding commitment schemes have been constructed under the assumption that collections of claw-free functions [70] or collision resistant hash functions [78] exist. Recently in [77] it is shown how to construct unconditionally hiding commitment schemes from any regular one-way function, but unfortunately, these schemes are not constant round.

Since commitment schemes are very useful primitives they are often used as building blocks
to construct larger protocols. In this sense it is often the case that the two basic requirements described above turn out to be insufficient. For this reason commitment schemes with additional properties have been proposed. Here we highlight on some of these constructions, other, more directly related to our results, will be discussed in the next section.

A trapdoor commitment scheme (sometimes also called chameleon commitment), is a commitment scheme with associated a pair of public and private keys (the latter also called the trapdoor). Knowledge of the trapdoor allows the sender to open the commitment in more than one way (this is often referred to as the equivocality property). On the other hand, without knowledge of the trapdoor, equivocality remains computationally infeasible. When the commitments computed by means of a trapdoor are distributed exactly as real commitments then the trapdoor commitment scheme is unconditionally hiding. Trapdoor commitments have been shown to exist under the assumption that one-way functions exist [63].

A commitment scheme is said non-malleable if – very informally – given a commitment $y$ on some message $m$, knowledge of $y$ does not help another party in constructing a new commitment $y'$ of a message $m'$ related to $m$. Such a property is referred to as non-malleability with respect to commitment and is studied for unconditionally binding commitment schemes. Another different property is non-malleability with respect to opening. In this case, $y'$ is a commitment that once $m$ is revealed, can be opened to a value $m'$ related to $m$. Non-malleable commitments with respect to commitment have been defined in [58], and the first constant-round construction in the plain model has been given in [7] for the stand-alone case (i.e., the protocol is interactive and has to be executed in isolation, concurrency is not allowed). Non-interactive non-malleable commitments with respect to opening have been originally constructed in [52, 66, 53]. Such constructions have been recently improved in [44] where once the commitment parameters have been established (by a trusted third party), it is possible to compute any polynomial number of non-malleable commitments.

An extractable commitment (also known as commitment scheme with extractability) is a commitment scheme where we allow the existence of a secret key whose knowledge permits to extract the message stored in the commitment. At the same time, without knowledge of the secret key, the message remains (computationally) hidden.

Finally a universally composable commitment is a commitment scheme with the very useful property that – informally – even if one concurrently composes it with any other protocol, the security of the commitment scheme is preserved. Universal composability is a very strong notion, which, for the case of commitment schemes, seems to require concurrent non-malleability and extractability.

The first construction of a universally composable commitment scheme has been presented in [27] and it has been later improved in [47] and in [44].

4.2.1 Other Notions of Commitments

Simulation-sound trapdoor commitments. Garay et al. [68] introduced the notion of simulation-sound trapdoor commitments (SSTCs, for short) which was later relaxed by MacKenzie and Yang in [87]. In a nutshell a SSTC scheme is a trapdoor commitment scheme with the additional property that an adversary can not equivocate a commitment with a certain tag, even after seeing a polynomial number of equivocations for commitments with different tags.

In [68] SSTCs are used to construct efficient universally composable zero-knowledge arguments that remain secure even when facing adaptive adversaries (i.e., adversaries which
are allowed to adaptively corrupt parties involved in the protocol). In [87] it is presented a simpler - and slightly weaker - definition of SSTC and the authors prove that the resulting primitive is actually equivalent to secure signatures [73]. The notion of SSTC introduced in [87], being weaker with respect to the one in [68] allows one to construct more efficient number-theoretic implementations. Interestingly, this weaker notion still remains sufficient to construct many applications (such as simulation-sound zero-knowledge argument systems).

**Multi-trapdoor commitments.** The notion of a multi-trapdoor commitment scheme was introduced by Gennaro [69]. A multi-trapdoor commitment scheme is a family of trapdoor commitments such that each scheme in the family is a trapdoor commitment scheme. The main feature of multi-trapdoor commitments is that they admit a master trapdoor whose knowledge allows one to equivocate for any commitment scheme in the family. Furthermore every commitment in the family admits also its own local trapdoor. However knowledge of the local trapdoor for a given scheme does not allow to equivocate on another scheme of the family (unless of course the master trapdoor is available). As pointed out in [69] multi-trapdoor commitments are related, but weaker than simulation-sound trapdoor commitments, this, once again, leads to more efficient number theoretic instantiations. In particular, in [69], Gennaro presents two very efficient constructions (based on the Strong RSA and on the strong Diffie-Hellman assumptions respectively) as well as several applications. The main application of multi-trapdoor commitments is the construction of left-concurrent non-malleable arguments of knowledge. Here the left-concurrent non-malleable setting is a model in which the adversary can play in a polynomial number of sessions the role of verifier while in one session he plays the role of prover. As already pointed out in [87], a left-concurrent non-malleable argument of knowledge is actually an unbounded simulation-sound zero-knowledge argument system. We stress that the multi-trapdoor commitments presented in [69] allows for simulation-sound zero-knowledge argument systems that are more efficient than the ones presented in [87]. The definition of multi-trapdoor commitments given in [69] requires unconditional hiding. This property is obviously enjoyed by both implementations of multi-trapdoor commitments given in [69]. However, in the applications given in [69], unconditional hiding is not used. Since even our applications will only use computational hiding, we will focus in both the definitions and the constructions on multi-trapdoor commitments that are computationally hiding.

We finally notice that the problem of establishing the minimal assumptions required to construct multi-trapdoor commitments was left open in [69].

**Hybrid, mixed and mercurial commitments.** By playing with the equivocality property of trapdoor commitment schemes, the two following notions of commitment schemes have been recently proposed, namely hybrid trapdoor commitments and mercurial commitments.

In [34, 35] Catalano and Visconti presented the notion of hybrid trapdoor commitment schemes. Informally an hybrid trapdoor commitment scheme is a general commitment primitive that allows for two commitment parameters generation algorithms. If the commitment parameters are obtained as the output of the first algorithm then the resulting scheme is an unconditionally binding commitment scheme, while if the parameters are generated by the second algorithm then the produced scheme is actually a trapdoor commitment scheme. Moreover, the output of any polynomially bounded adversary has the same distribution when it works with the parameters generated by the two algorithms. In [34] the authors show that non-interactive hybrid trapdoor commitments can be constructed from any one-way function
In the shared random string model. Moreover, they show that very efficient constructions under standard number-theoretic assumptions.

In [47] Damgård and Nielsen introduced the notion of mixed commitments. Informally, a mixed commitment scheme can be either a trapdoor commitment scheme or an extractable commitment scheme, the exact nature depends on the distribution according to which the public key is generated. On some of the keys (E-keys) the scheme is unconditionally hiding and equivocal, while on some others (X-keys) the scheme is unconditionally binding and extractable. Of course no key can be both an E-key and an X-key. A crucial property of mixed commitment schemes is that no polynomially bounded adversary (not having access to secret keys or trapdoors) should be able to distinguish E-keys from X-keys.

In [47] some efficient implementations have been derived from Damgård-Jurik’s [45] variant of Paillier [101] cryptosystem and from Okamoto-Uchiyama [97] cryptosystem. Moreover the authors showed how to use mixed commitments to build universally composable commitments with the constraint that the size of the public key depends on the number of players. More recently Damgård and Groth [44] improved this construction by proposing a universally composable commitment scheme based on the strong RSA assumption, that can work with a reference string whose size does not depend from the number of players involved in the protocol. This result is achieved by combining a non-malleable commitment scheme with mixed commitments.

In [36] Chase et al. considered two different ways for computing and opening commitments introducing the notion of mercurial commitment schemes. In such schemes, the sender is allowed to compute hard and soft commitments. An hard commitment is a classical unconditionally binding commitment. A soft commitment, on the other hand, can be teased (i.e., partially open) to any value by the sender, but can not be (fully) opened. In this sense, soft commitments are quite different than trapdoor commitments as they can be teased to any value but can not actually be opened to any of them. The sender can also tease an hard commitment as the same value that he can open. An important property of mercurial commitment schemes is that, by looking at a commitment, it is computationally infeasible to decide whether it is an hard or a soft commitment. More precisely, a mercurial commitment is secure if there exists a simulator that can produce commitments that it can later open or tease to any value and whose distribution remains indistinguishable with respect to the distribution of the commitments produced by the legitimate sender. Non-interactive mercurial commitments have been constructed, in the shared random string model, under the assumption that non-interactive zero-knowledge proof systems exist [36]. Mercurial commitments can be used to construct zero-knowledge sets by only adding the assumption that collision-resistant hash functions exist (see below).

By looking at the properties of mercurial commitments it may seem that they are actually a more powerful primitive than hybrid commitments. This intuition may lead to explain the current gap between the complexity-based assumptions used to construct non-interactive mercurial commitments (i.e., NIZK proofs) and non-interactive hybrid trapdoor commitments (i.e., one-way functions) in the shared random string model. In this paper, we show that this intuition is wrong by showing that non-interactive hybrid trapdoor commitments suffice for constructing non-interactive mercurial commitments in the shared random string model.

Recently, Dodis et al. [33] provide simple constructions of mercurial commitments from any trapdoor bit commitment scheme. Moreover, by plugging in various trapdoor bit commitment schemes, they get, in the common reference string model, all the efficient constructions from [89] and [36], as well as several immediate new (either generic or efficient) constructions. In
particular, they get a construction of mercurial commitments from any one-way function in
the common reference string model, and, by using hybrid trapdoor commitments, even in the
(weaker) shared random string (SRS) model. Their results imply that mercurial commitments
can be viewed as variations of trapdoor commitments.

**Non-black-box commitments.** Non-black-box techniques has been recently used for con-
structing non-malleable commitment schemes in the plain model, originally achieved with
black-box techniques in [58]. The first work in this direction is the one of [7, 8] where
constant-round statistically-binding non-malleable commitments have been constructed by
assuming the existence of trapdoor permutations. This result has been later improved by [105]
where both constant-round statistically-binding and statistically-hiding non-malleable com-
mitments have been constructed on top of the existence of families of collision-resistant hash
functions. Very recently, constant-round statistically-binding concurrent non-malleable com-
mitments have been constructed by [104]. Two main notions for non-malleable commitments
have been considered so far. The former notion is non-malleability with respect to commit-
ment and focuses on the fact that committed messages should be independent and thus this
independency should be preserved when messages are opened. Instead the latter one is non-
malleability with respect to opening and focuses on the fact that the adversary should not be
able to open related messages. Under the original notions, non-malleability with respect to
commitment implies non-malleability with respect to opening. Interestingly, in [105, 104] a
stronger notion of non-malleability is presented and under this notion non-malleability with
respect to commitment does not imply non-malleability with respect to opening. Moreover,
the protocol of [104] does not achieve concurrent non-malleability with respect to opening
under the stronger notion. Recently, in [99] it has been shown how to achieve constant-round
concurrent non-malleable commitment with respect to both commitment and opening under
this stronger notion and assuming that commitments and openings are performed in different
non-overlapping stages.

**Open Problem No. 8:** An important open question is whether or not statistically-
hiding concurrent non-malleable commitment schemes exist in the plain model.

### 4.3 Zero Knowledge Sets

In [89], Micali *et al.* introduced the concept of a zero-knowledge set (ZKS, for short). There a
prover $P$ commits to an arbitrary set $S$ so that for any string $x$ he can later prove to a verifier
$V$ that $x \in S$ or $x \notin S$. Such a proof is required to be both sound and zero knowledge. The
former requirement preserves the security for the verifier since he can not be convinced of a
false proof given by an adversarial prover. The latter requirement preserves the security for
the prover since no adversarial verifier can learn more information than the mere truthfulness
of the proved statements.

A light variation (actually extension) of zero-knowledge sets is that of zero-knowledge
elementary databases where $x$ is considered a key and $v(x)$ is the corresponding datum. In
this case, for any key $x$ the prover either proves that $x \notin S$ or proves that $x \in S \land v(x) = u$,
still preserving soundness and zero knowledge. We will focus on zero-knowledge sets but all
the discussions and results extend also to zero-knowledge elementary databases.

In [89] Micali *et al.*, presented a construction of a ZKS based on the following properties.
1. Non-interactiveness. In the proposed construction all messages are monodirectional from the prover to the verifier. Therefore their construction is round-optimal since each commitment of a set and each proof needs only one round.

2. Shared random string (SRS, for short) model. It is assumed that all parties have access to a shared source of random bits. This assumption is weaker than the so called common reference string (CRS, for short) model, where the shared string can have a particular distribution.

3. Discrete logarithm assumption (DLA, for short). The construction is proved secure under the number-theoretic assumption that computing discrete logarithms in special groups is hard.

In the conclusions of their work, Micali et al., left open the problem of constructing zero-knowledge sets under more general assumptions than the DLA. This open problem has been recently studied by Ostrovsky et al. in [100] in the context of consistent database queries. In [100] two main results were obtained: first they show that collision-resistant hash functions are sufficient for interactive ZKSs; then they prove that non-interactive ZKSs can be constructed in the shared random string model by using collision-resistant hash functions (CRHF's, for short) and non-interactive zero-knowledge (NIZK, for short) proof systems.

Mercurial commitments for ZKSs. Alternative proofs of the same results of [100] discussed above, are given in [36] by Chase et al. where the concept of a mercurial commitment is introduced along with their application for the construction of zero-knowledge sets. More specifically, zero-knowledge sets are constructed by using CRHFs and mercurial commitments. Since they construct interactive mercurial commitments by only using one-way functions and non-interactive mercurial commitments in the SRS model by only using NIZK proof systems, they achieve the same feasibility results of [100]. Moreover, in [36] other constructions of mercurial commitments are given by using different assumptions, for instance based on claw-free permutations or factoring in the common reference string (CRS, for short) model, and based on the discrete logarithm problem in the SRS model.

On top of the recent results by Dodis et al. [33] we have that the existence of non-interactive zero-knowledge sets is equivalent to the existence of collision-resistant hash functions.

Updatable zero-knowledge sets. In a recent paper [86], Liskov shows how to extend the zero-knowledge set construction based on mercurial commitments [36] by allowing efficient updates. This is a very challenging area since it is strictly connected to the notion of secure database.

Open Problem No. 9: A very challenging question is the possibility of constructing secure databases that efficiently support the complex operations of current (insecure) databases.

4.4 Implications on Practical Protocols

Browser-based services receive increasing attention for a variety of applications in practice: users send their requests for different services through a browser, which offers a set of basic functionalities, and receive the result of the desired computation transparently. In general the requested service could be offered by some service provider of affiliated enterprises.
In this context it is important that only registered and authorized users can obtain the desired services, and that the enterprises may simplify users management. Hence, identity federation is currently an area of particular interest. This technology is based upon linking a user’s (otherwise) distinct identities at several locations. Federated identity management systems allow individuals to use the same identification information to sign on to the networks of several enterprises in order to conduct transactions. The partners in such a system trust each other to authenticate their respective users and vouch for their access to services. The advantage is that companies can share applications without having to adopt the same back-end technologies for different services, security and authentication, and reduce user management costs (such the cost of password helpdesks and user registration and deletion.)

The most important class of identity federation protocols relies only on a standard web browser. These protocols are called browser-based (or zero-footprint) and particularly important for multi-party authentication and attribute exchange. The main reason behind this is usability and cost-effectiveness aspects: when using browser-based protocols one does not require the installation of special client software enabling a cost-efficient entry point into identity federation.

Examples of protocol frameworks that support browser-based federated identity management are Microsoft’s Passport [39] and its successor Cardspace [2], the Security Assertion Markup Language (SAML) [121], the Liberty Alliance project [115], the Shibboleth project for university identity federation [32], or WS-Federation [81], whereby SAML is an open protocol standard and basis for Liberty and Shibboleth.\(^1\)

Now, in the broader context the main security targets are to authenticate the user and to establish a secure channel. There is a number of well-studied solutions for achieving these goals. On the one hand, however, new protocols are designed and released by industry, where often these solutions are not considered due to various reasons (e.g., time-to-market issues). On the other hand, new applications and usability and business constraints associated with them set new requirements and demand for adaptation of existing schemes.

As long as the authenticity of the user is concerned we are mostly dealing with 3-party entity authentication. One of the first celebrated, and unfortunately broken, proposals was introduced by Needham and Schroeder [95]. There is a large body of literature on the tool-supported analysis of such protocols based on abstractions of cryptography, starting with [91]. As typical classical 3-party authentication protocols need no specific cryptographic techniques, it was not a favorite object of study in cryptography, and it seems that the first cryptographic proofs have been done for Needham-Schroeder-Lowe protocol [123, 3].

Establishment of a secure channel by a 3-party protocol is typically handled by the exchange of a session key and then some cryptography is used for authenticated encryption. This holds for practical protocol proposals such as Kerberos and public-key infrastructures as well as for cryptographic protocols. However, the channel-establishment technique of browser-based federated identity-management cannot be modularized as session key exchange followed by key usage because a standard browser would not use a key established in this way. Here one works with standard web constructs such as HTTP, SSL/TLS, i.e., a channel is established with unilateral authentication first (using SSL/TLS without client certificates) and to use additional information sent in this channel for third-party authentication of the so far anony-

\(^1\)It defines authentication and attribute tokens usable for identity federation, as well as basic profiles (protocols in typical security terms) for using these tokens. Several problems were recently found in a SAML profile [74].
mous user of this channel. Therefore, here one needs slightly different security requirements than usual, besides the novel need to model a browser and its user.\textsuperscript{2} Federated identity-management proposals typically treat such channels as a black-box submodule. Though a treatment of browser-based channels as a black-box is reasonable, it conceals the natural composition of the underlying SSL protocol for the 2-party setting. As with cryptographic protocols, the protocol includes a subroutine for the establishment of session keys and a subroutine that utilizes the session keys for the instantiation of secure channels. A composite protocol specification is common practice in the design of practical protocols.

**Open Problem No. 10:** An interesting question is whether the comprehensive theoretical security models of universal composition are appealing frameworks for the analysis of practical protocols. In particular, does the SSL protocol family UC-secure realizes ideal key exchange or secure channels functionalities? A positive answer to this question confirms the suitability of UC-like frameworks for the analysis of practical protocols and justifies their stronger security definitions for the merit of composition.

Although protocols for federated identity management are in use, their security is still unproven at least not justified publicly and there have been successful attacks. The common security analysis on browser-based identity federation protocols typically considers negative results, i.e., vulnerabilities of these schemes, and revised versions were released after the fact by introducing countermeasures. However, it is well-known that security proofs, but not vulnerability analysis, give the desired guarantee for the security. The first step in this direction was taken in \cite{75}.

Hence, the lack of precise protocol definitions and underlying formal models hamper a design for security and a faithful analysis. Formal models for defining protocols have been proposed in the literature (see e.g., \cite{108, 110, 21}) and applied to prove the security of cryptographic protocols (e.g., \cite{3}). The formal models in \cite{108, 110, 21} rely on the simulation paradigm.

In the context of browser-based protocols there are however multiple obstacles to overcome. Firstly, the protocols rely on a standard browser as one of the protocol participants. The browser is protocol-unaware, whereas all prior protocols analyzed cryptographically are assumed to be carried out by specific protocol machines that do nothing but executing the protocol as specified (unless the machine is corrupted). Browser-based protocols are aggregated protocols which sync cryptographic functionalities from multiple layers. It is a well-known fact, that two provably secure stand-alone do not imply that they are as secure as their composition. This is especially true for browser-based protocols. Many protocol flaws were exposed after taking into account the surrounding protocols. Secondly, due to the limited capabilities of browsers, the user at the browser is an active participant and certain assumptions must be made about the user as well. Recent studies point out however average-skilled Internet users understand neither server certificates nor browsers’ security indicators. Users are overstrained by the verification procedure and tend to ignore browser’s warnings; in fact, they evaluate web sites on the basis of non-technical indicators (e.g., brands, logos). This ceremony\textsuperscript{3} provides a wrong sense of security. In \cite{67}, a relaxed model for user authentication has been introduced.

\textsuperscript{2}Secure channels without mutual authentication were first treated in \cite{120}.

\textsuperscript{3}Carl Ellison coined at Crypto 2005 Rump Session the term ceremony to denote the paradigm that a provably secure cryptosystem becomes insecure when it is interfaced to a user. See \cite{62}.
based on the assumption of human-perceptible authenticator (HPA) indistinguishability. The user is disburdened from the necessity to verify cryptographic identities, but copes with easy-to-recognize identifiers as she is used to do in the physical world where identities are provided in an easily recognizable fashion. The authors introduced a technique called Browser-based Mutual Authentication based on the HPA-indistinguishability and firstly proved the security in the refined model for authentication due to Bellare and Rogaway [13].

**Open Problem No. 11:** Is it possible to relax the security assumption of human-perceptible authenticator indistinguishability and thus relax the security requirements of browser-based protocols within a formal model?

Thirdly, an alternative problem is the X.509 public key infrastructure for server authentication. Various Certification Authorities (CAs) have stored their root certificates in browsers and thus are trusted *per se*. Although CAs may considerably differ in their issuing policies browsers equally treat the certificates. Adversaries may exploit weak issuing policies and do a lot of harm.

**Open Problem No. 12:** Is it possible to design browser-based protocols that relax setup assumptions in a way that they do not prerequisite the existence of a trusted third party?

Finally, in addition to authentication and secure channel establishment, Federated identity protocols can also be analyzed for privacy targets. A detailed but informal treatment is given in [111]. The work in [112] describes a research proposal of browser-based federation protocol with optimal privacy using formal definitions to enable a more rigorous proof, however, no security analysis was made there based on these definitions.

The aim of the research work here is to build the theoretical fundament for the rigorous analysis and security proof of such browser-based protocols, in particular federated identity management protocols. The major building blocks for these protocols are to be precisely modeled based on a formal and abstract model for standard web browsers as well as browser channels and the semi-honest browsing behavior of a user as universally composable automatas.
Bibliography


