INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Survey of Symbolic Methods in Computational Analysis of Cryptographic Systems

Véronique Cortier — Steve Kremer — Bogdan Warinschi

N° 6912

Avril 2009

Thème SYM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6912--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

A Survey of Symbolic Methods in Computational Analysis of Cryptographic Systems

Véronique Cortier[*] , Steve Kremer[†] , Bogdan Warinschi[‡]

Thème SYM — Systèmes symboliques
Équipes-Projets Cassis et Secsi

Rapport de recherche n° 6912 — Avril 2009 — 42 pages

Abstract: Since the 1980s, two approaches have been developed for analyzing security protocols. One of the approaches relies on a computational model that considers issues of complexity and probability. This approach captures a strong notion of security, guaranteed against all probabilistic polynomial-time attacks. The other approach relies on a symbolic model of protocol executions in which cryptographic primitives are treated as black boxes. Since the seminal work of Dolev and Yao, it has been realized that this latter approach enables significantly simpler and often automated proofs. However, the guarantees that it offers have been quite unclear.

For more than twenty years the two approaches have coexisted but evolved mostly independently. Recently, significant research efforts attempt to develop paradigms for cryptographic systems analysis that combines the best of both worlds. There are two broad directions that have been followed. *Computational soundness* aims to establish sufficient conditions under which results obtained using symbolic models imply security under computational models. The *direct approach* aims to apply the principles and the techniques developed in the context of symbolic models directly to computational ones.

In this paper we survey existing results along both of these directions. Our goal is to provide a rather complete summary that could act as a quick reference for researchers who want to contribute to the field, want to make use of existing results, or just want to get a better picture of what results already exist.

Key-words: security protocols, formal methods, cryptography, abstraction, survey

This work has been supported by the ANR-07-SESU-002 AVOTÉ.

[*] Loria, CNRS & INRIA project Cassis, Nancy, France
[†] LSV, CNRS & ENS Cachan & INRIA project Secsi, France
[‡] Computer Science Department, University of Bristol, United Kingdom

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Un panorama sur les méthodes symboliques dans l'analyse calculatoire des systèmes cryptographiques

Résumé : Depuis les années 80, deux approches ont été développées pour l'analyse des protocoles de sécurité. L'une de ces approches repose sur un modèle calculatoire qui prend en compte les probabilités et la complexité algorithmique. Cette approche permet de définir une notion très forte de sécurité, garantissant contre n'importe quelles attaques probabilistes et polynomiales. L'autre approche repose sur une modélisation symbolique des exécutions du protocole, où les primitives cryptographiques sont traitées comme des boites noires. Depuis le travail précurseur de Dolev et Yao, il a été démontré que cette approche permet d'obtenir des preuves de sécurité beaucoup plus simples et souvent automatiques. Cependant, le niveau de garanties offert n'est pas clair.

Depuis plus de vingt ans, les deux approches ont évolué de manière plutôt indépendante. Des travaux de recherche récents cherchent à développer des techniques pour combiner le meilleur des deux approches. Deux directions principales ont été abordées. La *correction calculatoire* a pour but d'établir des conditions suffisantes pour que les résultats obtenus dans les modèles symboliques impliquent immédiatement des garanties dans les modèles calculatoires. L'*approche directe* cherche à appliquer directement aux modèles calculatoires les techniques développées dans le contexte des modèles symboliques.

Dans cet article, nous décrivons les résultats existants autour de ces deux axes. Notre but est de proposer un panorama le plus complet possible qui pourra servir à donner de manière synthétique un aperçu sur les résultats du domaine.

Mots-clés : protocoles de sécurité, méthodes formelles, cryptographie, abstraction, état de l'art

Contents

1 Introduction

Background. Security protocols are short distributed programs designed to achieve various security goals, such as data privacy and data authenticity, even when the communication between parties takes place over channels controlled by an attacker. Their ubiquitous presence in many important applications makes designing and establishing the security of cryptographic protocols a very important research goal. Two distinct approaches that have evolved starting with the early 1980's attempt to ground security analysis of protocols on firm, rigorous mathematical foundations. These two approaches are known as the computational (or the cryptographic) approach and the symbolic (or the Dolev-Yao, or the formal methods) approach. Each approach relies on mathematical models for the executions of protocols/primitives in adversarial environments, formally define security properties expected from cryptographic systems, and develop methods for rigorously proving that given constructions meet these requirements.

The central features of the computational approach are detailed, bit-level models for system executions and a powerful adversary: security is assessed against *arbitrary* probabilistic polynomial time machines. It is generally acknowledged that security proofs in this model offer powerful security guarantees. A serious downside of this approach however is that proofs for even moderately-sized protocols are usually long, difficult, tedious, and highly error prone.

In contrast, symbolic methods employ a highly abstract view of the execution where the messages exchanged by parties are symbolic terms. Furthermore, primitives are assumed absolutely secure, which in turn leads to severe restrictions on the power of the adversary. For instance, it is postulated the plaintext underlying a ciphertext can only be recovered if the adversary has or can derive the appropriate decryption key. The resulting models are considerably simpler than those of the computational approach, proofs are therefore also simpler, and can sometimes benefit from machine support. An important problem with this approach is that the high level of abstraction renders unclear the security guarantees that this approach offers.

A recent synergy. Due perhaps to the widely different set of tools and techniques, the two approaches have coexisted and developed independently for many years. The lack of interaction between the two communities also meant that the relation between models, security results and guarantees using the two approaches was only superficially understood. Abadi and Rogaway were the first to demonstrate that establishing close relations between the models is not only possible, but also that it holds significant promise. Through their work it became clear that it is possible to employ the tools and methods specific to the symbolic approach to directly obtain computational security guarantees. The crucial implication is that such guarantees can be obtained without making use of the typical computational proofs. This realization motivated a significant amount of follow-up work. We now know of several different approaches that leverage on technologies specific to the symbolic approach to simplify, avoid, or simply improve the rigorousness of computational proofs. The aim of this paper is to survey the plethora of papers that tackle this problem and briefly summarize their contribution. We hope that this survey will help researchers working in this field to get a better picture of all the different results. In addition, this survey should act as a fast reference for those researchers who want to enter the field or want to make use of existing results.

This survey. Existing results that span the gap between computational and symbolic security fall along two general directions. The first approach is known as the "computational soundness" approach. Here, the idea is to show that under certain conditions symbolic models are sound abstractions of cryptographic models, w.r.t. certain security properties. The second direction is called the "direct approach". In this line of research symbolic methods are applied directly to computational models. Although we survey both of these directions, we place more emphasis on computational soundness. This line of research is the "older" of the two, and had received significantly more attention. Next we describe the structure of our paper.

Computational soundness. Research on computational soundness was initiated by Abadi and Rogaway [AR00]. They considered the case of a passive adversary that eavesdrops on communication between honest parties. The basic question that they answer is under which conditions messages that are equivalent symbolically are also equivalent computationally. The setting they consider only uses symmetric encryption. Follow-up work treats variations of this questions with respect to different notions of symbolic equivalence, different sets of primitives, slightly more powerful adversaries, and within the context of particular applications. We describe the results for the *passive adversary* case in Section 2.

In Section 3 we survey results on computational soundness in the presence of *active adversaries*. These are adversaries who have absolute control over the communication network, and who may actively interfere with the execution of the protocol. There are two main approaches and a few variations and generalizations that we discuss. All of these are based on faithfulness results that show deep connections between computational and symbolic executions of protocols: essentially, assuming appropriately strong secure implementations, the actions of a computational adversary do not go beyond those of a symbolic one. One set of results that we describe in Section 3.1 and Section 3.2 are based on so-called *trace mapping lemmas*. Such lemmas state that each computational execution trace is the image of a symbolic execution. Using trace mapping, certain security properties proved using symbolic models find immediate translation in computational ones. A recent result described in Section 3.3 goes even further: not only traces can be transferred but it is shown that whenever two symbolic processes are observationally equivalent then the corresponding computational processes are computationally indistinguishable. This extends the scope of such results to an even larger set of security properties.

A second important direction uses the concept of *simulatability*. Roughly speaking, in such settings security is defined by relating a real system with an idealized version of the same system. The idea is to show that for any attack against the real system there exists an analogous attack against the idealized one. Since the ideal system is secure by construction it follows that no attack is possible against the real system. The desired connection between symbolic and computational models can therefore be obtained by showing a simulatability relation between an idealized, symbolic cryptographic system and a real, cryptographic implementation of such a system. In Section 3.4 we give some background regarding the general notion of simulatability. Then, in Section 3.5, we describe how simulatability had been applied in the context of a cryptographic library. In the same section we describe various applications of the simulatable cryptographic library.

The direct approach. In the remaining sections we describe a different direction for getting "the best of both worlds". These approaches aim at applying symbolic techniques directly to obtain computational security guarantees, without making use of abstract models.

One direction that we describe in Section 4 is to design logics with semantics given in terms of computational models. Proofs can then be carried out using well-designed proof rules which are shown to be computationally sound. Such a logic is obviously not complete and there might be security properties which hold but cannot be proven using the axioms of the logic. Nevertheless, the proof rules turn out to be powerful enough to allow proofs of a large range of properties and protocols. In the same section we describe work on a type system which ensures computational security. Then, in Section 5 we discuss a second technique which consists in introducing symbolic calculi that can be (provably) securely implemented at the cryptographic level. These symbolic calculi do not make explicit use of cryptography but provide high-level constructs such as confidential and authentic channels which are implemented using a secure cryptographic protocol. Finally, in Section 6 we discuss work using proof assistants for cryptographic proofs. We describe work that relies on the general purpose proof assistant Coq which mainly checks the correctness of the proof as well as work on the special purpose tool CryptoVerif which moreover achieves a high level of automation. Concluding remarks can be found in Section 7.

2 Computational soundness: the passive adversary case

The most basic setting for computational soundness is that of passive adversaries who can only observe the network traffic but cannot interfere with the execution of the protocol. This is the setting considered in seminal result of Abadi and Rogaway who were the first to show that links between symbolic and computational models can be established [AR00, AR02]. We give the details of their work in Section 2.1 and survey the many extensions that followed in Section 2.2.

2.1 The Abadi-Rogaway result

The result of Abadi and Rogaway shows that if a symbolic notion of secrecy of data that occurs in a message is satisfied, then a computational notion is also satisfied [AR00, AR02]. Their result holds for a class of messages constructed as in the following section.

2.1.1 Formal expressions and equivalence.

On the formal side, one considers a simple grammar for expressions. The expressions consider two base types for keys and Booleans which are taken from two disjoint sets **Keys** and **Bool**. Keys and Booleans can be paired and encrypted.

$$\begin{array}{ll}
M, N ::= & \quad \textit{expressions} \\
\quad K & \text{key } (K \in \mathbf{Keys}) \\
\quad i & \text{bit } (i \in \mathbf{Bool}) \\
\quad \langle M, N \rangle & \text{pair} \\
\quad \{M\}_K & \text{encryption } (K \in \mathbf{Keys})
\end{array}$$

For example the formal expression $\langle K_1, \{\langle 0, K_2 \rangle\}_{K_1} \rangle$ represents a pair: the first component of this pair is the key K_1, the second, the encryption with key K_1 of the pair consisting of the boolean constant 0 and the key K_2.

Before defining the equivalence relation between terms we first need to define the deducibility relation $\vdash$. Intuitively, $M \vdash N$, if the adversary can learn the expression N from the expression M. Formally, $\vdash$ is the smallest relation, such that

$$\begin{array}{l}
M \vdash M \qquad M \vdash 0 \qquad M \vdash 1 \\
\text{if } M \vdash N_1 \text{ and } M \vdash N_2 \text{ then } M \vdash \langle N_1, N_2 \rangle \\
\text{if } M \vdash \langle N_1, N_2 \rangle \text{ then } M \vdash N_1 \text{ and } M \vdash N_2 \\
\text{if } M \vdash \{N\}_K \text{ and } M \vdash K \text{ then } M \vdash N \\
\text{if } M \vdash N \text{ and } M \vdash K \text{ then } M \vdash \{N\}_K
\end{array}$$

For example, if $M = \langle K_1, \{\langle 0, K_2 \rangle\}_{K_1} \rangle$, then we have that $M \vdash K_2$. Moreover, $M \vdash 1$, as the constants 0 and 1 are always known to the attacker.

The equivalence relation between terms is based on the equality of the *patterns* associated to each term. A pattern represents the adversary's view of a term. Patterns extend the grammar defining terms by the special symbol $\Box$. The pattern of a term replaces encryptions for which the key cannot be deduced by $\Box$. This idea is formally captured by the following function p. The function takes as arguments a term and a set T of keys and is defined inductively as follows.

$$\begin{array}{rcll}
p(K, T) & = & K & (K \in \mathbf{Keys}) \\
p(i, T) & = & i & (i \in \mathbf{Bool}) \\
p(\langle M, N \rangle, T) & = & \langle p(M, T), p(N, T) \rangle & \\
p(\{M\}_K, T) & = & \begin{cases} \{p(M, T)\}_K & \text{if } K \in T \\ \Box & \text{else} \end{cases} &
\end{array}$$

The pattern of an expression M is defined by

$$\textit{pattern}(M) = p(M, \{K \in \mathbf{Keys} \mid M \vdash K\}).$$

For instance $\textit{pattern}(\langle K_1, \{\langle 0, \{1\}_{K_2} \rangle\}_{K_1} \rangle) = \langle K_1, \{\langle 0, \Box \rangle\}_{K_1} \rangle$.

Furthermore, expressions M and N are formally indistinguishable, written $M \equiv N$ if and only if $\textit{pattern}(M) = \textit{pattern}(N)\sigma$, where σ is a bijective renaming of keys. For example, we have that $0 \not\equiv 1$, $K_0 \equiv K_1$, $\{0\}_{K_1} \equiv \{1\}_{K_0}$ and $\langle K_0, K_0 \rangle \not\equiv \langle K_0, K_1 \rangle$.

2.1.2 Computational setting and hypotheses on the implementation.

In the computational setting, one reasons at the level of bitstrings and algorithms executed on Turing Machines, rather than on abstract terms. Expressions are interpreted as bitstrings by instantiating each of the symbolic operations (including the constants) via appropriate algorithms. In particular we assume a computational pairing function that takes as input two bitstrings m_1 and m_2 and outputs their concatenation $\langle m_1, m_2 \rangle$. The function is such that

m_1 and m_2 are easily extractable from $\langle m_1, m_2\rangle$. Furthermore, we use a concrete encryption scheme, which is a triple of polynomial time algorithms $\mathcal{K}, \mathcal{E}, \mathcal{D}$ for key generation, encryption and decryption respectively. The key generation algorithm is parameterized by a security, or complexity parameter $\eta \in 1^*$. Intuitively, η defines the key length. As expected we require that $\mathcal{D}_k(\mathcal{E}_k(m, r)) = m$ for any $k \in \mathcal{K}(\eta)$, message m, and random bitstring r (that represents the coins of the probabilistic encryption algorithm).

The Abadi-Rogaway result relies on a security notion for encryption schemes termed "type-0" in the original paper [AR00]. Here, we call schemes that satisfy this notion, which we recall bellow, simply *secure*. Informally, secure schemes hide all information about encrypted plaintexts (including their length) and hide all information about the encryption key. This notion is significantly stronger than more standard ones which allow for ciphertexts to reveal the length of the underlying plaintext as well as partial information about the encryption key. The stronger assumption is used for simplicity as the Abadi-Rogaway framework can be further refined to only rely on the more standard notions.

An encryption scheme is *secure* if for any security parameter η and any probabilistic polynomial time Turing machine $\mathcal{A}$ (the adversary) the advantage

$$\begin{array}{rcl} Adv(\mathcal{A}) & = & \Pr[k, k' \overset{R}{\longleftarrow} \mathcal{K}(\eta) : \mathcal{A}^{\mathcal{E}_k(\cdot), \mathcal{E}_{k'}(\cdot)}(\eta) = 1] - \\ & & \Pr[k \overset{R}{\longleftarrow} \mathcal{K}(\eta) : \mathcal{A}^{\mathcal{E}_k(0), \mathcal{E}_k(0)}(\eta) = 1] \end{array}$$

is a negligible function of η. Here, $x \overset{R}{\longleftarrow} \mathcal{D}$ denotes the random sampling of an element of distribution $\mathcal{D}$ and $\mathcal{A}^{\mathcal{O}}$ is the Turing Machine $\mathcal{A}$ that has access to a set of oracles $\mathcal{O}$. Intuitively, one requires that an adversary cannot distinguish the case where he is given two encryption oracles encrypting with two different keys from the case where he is given twice the same encryption oracle always encrypting the constant bitstring representing 0 with the same key. Note that this security under this notion implies that encryption needs to be randomized, so that an adversary does not see identical answers when confronted with the second pair of (identical) oracles. In [AR02], the authors provide constructions for such schemes from standard cryptographic assumption.

A recurrent theme in computational soundness is that of acyclic expressions. The reason is that an encryption scheme respecting the above security definition may be insecure as soon as the adversary is given a *key cycle*. We say that a key K_1 *encrypts* a key K_2 in a formal expression M if M contains a subexpression $\{N\}_{K_1}$ and K_2 occurs in N. In this way any expression M defines a binary relation *encrypts* on keys. We say that an expression contains a key cycle if and only if the corresponding encrypts relation is cyclic. For instance $M_1 = \{K\}_K$ contains a key cycle as K encrypts K. In $M_2 = \{\{K_1\}_{K_2}\}_{K_3}$ we have that K_3 encrypts K_1, K_3 encrypts K_2 and K_2 encrypts K_1 and hence M_2 does not contain any key cycle. In Abadi and Rogaway's main result, key cycles are therefore forbidden. Similar conditions can be found in most soundness results. To better understand the problem of key cycles suppose that $\mathcal{SE} = (\mathcal{KG}, \mathcal{E}, \mathcal{D})$ is a secure encryption scheme and let $\mathcal{SE}' = (\mathcal{KG}', \mathcal{E}', \mathcal{D}')$ be defined as follows:

$$\mathcal{KG}' = \mathcal{KG}, \qquad \mathcal{E}'_k(m, r) = \begin{cases} \mathcal{E}_k(m, r) & \text{if } m \neq k \\ \langle \mathsf{const}, k\rangle & \text{if } m = k \end{cases},$$

$$\mathcal{D}'_k(c) = \begin{cases} \mathcal{D}_k(c) & \text{if } c \neq \langle \mathsf{const}, k\rangle \\ k & \text{if } c = \langle \mathsf{const}, k\rangle \end{cases}$$

where const is a constant such that for any key k, the concatenation $\mathsf{const} \cdot k$ does not belong to the set of possible ciphertexts obtained by $\mathcal{E}$. Obviously, if the attacker is given a key cycle of length 1, *e.g.*, $\mathcal{E}'_k(k, r)$, the attacker directly learns the key. It is also easy to see that $\mathcal{SE}'$ is a secure encryption scheme as it behaves as $\mathcal{SE}$ in nearly all cases (in the security experiment the adversary can make a query for encrypting k with itself only with negligible probability).

The notion of computational indistinguishability requires that an adversary cannot distinguish two (families of) distributions, with better than negligible probability. Let $\mathcal{D} = \{\mathcal{D}_\eta\}$ and $\mathcal{D}' = \{\mathcal{D}'_\eta\}$ be two families of probability distributions. $\mathcal{D}$ and $\mathcal{D}'$ are computationally indistinguishable, written $\mathcal{D} \approx \mathcal{D}'$ if for any η and any probabilistic polynomial time Turing machine $\mathcal{A}$, the advantage

$$Adv(\mathcal{A}) = \Pr[x \xleftarrow{R} \mathcal{D}_\eta : \mathcal{A}(\eta, x) = 1] - \Pr[x \xleftarrow{R} \mathcal{D}'_\eta : \mathcal{A}(\eta, x) = 1]$$

is a negligible function of η.

2.1.3 Interpretation of formal expressions and soundness result.

The Abadi-Rogaway result links the notion of pattern equivalence on expressions defined in the previous section with an appropriate notion of computational equivalence defined on distributions. These distributions are associated to expressions using the following algorithms that convert formal expressions into bitstrings.

Bitstrings are tagged using types “key”, “bool”, “pair” and “ciphertext”. The **Initialize** procedure uses $\mathcal{K}$ to generate actual keys for each of the key symbols that occurs in M (that is for each key $K \in Keys(M)$). Then, then **Convert** procedure implements encryption using algorithm $\mathcal{E}$.

```
$\mathbf{Initialize}_\eta(M)$
    for $K \in Keys(M)$ do $\tau(K) \xleftarrow{R} \mathcal{K}(\eta)$

$\mathbf{Convert}(M)$
    if $M = K$ $(K \in \mathbf{Keys})$ then
        return $\langle \tau(K), \text{“key”} \rangle$
    if $M = b$ $(b \in \mathbf{Bool})$ then
        return $\langle b, \text{“bool”} \rangle$
    if $M = \langle M_1, M_2 \rangle$ then
        return $\langle \langle \mathbf{Convert}(M_1), \mathbf{Convert}(M_2) \rangle, \text{“pair”} \rangle$
    if $M = \{M_1\}_K$ then
        $x \xleftarrow{R} \mathbf{Convert}(M_1)$
        $y \xleftarrow{R} \mathcal{E}_{\tau(K)}(x)$
        return $\langle y, \text{“ciphertext”} \rangle$
```

The **Initialize** and **Convert** procedures associate to a formal term M a family of probability distributions $[\![M]\!] = \{[\![M]\!]_\eta\}$.

Abadi and Rogaway’s main result is that for any formal expressions M and N that do not contain key cycles, whenever the computational interpretation of the terms uses a secure encryption scheme (as defined above), then $M \equiv N$ implies that $[\![M]\!] \approx [\![N]\!]$. In other words, they show that pattern-based equivalence is a sound abstraction of cryptographic indistinguishability.

2.2 Extensions of the Abadi-Rogaway result

The initial result of Abadi and Rogaway has given rise to many extensions. Some of these extensions consider the question of completeness of their logic. Other extensions consider different implementations of encryption (with variants of the initial patterns) as well as other cryptographic primitives.

2.2.1 Completeness of the Abadi-Rogaway logic.

In [MW02, MW04a], Micciancio and Warinschi show that the Abadi-Rogaway logic is not *complete* as presented in the original paper. Here, by completeness we mean that $M \not\equiv N$ implies that $[\![M]\!] \not\approx [\![N]\!]$, i.e., whenever two formal expressions are not equivalent, then the computational interpretation of these two messages should be distinguishable. Micciancio and Warinschi exhibit a counter-example by constructing a secure encryption scheme and two symbolic expressions that are not symbolically equivalent, which yet give rise to indistinguishable probability distribution ensembles.

They show that completeness can be recovered by implementing encryption with a scheme that is *authenticated*. Informally, an encryption scheme is authenticated if an adversary cannot produce a valid ciphertext different from ciphertexts honestly produced by the parties that posses the encryption key. Gligor and Horvitz [HG03] further refine this completeness result. They introduce a new security criterion for encryption schemes, *weak key-authenticity test for expressions* (WKA-EXP), which is strictly weaker than authenticated encryption. WKA-EXP is both sufficient and necessary for completeness.

2.2.2 Public-key encryption.

In [Her03, Her05], Herzog shows a similar result as Abadi and Rogaway, but for public-key encryption. Patterns are generalized in the expected way for expressions that use public-key encryption. The problem of key-cycles also persists in this setting. To define a key-cycle of an expression M in the public-key setting one constructs a graph G_M: the set of vertices is the set of public/private key pairs $\{(pubK_1, privK_1), \ldots, (pubK_n, privK_n)\}$; there exists an edge from $(pubK_i, privK_i)$ to $(pubK_j, privK_j)$ if $pubK_i$ encrypts $privK_j$ in M. M has no key-cycle if G_M is acyclic. Herzog presents a soundness theorem, similar to the one of Abadi and Rogaway, whenever the encryption scheme used for the computational interpretation provides indistinguishability under chosen-ciphertext attacks (IND-CCA2 security).

2.2.3 Composed keys.

In [LC04], Laud and Corin extend the original soundness theorem to allow arbitrary expressions as keys. The tricky part is again to handle key-cycles correctly. As arbitrary expressions are used in the position of keys, the definition of what is a key cycle is not obvious. Rather than giving an explicit definition of what is a key-cycle, key-cycles are directly captured in the formal equivalence relation. More precisely, an expression is not formally equivalent to its pattern whenever a “key-cycle” is present. For instance, $\{\langle K_1, K_2\rangle\}_{\langle K_1, K_2\rangle} \not\equiv \Box$ and $\langle \{K_1\}_{\langle K_1, K_2\rangle}, \{K_2\}_{\langle K_1, K_2\rangle}\rangle \not\equiv \langle \Box, \Box\rangle$. However, $\{K_1\}_{\langle K_1, K_2\rangle} \equiv \Box$, because the second part of the key K_2 is never encrypted.

2.2.4 Handling key cycles.

Key cycles have gained a lot of attention in the context of computational soundness. The reason is that there is an inherent difference between their treatment in symbolic models (where such cycles do not cause any troubles) and the computational model (where standard security definitions do not guarantee security in presence of key-cycles.) There are two natural approaches to reconcile this apparent difference.

One possibility is to strengthen the symbolic attacker. This is the direction explored by Laud in [Lau02]. The idea is to modify the symbolic deduction relation so that whenever a key occurs in a key cycle then it becomes known to the attacker. Laud shows an unconditional soundness theorem in the style of Abadi and Rogaway (unconditional in the sense that formal expressions may contain key cycles).

The second possibility is to strengthen the computational notion as to guarantee security even in the presence of key-cycles. This is the approach adopted in [ABHS05], Adão et al. They consider a stronger security notion, called *key-dependent message* (KDM) security which demands security even in the presence of such cycles. They show that soundness holds in a public-key setting in the presence of key cycles when a KDM secure encryption scheme is used for the computational interpretation of encryption. They also prove a separation between standard security notions (IND-CCA2) and KDM security and demonstrate that IND-CCA2 security is not sufficient to provide soundness in the presence of key-cycles. Schemes secure under the KDM notion can be easily constructed in the random oracle model, but schemes secure in the standard model seem much harder to construct. Recently, Boneh et al. [BHHO08] demonstrated the existence of an asymmetric encryption scheme secure under key dependent message attacks in a restricted sense: their scheme does not permit the encryption of messages that depend in arbitrary ways on the set of secret keys.

In most of the other approaches, one has to assume that key cycles cannot be generated, even when the adversary interacts arbitrarily with the protocol. Whether a key cycle can be generated is undecidable in the general case but it has been shown to be NP-complete in the symbolic setting, for an active adversary and a bounded number of sessions [CZ06].

2.2.5 Partial information leakage and information theoretic security.

Adão et al. [ABS05] consider different computational implementations of the encryption function. In particular they show soundness and completeness when *which-key* and *length-key revealing* encryption schemes are used. A which-key revealing encryption scheme allows the adversary to detect when two ciphertexts have been encrypted with the same key. At the symbolic level this is reflected by indexing the boxes with the encryption key, yielding a more precise equivalence relation. For instance, $pattern(\{0\}_K) = \Box_K$ and hence we have that $\langle \{0\}_K, \{1\}_K \rangle \not\equiv \langle \{0\}_K, \{1\}_{K'} \rangle$. A length-key revealing encryption scheme allows the attacker to learn the length of the plaintext. At the symbolic level the boxes are indexed with the length of the plaintext to reflect this partial information leakage.

The authors also consider the case where encryption is implemented by a one-time pad. Whenever encryption keys are only used once they show that one obtains soundness and completeness with respect to an information-theoretic setting. In such a setting the equivalence is the equality of the probability distributions rather than indistinguishable by a polynomial-time bounded adversary.

2.2.6 Hash functions.

Garcia and van Rossum [GvR06] extend the Abadi-Rogaway logic to hash functions. Soundness theorems for hash functions are particularly tricky as in the symbolic model, hash functions do not leak any partial information about the hashed message. Typical computational security definitions for hash functions provide weaker guarantees, such as one-wayness. Garcia and van Rossum show a soundness result when hash functions are implemented using oracle hashing. Oracle hashing has been introduced by Canetti: it is a probabilistic hash function which requires a verification algorithm to check whether a hash corresponds to a given message. These are hash functions that do hide all partial information about the message that is being hashed. In the journal version [GvR08], they extend Micciancio and Warinschi's completeness result to hash functions in a similar way.

2.2.7 Modular exponentiation.

Bresson et. al [BLMW07] give an extension of the Abadi-Rogaway logic with modular exponentiation. They show how to extend the notion of patterns in order to capture the information that is leaked through exponentiation, which are essentially linear dependencies between the various exponents. For example, the symbolic secrecy notion captures the idea that an adversary can observe that in the expression (g^x, g^y, g^{2x+y}) the third term can be obtained by squaring the first one and multiplying it with the second. Non-linear relations, as in the expression (g^x, g^y, g^{x+xy}), cannot be observed by the adversary. The soundness for the resulting language relies on a generalization of the Diffie-Hellman assumption. The authors also argue that in the most interesting situations, the authors show that the basic Diffie-Hellman assumption suffices.

In the same vein than [BLMW07], Mazaré [Maz07, KM09] presents an extension the Abadi-Rogaway logic with a bilinear pairing operation. Their soundness result assumes the hardness of the bilinear decisional Diffie-Hellman problem and an IND-CPA encryption scheme. The soundness result is illustrated on the Joux tripartite Diffie-Hellman protocol, as well as the TAK-2 and TAK-3 protocols.

2.2.8 Offline guessing attacks.

In security protocols passwords or other weak data are often used as encryption keys. For such protocol an important security property is resistance to offline guessing attacks. In such attacks an attacker first collects (possibly by interacting with the protocol) some data. In a second phase, he *guesses* a password out of a dictionary. If the attacker has a means to verify that his guess was correct using the data he had gathered, then the protocol is subject to a guessing, or dictionary attack. In [AW05a], Abadi and Warinschi have shown soundness results for protocols that use password encryptions. They define the computational security of a password encryption primitive: for any two passwords, any polynomially bounded adversary, that is given these two passwords and given access to an oracle, encrypting samples drawn from a plaintext distribution, is not able to distinguish whether the oracle uses the first or the second password for encryption. They also define formal and computational security of expressions against offline guessing attacks in terms of indistinguishability. Then for symmetric, asymmetric and password encryptions with secure implementation they show two soundness theorems. The first one is an extension of the Abadi-Rogaway soundness theorem for indistinguishability. The second theorem states that whenever a formal expression E hides passwords, then its computational interpretation also hides passwords. These results hold for IND-CPA secure symmetric and asymmetric schemes, and for password-based encryption schemes that "securely" encrypt keys and ciphertexts of the symmetric and asymmetric schemes. In addition, it only holds for expressions that do not contain key cycles.

2.2.9 Cryptographically controlled access control to XML.

A compelling application of computational soundness against passive adversaries was given by Abadi and Warinschi [AW05b, AW08]. The focus of that work is the security of a scheme that uses encryption to enforce access control policies to XML documents. The scheme, designed by Miklau and Suciu [MS03] explains how to obtain from a given XML document and a given access policy a so-called protection: a partially encrypted XML document which enforces the original access policy. The guarantees for the scheme were rather informal.

Abadi and Warinschi formalize the scheme using a symbolic language for expressions that extends the one of Abadi and Rogaway with secret sharing schemes. Then, they show that secrecy as demanded by the policy used to create a certain protection on an XML document is satisfied in a symbolic sense: data that should be secret according to the policy is *symbolically secret* in the expression that describes the protection. It then follows using the computational soundness of the language for expressions that the same data is also computationally secret. The soundness results hold for implementations that use IND-CPA encryption schemes and n-out-of-n secure secret sharing schemes.

2.2.10 Soundness against an adaptive adversary.

Micciancio and Panjwani [MP05] show a soundness result for encryption and pairing in the presence of a slightly stronger, *adaptive* adversary. Soundness is defined through the following experiment. An adversary has access to a left-right oracle, which given on input two terms M_1 and M_2, returns a sample of the computational interpretation of M_b, where b is the challenge bit of the oracle. The adversary can interact with the oracle but is only allowed to submit queries such that the sequence of queries $(M_1^1, M_2^1), \ldots, (M_1^\ell, M_2^\ell)$ sent to the oracle is such that $\langle M_1^1, \ldots, M_1^\ell \rangle$ is formally equivalent, i.e. has the same pattern up to renaming, to $\langle M_2^1, \ldots, M_2^\ell \rangle$. The adversary wins if he succeeds in outputting b with non-negligible probability. Note that the oracle is stateful and implements terms in a consistent way, i.e. if a key has been drawn in a previous query the same value is reused in subsequent queries. An adaptive adversary is strictly stronger than a purely passive one as he can choose his queries after having already obtained the implementation of some terms. On the technical level, the fact of having an adaptive adversary raises the problem of selective decommitment which is overcome by imposing the following condition: if a key is used to encrypt a message it either must have been sent previously in plaintext or it never appears in plaintext. The usefulness of an adaptive adversary is illustrated by deriving computationally sound symbolic model for the analysis of multicast key distribution protocols. In this model, the adversary cannot directly interact with the protocol participants, but he can influence the control flow.

2.3 Soundness of static equivalence

Baudet, Cortier, and Kremer have considered a more general alternative to the approach described in the previous sections. They develop a framework in which symbolic secrecy is expressed in terms of *static equivalence*, a well-established equivalence relation from cryptographic pi-calculi[BCK05, BCK09]. This approach is more general in that it does not depend on a particular set of primitives.

2.3.1 Abstract and computational algebras.

Independence from a particular primitives is reflected in their use of an arbitrary *abstract algebra* to describe the messages exchanged in a protocol. The algebra is defined over a many-sorted first-order signature equipped with an *equational theory*. For instance, symmetric, deterministic encryption is modeled by the theory E_{enc} generated by the classical equation $E_{enc} = dec(enc(x, y), y) = x$. Equality between two terms is generally interpreted modulo the equational theory (denoted $=_E$ for an equational theory E). For example, $dec(enc(m, k), k) =_{E_{enc}} m$. Given an abstract signature a computational algebra A is defined by associating to every sort s of the abstract algebra a set of bitstrings $[\![s]\!]_A \subseteq \{0, 1\}^*$ with an efficient procedure for drawing random elements, and to every function f a computational function $[\![f]\!]_A$. Given a symbolic term T, a distribution $[\![T]\!]_A$ is associated by drawing a random element of the corresponding sort for each name and replacing each function symbol by its computational counterpart.

2.3.2 Security notions, soundness, and faithfulness.

The two security notions which are considered are deducibility and static equivalence. Deducibility formalizes which are the terms that an attacker can compute from a given sequence of terms. Static equivalence models whether two sequences of terms can be distinguished. Both deducibility and static equivalence are parameterized by an equational theory. In this approach, static equivalence replaces the pattern-based formal equivalence.

To reason about the soundness of implementations Baudet et al. define soundness for the three relations $=_E$, $\vdash_E$ and $\approx_E$. Soundness of $=_E$ means that whenever two terms are symbolically equal (modulo E), any sample drawn from the distribution implementing those terms should be equal with overwhelming probability. Soundness of $=_E$ is generally a hypothesis which reflects that the equational theory is a reasonable abstraction of the primitives. Similarly, they define soundness for deducibility and static equivalence. When a term is not deducible from a sequence of terms, then an attacker given the distribution implementing the given sequence of terms, should be able to output a sample of the distribution implementing the term with only negligible property. When two sequences of terms are statically equivalent, then the distributions associated to these sequences should be indistinguishable.

Faithfulness of those three relations on the other hand represents a strong version of completeness. Whenever two terms are not equal, a term is deducible or two sequences of terms are not statically equivalent, a computational adversary can show this with overwhelming probability (rather than non-negligible probability which would be completeness). Intuitively, when the relations are faithful, for any symbolic attack there exists an efficient computational attack.

It is shown that for many theories $\approx_E$-soundness implies all other notions of soundness and faithfulness. This emphasizes the importance of $\approx_E$-soundness.

2.3.3 Examples: groups, XOR, ciphers and lists

In [BCK05, BCK09], Baudet et al. consider several equational theories to illustrate their framework. First they show the $\approx_E$-soundness of an equational theory modeling groups implies the hardness of several classical cryptographic problems: the discrete logarithm, computational Diffie-Hellman, decisional Diffie-Hellman and RSA problems. Note that this is not a soundness result. It shows that any candidate implementation for $\approx_E$-soundness requires at least the hardness of the usual cryptographic problems. Second, they show the unconditional $\approx_E$-soundness of a theory of XOR. The soundness proof reflects the unconditional security (in the information-theoretic sense) of the One-Time Pad. Finally, they show $\approx_E$-soundness of a theory of ciphers and lists (ciphers are deterministic, length-preserving, symmetric encryption schemes).

2.3.4 Soundness of offline guessing attacks and static equivalence.

In [ABW06], Abadi, Baudet and Warinschi use the framework of [BCK05, BCK09] to show $\approx_E$-soundness for an equational theory useful in the context of offline guessing attacks. This theory includes symmetric, and asymmetric encryption as well as pairing. A consequence of this soundness result is its applicability to defining and reasoning about off-line guessing attacks in terms of static equivalence. The result is an intuitively appealing implication to computational security against off-line attacks.

2.3.5 Static equivalence vs formal indistinguishability relations.

In [BMS06], Bana, Mohassel and Stegers argue that the notion of static equivalence is too coarse and not sound for many interesting equational theories. They introduce a general notion of formal indistinguishability relation. This highlights that soundness of static equivalence only holds for a restricted set of well-formed frames (in the same vein Abadi and Rogaway used restrictions to forbid key cycles). They illustrate the unsoundness of static equivalence for modular exponentiation.

2.3.6 Adaptive soundness of static equivalence.

The analogue of [MP05], but for the setting where pattern based equivalence is replaced with static equivalence, has been provided by Kremer and Mazaré [KM07] who extend the framework of [BCK05]. In this case, adaptive soundness is defined through an experiment. The adversary interacts with a left-right oracle, which given two symbolic terms, returns either a sample of the concrete implementation of the first or the second term, according to the oracle's challenge bit. As in [MP05], the adversary is restricted to only provide queries such that the left-hand terms and the right-hand terms form two statically equivalent sequences, rather than pattern-equivalent sequences. They show adaptive soundness of static equivalence for an equational theory modeling modular exponentiation (for a class of well-formed frames, hence not contradicting [BMS06] and under similar assumptions as in [BLMW07]), as well as symmetric encryption with composed keys which can be computed using modular exponentiation or exclusive or.

2.4 Computationally secure information flow

A different kind of soundness results have been obtained in the area of information flow. Informally, a program has secure information flow if the public outputs of the program do not leak information about its confidential inputs. Classically, information flow is defined as non-interference requiring that no information, in the information-theoretic sense, is leaked. In particular such a definition forbids publishing the encryption of a confidential value. Allowing encrypted confidential values to be published is generally refered to as cryptographic declassification.

Laud [Lau01] pioneered the area of computationally secure information flow. He proposes a computational definition of secure information flow in the presence of a probabilistic polynomial time adversary. The programming language he considers contains assignment, loops, conditional, sequential composition and application of some operators. In particular the operators contain symmetric encryption and key generation. Laud presents a static analysis which ensures computational secure information flow assuming an implementation that uses a which-key and repetition-concealing IND-CPA encryption scheme. Two limitations of the paper are that keys can only be used at a key position, and not as data, as well as the fact that one must be able to decide statically whenever two variables contain the same encryption key. These two restrictions are relaxed in [Lau03] by refining the static analysis.

In [LV05], Laud and Vene propose a computationally sound type system which ensures secure information flow. A similar approach is presented by Smith and Alpízar [SA06]. A difference is that Smith and Alpízar allow an explicit decryption operator (and hence require IND-CCA security to achieve soundness). However, they do not manipulate keys, but only consider a single key which is used for encryption and decryption, but never as plaintext. Courant, Ene and Lakhnech [CEL07] also design a cryptographically sound type system. The basic data contain constants and uniformally sampled bitstrings. Operations include exclusive or and applications of deterministic, length preserving encryption, i.e. ciphers. As Smith and Alpízar they consider a single key which is only used for encryption and decryption. Due to the deterministic nature of ciphers, which do not hide repetitions, subtle flows may still arise. Courant et al. show cryptographic soundness of their typing system under the hypothesis that encryption scheme respects the pseudo random permutation, PRP for short, security notion. Moreover, the soundness result is shown in the concrete (or exact) model, rather than being asymptotic.

An abstract model for reasoning about secure information flow is the framework of Askarov, Hedin and Sabelfeld for dealing with *cryptographically-masked flows* [AHS06]. Here, they consider an imperative language with encryption and decryption operations which comes with a non-deterministic semantics, avoiding reasoning about probabilities. In [Lau08], Laud investigates the computational soundness of cryptographically masked flows and identifies the necessary restrictions and cryptographic assumptions. In particular, symmetric encryption needs to satisfy length- and which-key concealing KDM, as well as a key-dependent message variant of plaintext integrity. Laud also suggests a simpler but equivalent model with a completely deterministic abstract semantics. The security definition in this model is based on Abadi and Rogaway's pattern equivalence. The soundness result directly follows from [ABHS05], discussed earlier in this survey.

3 Computational soundness: the active adversary case

The focus of the previous section was on the case where the adversary only observe the network traffic and tries to gain information about the secrets used in an execution. In this section we shift attention to the case of active adversaries, namely adversaries that can interfere with the execution of protocols.

As explained in the previous sections the main abstraction used by symbolic models is to represent messages (that is bitstrings) by symbolic terms. A second abstraction which is quite important for the active setting is given by how adversarial capabilities are treated.

In symbolic models, the adversary can build new messages using an *a priori* fixed set of symbolic inference rules. For example, he can get information from a encrypted messages only if he has the appropriate decryption key. On the other hand, in computational models the adversary is a probabilistic polynomial-time (p.p.t. for short) Turing machine. This captures the idea that a potential adversary can perform arbitrary computations while tampering with the protocol, provided it takes a *reasonable*, that is polynomial, time. In particular, this assumption captures the possibility that the adversary may try to guess secrets (e.g. keys). Note that in both models it is assumed that the adversary has complete control of the network: he can intercept, send and block messages. An additional gap between the symbolic and the computational models is in how security properties are specified. For example, secrecy is usually stated in symbolic models as a reachability property while in computational models, it is formalized as the indistinguishability of adversary views.

In this section we survey three approaches developed to bridge the gap between symbolic and computational models. Recall that the goal is to understand when security proved using symbolic models implies meaningful security properties for protocols with respect to computational ones. These approaches are the trace mapping approach, the process mapping approach, and the simulation based approach.

3.1 The trace-mapping based approach

3.1.1 Syntax.

Messages are modeled by a term algebra, given with sorts. For example, the algebraic signature Σ may contain sorts Nonce, Label, $\mathsf{Ciphertext}$, $\mathsf{Signature}$, and Pair for respectively nonces, labels, ciphertexts, signatures, and pair. Typical operations are pairing $\langle \_, \_ \rangle$, public key encryption $\{\_\}^{\_}_{\_}$, and signing $[\_]^{\_}_{\_}$. One may already notice a difference with the passive case as described by Abadi and Rogaway. Probabilistic primitives like encryption or signatures are now represented with ternary symbols instead of binary symbols. The third argument explicitly models the randomness used in these primitives and allows one for instance to capture the fact that encrypting twice the same message m with the same key k yields different ciphertexts represented by $\{k\}^{r_1}_m$ and $\{k\}^{r_2}_m$.

Protocols are specified using the algebra of terms constructed over the above signature from a set X of sorted variables. The messages that are sent by participants are specified using terms in $\mathsf{T}_\Sigma(\mathsf{X})$, the free algebra generated by X over the signature Σ. The individual behavior of each protocol participant is defined by a *role* describing a sequence of message receptions/transmissions, and a k-party protocol is given by k such roles.

It is worth emphasizing that the term algebra used in this setting is richer than the algebra typically used in automatic tools for security protocols. One important difference is that the latter typically omit the explicit randomness argument discussed above. Furthermore, the model uses different symbolic operations for signature and encryption. Quite often, the models for tools model signatures as encryptions with the decryption key. Nevertheless, it can be shown that under certain conditions security with respect to the simpler models implies security with respect to the richer models above [CHW06].

3.1.2 Execution models.

Two types of executions are then defined for protocols: the symbolic and the concrete ones. In both models, the adversary has a complete control of the network: he can intercept, send and block messages. More precisely, the adversary can interact with the protocol through three kinds of actions.

- $\textbf{corrupt}(a_1, \ldots, a_l)$: the adversary can corrupt parties by outputting a set of identities. He receives in return the secret keys corresponding to the identities. It happens only once at the beginning of the execution.

- $\textbf{new}(i, a_1, \ldots, a_k)$: the adversary can initiate new sessions selecting the role i and the instantiation $a_1, \ldots, a_k$ for the agents involved in that session.

- $\textbf{send}(\mathsf{sid}, m)$: the adversary can send a message m to a target session sid.

In the symbolic setting the honest parties and the adversary exchange elements of a certain term algebra; the adversary can only send messages deducible from the previously received messages following the standard Dolev-Yao rules described in Figure 1.

In the concrete execution model, the honest parties and the adversary are p.p.t. Turing machines and the messages that are exchanged are bit-strings and

$$
\begin{array}{llll}
\dfrac{}{S \vdash m}\; m \in S & \dfrac{}{S \vdash b, \mathsf{ek}(b), \mathsf{vk}(b)}\; b \in \mathsf{X}.a & & \text{Initial knowledge} \\[2ex]
\dfrac{S \vdash m_1 \quad S \vdash m_2}{S \vdash \langle m_1, m_2 \rangle} & \dfrac{S \vdash \langle m_1, m_2 \rangle}{S \vdash m_i}\; i \in \{1,2\} & & \text{Pairing and unpairing} \\[2ex]
\dfrac{S \vdash \mathsf{ek}(b) \quad S \vdash m}{S \vdash \{m\}^{\mathsf{adv}(i)}_{\mathsf{ek}(b)}}\; i \in \mathbb{N} & \dfrac{S \vdash \{m\}^{l}_{\mathsf{ek}(b)} \quad S \vdash \mathsf{dk}(b)}{S \vdash m} & & \begin{array}{c}\text{Encryption}\\ \text{and decryption}\end{array} \\[2ex]
\dfrac{S \vdash \mathsf{sk}(b) \quad S \vdash m}{S \vdash [m]^{\mathsf{adv}(i)}_{\mathsf{sk}(b)}}\; i \in \mathbb{N} & \dfrac{S \vdash [m]^{\mathsf{ag}(i)}_{\mathsf{sk}(b)}}{S \vdash [m]^{\mathsf{adv}(j)}_{\mathsf{sk}(b)}}\; i, j \in \mathbb{N} & \dfrac{S \vdash [m]^{l}_{\mathsf{sk}(b)}}{S \vdash m} & \text{Signature}
\end{array}
$$

Figure 1: Deduction rules for the formal adversary.

depend on a security parameter η (which is used, for example to determine the length of random nonces). A PKI-like setting is assumed such that the public keys of parties (those for encryption and signature verification) are accessible to all parties. Encryption and signing are implemented with an encryption scheme and a digital signature scheme respectively. Pairing is implemented by some standard (efficiently invertible) encoding function. Each time a session is initialized, random values are generated for the nonces of the session.

3.1.3 Trace mapping.

The trace mapping approach attempts to link directly concrete and symbolic executions [MW04b]. The idea is to show that any concrete trace is the image of a symbolic trace (with overwhelming probability). The first result of this kind establishes such a statement for protocols that use random nonces, party identities, pairing, and asymmetric encryption under the assumption that the encryption scheme satisfies indistinguishability under chosen ciphertext attacks ($\mathsf{IND\text{-}CCA}$). We refer to a property of this type as a *mapping lemma*. Informally, the mapping lemma implies that all of the behaviors of concrete adversaries are captured by those of the symbolic adversaries. Consequently, any trace property such as authentication can be transferred from symbolic models to concrete ones: whenever a protocol satisfies (symbolically) a trace property it also satisfies the property computationally.

3.2 Extensions

3.2.1 Signatures.

The mapping lemma has then been extended in a setting with signatures and variables of sort ciphertext (to allow ciphertext forwarding) in [CW05], provided that signatures are implemented using an existentially unforgeable scheme under chosen message attacks. A similar result has been proved in [JLM05] where public key can also be sent in plaintext. They also propose a general criterion for reducing the correctness of two cryptographic schemes to the correctness of each one. This is useful when proving soundness of symbolic models when several primitives are used.

3.2.2 Hash functions.

The mapping lemma has then been extended in [CKKW06] (removing signatures) to hash function implemented in the random oracle. A weaker criterion, called HASH, for hash is proposed in [JLM06]. It is shown that the mapping lemma holds for hash functions satisfying this criteria and for asymmetric encryption (implemented with an IND-CCA encryption scheme), provided that the protocol does not have *temporary secret* (each atomic value is either initially known to the intruder or will never be revealed). The HASH criterion can be realized in the random oracle. However, it is not known whether an actual implementation realizes the HASH criterion.

3.2.3 Non-Malleable Commitment.

Galindo *et. al.* [GGvR08] have extended the mapping lemma to commitment schemes. Commitment schemes are used in protocols like zero-knowledge proofs or contract-signing. They consist of two phases: a first phase (commitment) where the principal *commits* to a message without revealing any information and a second phase (opening) where the principal reveals the message and it is possible to verify that it corresponds to the value committed during the previous phase. They abstract these primitives symbolically by introducing two functional symbols:

$$\begin{aligned} \mathsf{com}^{-}(\_) &: \mathsf{Term} \times \mathsf{Label} \to \mathsf{Com} \\ \mathsf{dec}^{-}(\_) &: \mathsf{Term} \times \mathsf{Label} \to \mathsf{Dec} \end{aligned}$$

where Com and Dec are new sorts. The corresponding deduction rules are the two following rules:

$$\frac{S \vdash m}{S \vdash \mathsf{com}^r(m)} \qquad \frac{S \vdash \mathsf{dec}^r(m)}{S \vdash m}$$

They show that the mapping lemma holds for asymmetric encryption and commitment provided that encryption is IND-CCA and that commitment is *non-malleable against chosen commitment attacks (NMC-CCA)*. NMC-CCA is a definition of security for commitment schemes that they introduce in order to prove the mapping lemma. It intuitively means that an attacker cannot produce a commitment c_2 related to another commitment $c_1 = \mathsf{com}^{r_1}(m_1)$ (where m_1 is chosen by the attacker) even if it has access to an oracle that can open commitments. This security notion can be realized: Galindo *et. al.* propose a new commitment scheme that is NMC-CCA secure.

3.2.4 Zero-knowledge proof.

A zero-knowledge proof is a message or a sequence of messages that forms a proof of a statement x (e.g. “the message within the ciphertext contains two identical nonces”) that does not reveal any information besides that x is true. Zero-knowledge proofs can be used to prove various statements. Backes *et. al* have introduced in [BMU08] an abstraction of zero-knowledge proofs for symbolic models by introducing a small logic. A $\mathsf{Formula}$ is a Boolean formula over atomic formula ZKTerm defined by:

$$\mathsf{ZKTerm} = \mathsf{ek}(\beta_i) \mid \alpha_i \mid \beta_i \mid \langle \mathsf{ZKTerm}, \mathsf{ZKTerm} \rangle \mid \{\mathsf{ek}(\beta_j)\}^{\rho_i}_{\mathsf{ZKTerm}}$$

The set of Term is enriched by a constructor $\mathsf{ZK}^R_F(\overline{r}, \overline{a}, \overline{b})$ where F is a $\mathsf{Formula}$ and $\overline{x}$ denotes $x_1, \ldots, x_n$. Intuitively, R and $\overline{r}$ are the randomness used in the formula, $\overline{a}$ represents the secret values and $\overline{b}$ the public values. $\mathsf{ZK}^R_F(\overline{r}, \overline{a}, \overline{b})$ is evaluated by replacing the α_i by the a_i, the β_i by the b_i and the ρ_i by the r_i.

Backes and Unruh extend the mapping lemma in [BU08] for this symbolic model of (non-interactive) zero-knowledge proofs by introducing a new definition for security of zero-knowledge proofs called *symbolically-sound zero-knowledge proof system*. This definition is rather involved. It is assumed that zero-knowledge proof are based on circuits and proofs that a particular circuit is satisfiable. Their new security definition requires in particular:

- *Extractability*: out of a proof for a circuit C, it is possible to extract a witness, *i.e.* a solution for C.
- *Unpredictability*: two independently produced proofs are different with overwhelming probability.
- *Extraction Zero-Knowledge*: this property is designed to prevent an adversary from building a valid proof out of previous proofs.

This definition can be realized by an existing zero-knowledge protocol defined by Groth and Ostrovsky [GO07].

3.2.5 Linking cryptographic and symbolic secrecy.

In the symbolic model, secrecy is naturally expressed as a trace property: a message is secret if it cannot be derived by the adversary. In the computational model however, typical definitions are much stronger. It is usually required that an attacker is not only unable to obtain the secret, but also *any* partial information about the secret (which is an indistinguishability notion). Typically, secrecy of a nonce N in a protocol Π is defined in cryptographic models using an experiment $\mathsf{Exp}^{\text{sec\_}b}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(\eta)$ that we describe below. The experiment is parameterized by a bit b and involves an adversary $\mathcal{A}$. The input to the experiment is a security parameter η. It starts by generating two random nonces n_0 and n_1 whose length depends on the security parameter. Then the adversary $\mathcal{A}$ interacts with the protocol Π where the nonce N has been instantiated by n_b according to the bit selection b. The adversary generates new sessions, sends messages and receives messages to and from these sessions (as prescribed by the protocol). In the end, the adversary is given n_0 and n_1 and outputs a guess d, which is the result of the experiment. The nonce N is computationally secret in Π if for every p.p.t. adversary $\mathcal{A}$ its advantage

$$\Pr\left[\mathsf{Exp}^{\text{sec\_1}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(\eta) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{sec\_0}}_{\mathsf{Exec}_{\Pi,\mathcal{A}}}(\eta) = 1\right]$$

is negligible.

3.2.6 Pairing and asymmetric encryption.

In [CW05], it is shown that, in a setting with asymmetric encryption and pairing, whenever a nonce is deemed secret using symbolic techniques, then the nonce is secret with respect to the stronger, computational definition.

3.2.7 Hash functions.

In [JLM06], soundness of symbolic secrecy is extended to hash functions under the HASH criterion and for nonces that never appear under a hash function. Transferring the usual symbolic secrecy definition to indistinguishability is indeed not possible when the target secret value appears under a hash function since, unlike ciphertexts, hashes have to be publicly verifiable, i.e., any third party can verify if a value h is the hash value corresponding to a given message m. Assume, for example, that in some protocol the hash $h = h(s)$ of some secret s is sent in clear over the network. Then, while virtually all symbolic models would conclude that s remains secret (and this is also a naive assumption often made in practice), a trivial attack works in computational models: given s, s' and h, compare h with $h(s)$ and $h(s')$, and therefore recover s. Cortier *et al.* [CKKW06] propose a new symbolic definition for nonce secrecy in protocols that use party identities, nonces, hash functions, and public key encryption. The definition is based on the concept of patterns presented in Section 2.1.1. They show that nonces that are secret according to their symbolic criterion are also secret according to a standard *computational* definition (indistinguishability). The result holds for protocols implemented with encryption schemes that satisfy standard notions of security (IND-CCA), and for hash functions modeled as random oracles. They also show decidability (NP-completeness) of the symbolic secrecy criterion (w.r.t. a bounded number of sessions).

3.3 Soundness of observational equivalence

We have just seen that “computational secrecy” can be soundly abstracted by a trace property in symbolic models, in a number of particular cases. It is not clear, however, that such a property can be expressed as a trace property in general. More generally, several security properties cannot be defined (or cannot be naturally defined) as trace properties such as e.g. anonymity, privacy related properties involved in electronic voting protocols [DKR06], or strong (also called “black-box”) simulatability [BPW07b, KT08]. These security properties are usually formalized by indistinguishability properties. There is a well-known similar notion in concurrency theory: observational equivalence, introduced by Milner and Hoare in the early 80s. Two processes P and Q are observationally equivalent, denoted by $P \sim_o Q$, if for any process O (a symbolic observer) the processes $P\|O$ and $Q\|O$ are equally able to emit on a given channel. This means that O cannot observe any difference between P and Q. It is shown in [CLC08] that computational indistinguishability in presence of an active attacker is implied by the observational equivalence of the corresponding symbolic processes, in the case of IND-CCA-2 symmetric encryption.

3.4 The simulation based approach

A different approach towards relating computational and symbolic executions of protocols relies on the concept of simulatability. Roughly, security is defined by requiring that a real system that supposedly implements some cryptographic system, is as secure as an ideal version of the protocol/primitive (which typically is secure by construction).

The concrete instance of such a simulation-based setting used in computational soundness is that of reactive simulatability/universal composability, RSIM/UC in short [Can01, BPW07a]. This setting relies on a general model for (polynomial-time) executions of interactive asynchronous programs. Related models for such executions have been defined elsewhere, with a similar goal in mind. These works include those for a probabilistic polynomial time process calculus [LMMS98, MRST01, RMST04, RMST06], the model of Canetti [Can01], more rigorously formalized and further refined using the framework of Task Structured Probabilistic Input/Output Automata (Task PIOAs) [CCK$^+$06].

The details of such models are outside the scope of this survey. We refer to the work of Küsters *et* al. for a detailed analysis and comparison of the different existent frameworks [KDMR08].

As sketched above, the definition of security involved an ideal system maintained by a trusted host TH. The real system is given by a set of interactive machines $\mathsf{M_i}$, one for each user i. Both systems interact with an *environment* Env which should be thought of as protocols (or users) that provide input/obtain output to/from the system that is being analyzed. Furthermore, the interaction also involves an *adversary* Adv. In the real world, the adversary interacts directly with the system (i.e. it communicates directly with the machines $M_1, M_2, \ldots, M_n$). In the ideal world, the communication between the adversary and the trusted host is mediated by a *simulator* Sim. The two different setups are described in Figure 2.

We write (rather informally) $\mathsf{TH} \mid \mathsf{Sim} \mid \mathsf{Adv} \mid \mathsf{Env}$ for the result of the execution of the ideal system, i.e. for the output of the environment. Similarly, we write $(M_1||M_2||\ldots||M_n) \mid \mathsf{Adv} \mid \mathsf{Env}$ for the result of the real execution.

We say that $M_1, M_2, \ldots, M_n$ RSIM/UC implements the system described by TH and we write $M_1||M_2||\ldots||M_n \leq^{RSIM} \mathsf{TH}$ if there exists a simulator Sim (that mediates the interaction between the adversary and the TH) such that no combination of environment and adversary can determine whether the interaction takes place in the ideal world, or in the real world:

$$(\exists \mathsf{Sim})(\forall \mathsf{Adv})(\forall \mathsf{Env})\ (\mathsf{TH} \mid \mathsf{Sim} \mid \mathsf{Adv} \mid \mathsf{Env}) \cong (M_1||M_2||\ldots||M_n \mid \mathsf{Adv} \mid \mathsf{Env})$$

In the above, the notation $\cong$ represents some version of indistinguishability (i.e. perfect equality, statistical closeness, or computational indistinguishability of distributions).

3.4.1 Preservation of integrity properties.

The intuition behind the above definition is that the protocol defined by machines $M_1, M_2, \ldots, M_n$ does not leak any more information to an adversary than the ideal version defined by TH, and thus the former is as secure as the latter. Usually, the security of the ideal system can be assessed by simple inspection, or

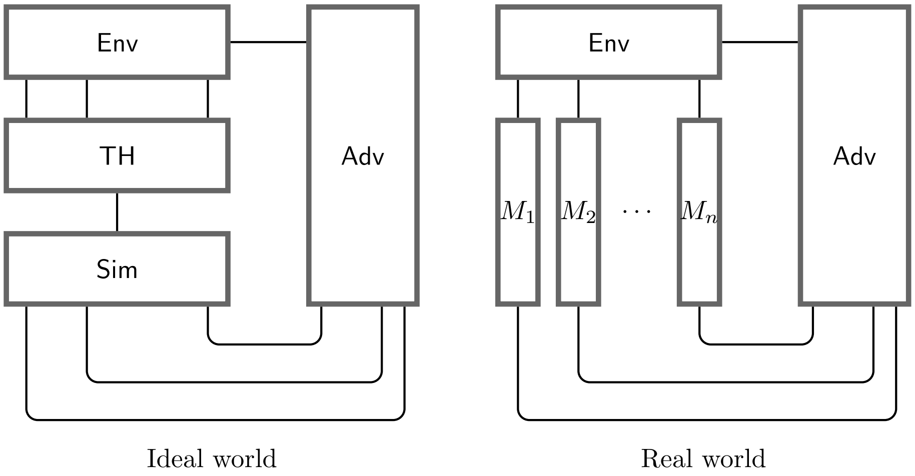

Figure 2: The RSIM setting: in the ideal world the interaction is between a trusted host TH, an adversary Adv, a simulator Sim, and environment Env. In the ideal world, the interaction is between an actual implementation of the protocol by machines $M_1, M_2, \ldots, M_n$, and environment Env and an adversary Adv. The machines $M_1, M_2, \ldots, M_n$ RSIM/UC implement the system defined by TH, if $(\exists \mathsf{Sim})(\forall \mathsf{Adv})(\forall \mathsf{Env})$ $(\mathsf{TH} \mid \mathsf{Sim} \mid \mathsf{Adv} \mid \mathsf{Env}) \cong (M_1 || M_2 || \ldots || M_n \mid \mathsf{Adv} \mid \mathsf{Env})$

proved through some simple means, and the security of the real implementation thus follows.

Importantly, security defined in the sense above is composable: if $M = M_1 || M_2 || \ldots || M_n$ is such that $M \leq^{RSIM}$ TH then M can be replaced in a system by the combination TH | Sim without changing the behavior of the system. In particular, properties of the ideal system should also be satisfied by the implementation [BJP02].

3.5 A composable cryptographic library

The framework for reactive simulatability sketched in the previous section has been used by Backes, Pfitzmann, and Waidner to obtain computational soundness results. They define an ideal cryptographic library $\mathsf{Lib}^{\mathsf{ideal}}$ which offers an interface through which programs can manipulate data. Commands that can be passed to the library include the ability to generate nonces and cryptographic keys, to encrypt and decrypt messages, to generate and verify signatures etc. The internal workings of the library, i.e. the semantics of all of the commands is entirely deterministic, and is based on the ideas behind Dolev-Yao models. Roughly speaking, $\mathsf{Lib}^{\mathsf{ideal}}$ maintains an internal database of *symbolic terms* which the programs can manipulate via handles to these terms which it obtains from the library. A party would be able to obtain the plaintext in an encryption only if it has handles to both the term that represents the encryption and to the appropriate decryption key. Importantly, since the final goal is to relate $\mathsf{Lib}^{\mathsf{ideal}}$ with a real implementation, the library needs to keep track of all the various pieces of information which real cryptographic primitives may leak. The reason is that the environment would be able to tell the difference between the real and the ideal executions by observing such leaks. Typical examples include the length of encrypted plaintexts, as well as the length that corresponds to the ideal terms in a real instantiation. In the context of the reactive simulatability setting presented in the previous section, $\mathsf{Lib}^{\mathsf{ideal}}$ plays the role of the ideal system, i.e. that of TH.

To obtain computational soundness it is sufficient to exhibit a real implementation, i.e. a library $\mathsf{Lib}^{\mathsf{real}}$ implemented with actual cryptographic primitives, which offers the same interface as $\mathsf{Lib}^{\mathsf{ideal}}$ such that $\mathsf{Lib}^{\mathsf{real}} \leq^{RSIM} \mathsf{Lib}^{\mathsf{ideal}}$. In [BPW03a] Backes, Pfitzmann, and Waidner exhibit an ideal, and a real library that are related as described above. The cryptographic operations considered in [BPW03a] are nonce generation, asymmetric encryption, and digital signatures. The main result is that $\mathsf{Lib}^{\mathsf{real}} \leq^{RSIM} \mathsf{Lib}^{\mathsf{ideal}}$ provided that the digital signature scheme is memory-less (a signature does not leak any information about previous signatures) and existentially unforgeable under chosen message attack and the encryption scheme is $\mathsf{IND\text{-}CCA}$ secure.

3.5.1 Message authentication codes(MACs).

The above result had been extended to a library that includes message authentication codes [BPW03b]. The security condition under which the desired result holds is that the MAC used in the implementation of the real library is existentially unforgeable under chosen message attacks. In addition, given a tag created by the MAC scheme, it must be possible to fully recover the message to which it corresponds, and it must be possible to determine whether two tags have been created under the same key or not, even if one does not posses the key. Finally, the protocol where the MAC is used has to append a random nonce to the message which is MACed.

3.5.2 Symmetric encryption.

Subsequent work introduced symmetric encryption among the primitives that the simulatable cryptographic library offers [BP04b]. A RSIM/UC relation between the resulting ideal system and a concrete realization requires several restrictions on the way the library is used by the surrounding protocol. We only list some of them. First, they of course forbid encryption cycles by assuming a key hierarchy based on the order in which keys are used for encryption for the first time. Second, an important issue with symmetric encryption is the so-called *commitment problem* which appears when a key is used for encryption and is later revealed. Thus it is assumed here that keys are never revealed after being used. Third, they assume authenticated encryption schemes (i.e. the adversary is not able to compute a ciphertext that can be validly decrypted with an unknown specific key) and they assume that ciphertexts are tagged with key identifiers.

3.5.3 Transferring secrecy properties via RSIM/UC.

As discussed in previous sections, in symbolic models secrecy properties can be expressed as trace properties, while in computational models they are not. Thus, transference of secrecy properties does not follow from the preservation theorems proved for integrity properties. The result that symbolic secrecy implies computational secrecy for the cryptographic library described above appears in [BP05b]. The result holds for payload data, and for symmetric keys (which had not been used for encryption or authentication), i.e. nonces. The precise formulation of computational secrecy is indistinguishability based, and is similar to the one for the trace mapping approach.

3.5.4 Impossibility of results of RSIM/UC soundness.

The strong relation imposed between the ideal and the real system by the RSIM/UC relation leads to several impossibility results. In [BP05a], Backes and Pfitzmann offer impossibility results for an ideal library that contains a model for XOR. The results are rather general, in that they are not for a fixed abstraction of XORs. Instead, they show that if such a library is rich enough to allow the specification of some simple protocols where secrecy of some piece of data is desired, then a RSIM/UC concrete realization would imply that the library itself is not abstract: using the library one is able to compute concrete cryptographic functions, e.g. signatures. To complement the impossibility results, the authors show soundness for the case of passive adversaries.

A second impossibility result, reminiscent of the restrictions imposed for the case of symmetric encryption, has been obtained for the case of hashes [BPW06]. The authors show several impossibility results for various restrictions on the class of protocols that one is able to specify. The impossibility results hold for essentially all natural abstractions of (one-way) hash functions.

3.5.5 Key Dependent Message security.

As observed as early as the initial work of Abadi and Rogaway, settings where key-dependent encryption occur either in normal executions of protocols, or due to the malicious activities of the adversary pose a real problem to computational soundness, especially when encryption cycles occur. The problem is that although such settings are smoothly treated via symbolic methods, in computational models it may be the case that encryption breaks completely. Two possible work-arounds the problem is to either prohibit the occurrence of such situations (e.g. via syntactic restrictions on the protocols that are analyzed or via checking symbolically that an adversary cannot obtain key cycles while interacting with the protocols [CZ06]) or to require that in computational settings encryption is stronger and does not break even when used in such more esoteric ways.

The second approach has recently been taken by Backes, Pfitzmann, and Scedrov [BPS08]. They build on the earlier work of Black, Rogaway, and Shrimpton [BRS02] and put forth a security notion for encryption that takes into account key-dependent message attacks. They show that the notion can be achieved in the random oracle model, and show that the notion is indeed sufficient to obtain soundness for the BPW cryptographic library, even when encryption cycles occur.

3.5.6 Simulatability implies trace mapping.

The first work that investigates the relation between the trace mapping approach and the one based on reactive simulatability is by Backes, Dürmuth, and Küsters [BDK07]. They show that if two systems are related in the sense of the latter, then a relation in the sense of trace mapping also holds.

3.5.7 Case studies.

The applicability of the above described cryptographic library has been illustrated on a number of case studies. In [BP03, BP04a], Backes et al. give a cryptographically sound proof that the Needham-Schroeder protocol satisfies authentication using the ideal library. More precisely, they show that an honest participant A only successfully terminates a protocol with an honest participant B if B has indeed started a protocol with A. Backes et al. have also analyzed the Otway-Rees protocol [Bac04], which relies on symmetric encryption. Therefore the security proof also needs to show the absence of the above discussed commitment problem. Moreover, a confidentiality property of the established key is shown. The confidentiality property shown here is not cryptographic key secrecy, but ensures that an adversary can never obtain a handle to that key, which is close to deducibility in symbolic models.

In [BP06a], the authors illustrate the use of their library for showing cryptographic key secrecy, relying on their secrecy transferring result [BP05b] described above. They study the Yahalom protocol. The first remark is that cryptographic key secrecy, i.e. indistinguishability of a real and a random key, is not guaranteed by the Yahalom protocol as it ends with a key confirmation. A slightly simplified version, omitting the last message, is then shown to guarantee key secrecy.

The approach is also illustrated on more complex protocols. In [BD05], a correctness proof of an electronic payment protocol, a slight simplification of the 3KP protocol, is given. In [BMP$^+$06], a web service protocol, the WS-Reliable Messaging scenario is analyzed and in [BCJ$^+$06] security proofs are given for the Kerberos 5 protocol.

The above discussed case studies rely on hand proofs, but it is argued that the proofs in the ideal system are in the scope of existing automated tools.

3.6 Towards automated proofs of simulatability

Several results study the automation of proof of simulatability in the context of the simulatable cryptographic library of Backes, Pfitzmann and Waidner. Laud [Lau05] proposes a type system for checking secrecy of messages handled by protocols. He defines a language for cryptographic protocols, similar to the spi-calculus [AG97] tailored to the BPW cryptographic library for symmetric and asymmetric encryption. He presents a type system such that if a protocol types then it preserves the secrecy of the messages given to it by the users. In [BL06], Backes and Laud propose a mechanized approach (implemented as a tool) for proving secrecy of payloads data in cryptographic protocols modeled in the framework of [Lau05], for an unbounded number of sessions using a typing system.

In [SBB$^+$06, SB08a, SB08b], Sprenger *et al.* formalize the BPW model in the theorem prover Isabelle/HOL for public-key encryption. Since this model is too complex to directly analyze protocols, they propose several cryptographically sound abstractions of the initial model, providing a proof of the soundness of the abstractions within the Isabelle/HOL prover. As a case study, they show how the more abstract models can be used for proving the security of the Needham-Schroeder-Lowe protocol [Low96].

Canetti and Herzog [CH06] define a mapping between protocols that use public key encryption, in the UC-framework, and symbolic protocols such that the concrete protocol realizes mutual authentication functionality if and only if its translation fulfills the symbolic mutual authentication criteria. For the key exchange functionality, they propose a new symbolic criteria such that a concrete protocol realizes the key exchange functionality if and only if its translation fulfills the new symbolic criteria. Then they apply an existing tool (ProVerif [Bla01]) to verify whether or not the key exchange criteria is satisfied by known protocols.

4 Computational sound proof systems and logics

In this section we consider cryptographically sound proof systems. These soundness results are of a rather different flavor. They do not show a general mapping lemma ensuring that any computational trace corresponds to a symbolic trace with overwhelming probability. They rather prove that any property which can be proved in a symbolic setting also holds in the computational setting with overwhelming probability.

4.1 Computational Protocol Composition Logic

The Protocol Composition Logic, PCL for short, is a Floyd-Hoare like logic for proving properties of security protocols in a compositional way. The logic includes a modal operator $\psi[P]_X\varphi$ which intuitively means that if the pre-condition ψ holds and participant X executes protocol actions P then the post-condition φ will hold. Protocols are described using a simple calculus for specifying roles. A role is a sequence of actions including new nonce generation, send, receive and application of cryptographic functions. The logic comes with a number of axioms and proof rules which implicitly assume the presence of a Dolev-Yao like active adversary. As an example, the two proof rules

$$\frac{\varphi \quad \varphi \Rightarrow \psi}{\psi} \qquad\qquad \frac{\psi[P_1]_X\theta \quad \theta[P_2]_X\varphi}{\psi[P_1P_2]_X\varphi}$$

allow sequential composition, given that the post-condition of a first protocol implies the pre-condition of a second protocol. The logic also allows assume-guarantee-like parallel composition: provided that invariants of protocol P_1 are preserved by protocol P_2 and vice-versa, properties are preserved by the parallel composition of P_1 and P_2. One may note that composition is *conditional* in PCL as opposed to the *universal* composability described in Section 3.4. Giving a complete account of the logic is beyond the scope of this paper. A survey on PCL and the numerous case studies carried out in this framework can be found in [DDMR07].

In [DDM$^+$05], Datta et al. define *Computational PCL*, CPCL for short, by giving a computational semantics for a variant of PCL. The protocols are hence executed in the presence of an arbitrary PPT adversary. In [DDM$^+$05] the only cryptographic primitive is asymmetric encryption and the logic is equipped with two new predicates, Indist and Possess. Intuitively, Indist expresses that a nonce is computationally indistinguishable from a random nonce, for an arbitrary active adversary that is allowed to interact with the protocol. Possess is used to model that a bitstring corresponding to a given term cannot be built by the adversary using Dolev-Yao deduction rules, i.e. it supposes a fixed algorithm for constructing this bitstring rather than an arbitrary PPT algorithm. The main result of the paper is a soundness result showing that if a formula can be deduced using the axioms and proof rules and if the encryption scheme is IND-CCA-2 secure, then the formula holds with overwhelming probability in the computational semantics, i.e. in the presence of an arbitrary PPT adversary. The logic and soundness result has been substantially extended in the following.

In [DDMW06] the logic is extended for proving the security of key exchange protocols. As already noted, cryptographic key secrecy, stating that a key is indistinguishable from a random key, is too strong if the protocol contains a key confirmation step or if the key is to be used by another protocol. Therefore, a new, weaker security property called *key usability* is presented. Intuitively, key usability holds if the established key can be used safely afterwards. The definition is therefore parameterized by the intended use of the key. More precisely, the property is formalized by an experiment involving a two-stage adversary $(\mathcal{A}_e, \mathcal{A}_c)$: in the first phase $\mathcal{A}_e$ interacts with the key exchange protocol; in the second phase the adversary $\mathcal{A}_c$ receives state information of $\mathcal{A}_e$ and plays a security game, e.g. IND-CPA. The definition is illustrated by showing the security of the ISO-9798-3 key exchange protocol, followed by a secure session using the exchanged key. The secure session requires the use of an IND-CPA secure symmetric encryption key. The case study also required the extension of the logic and soundness theorem to symmetric encryption, Diffie-Hellman exponentiation and secure signature schemes, requiring respectively IND-CPA secure symmetric encryption, the decisional Diffie-Hellman assumption and a CMA secure signature scheme.

In [RDDM07], Roy et al. define a trace-based property, called *secretive*, which is suitable for inductive and compositional proofs. This property guarantees a black-box reduction from attacks on the protocol to attacks on the underlying primitives. Moreover this property implies computational secrecy properties (which is not a trace based property) including key indistinguishability and key usability. The result is illustrated by giving formal proofs of computational authentication and secrecy of Kerberos V5. In [RDM08], Roy et al. further refine the logic for studying Diffie-Hellman based key exchange protocols. The techniques are illustrated on the initial authentication of Kerberos V5 and IKEv2, which is the IPSEC key exchange standard.

Gupta and Shmatikov [GS05] also study a variant of PCL tailored to the analysis of key exchange protocols. The fragment they consider only contains signatures and a restricted form of Diffie-Hellman exponentiation, which requires the exponentiations to be signed. The security property they consider differs from the work discussed just above. Gupta and Shmatikov consider indistinguishability of the key exchange protocol and an ideal key exchange functionality together with a simulator. They define a symbolic criteria which under standard security definitions (DDH and CMA) implies that for any computational adversary, there exists a simulator, such that the transcripts of the adversary interacting with the real and the ideal system are indistinguishable.

The method is illustrated on the authenticated Diffie-Hellman key exchange protocol. In [GS06], they refine their results to allow *adaptive corruption* of long-term secrets (but not *strong* adaptive corruption which reveals the entire internal state of corrupted participants, rather than just the long term secret).

4.2 Static analysis techniques

In [Lau04], Laud presents static analysis techniques which are computationally sound for protocols that use symmetric encryption in order to show computational secrecy properties. The protocols are described in a basic programming language which allows sending and receiving messages and application of encryption and decryption functions, manipulation of tuples, random number generation and equality testing. The technique consists in a protocol transformation, which is correct in the sense that an incorrect protocol is never transformed into a correct protocol. To achieve correctness the symmetric encryption scheme is supposed to be IND-CCA and to provide ciphertext integrity. The protocol transformation mainly consists in removing unreachable code, replacing bitstrings by formal terms and encryptions by encryptions of sequences of 0s. The resulting protocol can then be analyzed using symbolic information flow techniques. These results have been further extended in [LT05] to protocols that use digital signatures.

5 Computationally sound implementation of higher level symbolic constructs

The work that we discuss in this section is conceptually close to computational soundness. All of these papers relate abstract symbolic languages and their concrete implementation in such a way that reasoning at the abstract layer yields meaningful results about the actual implementation. Unlike the papers discussed in previous sections, the abstract languages that are considered do not deal with cryptographic primitives explicitly, but use constructs or concepts that are security related. Cryptography is then used to ensure that the implementation reflects the security concerns captured at the higher level of abstraction.

5.0.1 Secure channels.

Adaõ and Fournet [AF06] introduce a process calculus-based language which has, as part of the core set of operations, built-in constructs that allow parties to 1) make use of certificates issued by authorities and 2) communicate on authenticated channels. At this level of abstraction, the use of cryptography is transparent, and the desired security properties of these constructs are captured by their semantics. In the next step the authors give an implementation of the two high-level constructs described above; both implementations are based on digital signatures and are rather straightforward. The authors prove a soundness result that relate the two levels of abstraction provided that the digital signature scheme used in the implementation is universally unforgeable under chosen-message attacks [GMR88] and that the semantics of the abstract level is preserved by the implementation. It is worth noting that the paper only studies authentication, and is not concerned with secrecy properties.

Abadi, Corin, and Fournet [ACF06] define a process calculus which allows parties to create and use secure (that is, secret and authenticated) channels. The desired intuitive security properties are captured via the semantics that they attach to processes specified in this language. A standard notion of secrecy can be defined at this level, and a type system is used to reason about it. Next, the authors describe a lower-level language where cryptography occurs as part of the core operations that can be performed and give a distributed implementation for the abstract processes. Interestingly, the implementation and the results of the paper rely on a previous computational soundness result. Indeed, the low-level implementation language is essentially the one introduced by Laud [Lau05], which we discuss in Section 3.6. Recall that programs written in this language that are typable preserve the secrecy of the messages sent by the honest parties. The result of Abadi, Corin, and Fournet build on the above. They prove that a typable process is translated into a typable program. It then follows, by Laud's soundness result [Lau05], that data which is secret at the abstract level is also secret at the level of the concrete implementation (according to a computational notion of secrecy).

5.0.2 Information flow.

Fournet and Rezk [FR08] investigate the use of cryptography for enforcing secure information flow for both confidentiality and integrity. In more details their result is as follows. They first give a simple programming language with an associated language for specifying information flow policies. Satisfaction of such policies can be checked using a type system that they also design. Next, they give a lower-level implementation language which includes encryption and digital signing as part of the primitives that can be used. The type system is such that programs that type-check do not have undesired information flow, computationally. The main result of the paper uses a typed translation of abstract programs to concrete ones. They show that if the source program is typable then its translation is also typable. They conclude that the implementation satisfies non-interference against probabilistic polynomial time adversaries.

6 The direct approach: formal cryptographic proofs

Bruno Blanchet [Bla06, Bla07b] has designed a mechanized prover, named CryptoVerif, for security properties of cryptographic protocols. In contrast to most previous approaches, the tool does not rely on soundness results for symbolic model but directly automate the proofs made in cryptography, based on sequences of games. The security property of protocol is specified as a game and is step by step reduced to the game defining the security of the cryptographic primitives. CryptoVerif handles shared-key and public-key encryption, signatures, message authentication codes, and hash functions. It provides a general strategy for transforming games. In case the strategy fails, it is possible to use an interactive mode where the user specifies by hand which transformation should be used.

The first version of CryptoVerif [Bla06, Bla07b] was designed for secrecy property. It has then been extended for proving correspondence assertions [Bla07a]. Correspondence assertions are useful for specify properties like authentication. The tool has been tested on several protocols from the literature (e.g. Otway-Rees, Needham-Schroeder shared-key, Denning-Sacco public-key). It has been recently used to analyze Kerberos 5, a full industrial protocol [BJST08]. The CryptoVerif tool can also be used not only to automate security proofs of protocols but also to automate security proofs of cryptographic primitives, reducing their security to standard cryptographic assumptions [BP06b]. To illustrate their technique, they show in particular that the Full-Domain Hash signature scheme enjoys unforgeability under chosen-message attacks (UF-CMA) under the assumption of (trapdoor) one-wayness of some permutations.

There have also been symbolic proofs of security for cryptographic primitives. In [BCT04] Barthe et al. formalize the random oracle model and the generic model in the proof assistant COQ. This formalization is used by Tarento [Tar05] to machine-check a security proof of signature schemes against forgery attacks for arbitrary generic adversaries. In the same vein, Courant et al. [CDCEL08] present an (incomplete) automated procedure for analyzing generic asymmetric encryption schemes in the random oracle model. More precisely, they define a programming language to specify generic encryption algorithms, i.e. encryption algorithms that rely on generic one-way functions and hash functions. On top of this language they define a Hoare logic to establish invariants that allow the proof of IND-CPA security. They also present a syntactic condition which guarantees plaintext-awareness, which together with IND-CPA security implies IND-CCA-2 security. Although not complete the tool has been successfully applied to the construction of Bellare-Rogaway 1993, of Pointcheval at PKC'2000 and REACT.

7 Conclusion

In this paper we survey existing results that aim to bridge the gap between the two approaches used in security analysis. The direct approach is rather recent and work in this direction is in full swing. Currently, existent formalisms can tackle various game transformation based proofs. Two important directions that need to still be explored are game-based transformations based on rewinding (e.g. the techniques used in proving Schnorr signature schemes) and those based on hybrid arguments, where the number of hybrids depends on the security parameter.

On the computational soundness side, there are also many questions still open. First, several primitives appear to be difficult to abstract (soundly) in symbolic models. An important example is that of hash functions. In symbolic models hash functions are usually represented by a free symbol (usually denoted by h). This formalization seems to account for very strong security properties that cryptographic hash functions do not necessarily have. Another example is that of symmetric encryption where symbolic models do not seem to capture accurately the associated cryptographic behaviors.

Finally, most soundness results require strong security assumptions on the primitives (e.g. IND-CCA-2 encryption in the active case), and this may seem to be unavoidable. Indeed, it has been shown that weaker but still standard assumptions may indeed compromise security [War05]. Nevertheless, in practice it is not always possible to use strong secure primitives due to legacy or efficiency reasons. For example, one might need to use deterministic encryption, in which case the encryption scheme cannot be IND-CCA or even IND-CPA. It would be particularly interesting to see if it is possible to obtain computational soundness for weaker security assumptions on the implementation of the primitives.

References

[ABHS05] Pedro Adão, Gergei Bana, Jonathan Herzog, and Andre Scedrov. Soundness of formal encryption in the presence of key-cycles. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *LNCS*, pages 374–396, 2005.

[ABS05] Pedro Adão, Gergei Bana, and Andre Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 170–184, 2005.

[ABW06] Martín Abadi, Mathieu Baudet, and Bogdan Warinschi. Guessing attacks and the computational soundness of static equivalence. In *Proc. 9th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'06)*, volume 3921 of *LNCS*, 2006.

[ACF06] Martín Abadi, Ricardo Corin, and Cédric Fournet. Computational secrecy by typing for the pi calculus. In *Programming Languages and Systems, 4th Asian Symposium, (APLAS'06)*, volume 4279 of *LNCS*, pages 253–269. Springer, 2006.

[AF06] Pedro Adão and Cédric Fournet. Cryptographically sound implementations for communicating processes. In *Automata, Languages and Programming, 33rd International Colloquium (ICALP'06)*, volume 4052 of *LNCS*, pages 83–94. Springer, 2006.

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. of the 4th ACM Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.

[AHS06] Aslan Askarov, Daniel Hedin, and Andrei Sabelfeld. Cryptographically-masked flows. In *Proc. 13th International Static Analysis Symposium (SAS'06)*, volume 4134 of *LNCS*, pages 353–369, 2006.

[AR00] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Proc. 1st IFIP International Conference on Theoretical Computer Science (IFIP–TCS'00)*, volume 1872 of *LNCS*, pages 3–22, 2000.

[AR02] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[AW05a] Martín Abadi and Bogdan Warinschi. Password-based encryption analyzed. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 664–676. Springer, 2005.

[AW05b] Martín Abadi and Bogdan Warinschi. Security analysis of cryptographically controlled access to xml documents. In *Proc. 24th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS'05)*, pages 108–117. ACM Press, 2005.

[AW08] Martín Abadi and Bogdan Warinschi. Security analysis of cryptographically controlled access to xml documents. *J. ACM*, 55(2):1–29, 2008.

[Bac04] Michael Backes. A cryptographically sound Dolev-Yao style security proof of the Otway-Rees protocol. In *Proc. 9th European Symposium on Research in Computer Security (ESORICS'04)*, volume 3193 of *LNCS*, pages 89–108, 2004.

[BCJ$^+$06] Michael Backes, Iliano Cervesato, Aaron D. Jaggard, Andre Scedrov, and Joe-Kai Tsay. Cryptographically sound security proofs for basic and public-key kerberos. In *Proc. 11th European Symposium on Research in Computer Security(ESORICS'06)*, volume 4189 of *LNCS*, pages 362–383. Springer, 2006.

[BCK05] Mathieu Baudet, Véronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. In *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, volume 3580 of *LNCS*, pages 652–663. Springer, 2005.

[BCK09] Mathieu Baudet, Véronique Cortier, and Steve Kremer. Computationally sound implementations of equational theories against passive adversaries. *Information and Computation*, 207(4):496–520, April 2009.

[BCT04] Gilles Barthe, Jan Cederquist, and Sabrina Tarento. A machine-checked formalization of the generic model and the random oracle model. In *Proc. 2nd International Joint Conference on Automated Reasoning (IJCAR'04)*, volume 3097 of *Lecture Notes in Artificial Intelligence*, pages 385–399. Springer-Verlag, 2004.

[BD05] Michael Backes and Markus Duermuth. A cryptographically sound Dolev-Yao style security proof of an electronic payment system. In *Proc. 18th IEEE Computer Security Foundations Workshop (CSFW'05)*, pages 78–93, 2005.

[BDK07] Michael Backes, Markus Dürmuth, and Ralf Küsters. On simulatability soundness and mapping soundness of symbolic cryptography. In *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *LNCS*, pages 108–120, New Delhi, India, December 2007. Springer.

[BHHO08] Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *LNCS*, pages 108–125. Springer, 2008.

[BJP02] Michael Backes, Christian Jacobi, and Birgit Pfitzmann. Deriving cryptographically sound implementations using composition and formally verified bisimulation. In *International Symposium of Formal Methods Europe*, volume 2381 of *LNCS*, pages 310–329, 2002.

[BJST08] Bruno Blanchet, Aaron D. Jaggard, Andre Scedrov, and Joe-Kai Tsay. Computationally sound mechanized proofs for basic and public-key kerberos. In *ACM Symposium on Information, Computer and Communications Security (ASIACCS'08)*, pages 87–99, Tokyo, Japan, March 2008. ACM.

[BL06] Michael Backes and Peeter Laud. Computationally sound secrecy proofs by mechanized flow analysis. In *Proc. 13th ACM Conference on Computer and Communications Security (CCS'06)*, pages 370–379, Alexandria, VA, USA, 2006.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *Proc. of the 14th Computer Security Foundations Workshop (CSFW'01)*, pages 82–96. IEEE Computer Society Press, 2001.

[Bla06] Bruno Blanchet. A computationally sound mechanized prover for security protocols. In *IEEE Symposium on Security and Privacy*, pages 140–154, Oakland, California, May 2006. IEEE Computer Society Press.

[Bla07a] Bruno Blanchet. Computationally sound mechanized proofs of correspondence assertions. In *20th IEEE Computer Security Foundations Symposium (CSF'07)*, pages 97–111, Venice, Italy, July 2007. IEEE.

[Bla07b] Bruno Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 2007.

[BLMW07] Emmanuel Bresson, Yassine Lakhnech, Laurent Mazaré, and Bogdan Warinschi. A generalization of ddh with applications to protocol analysis and computational soundness. In *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *LNCS*, pages 482–499. Springer, 2007.

[BMP$^+$06] Michael Backes, Sebastian Moedersheim, Birgit Pfitzmann, , and Luca Vigano. Symbolic and cryptographic analysis of the secure ws-reliablemessaging scenario. In *Proc. 9th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'06)*, volume 3921 of *LNCS*, 2006.

[BMS06] Gergei Bana, Payman Mohassel, and Till Stegers. The computational soundness of formal indistinguishability and static equivalence. In *Proc. 11th Asian Computing Science Conference (ASIAN'06)*, volume 4435 of *LNCS*, pages 182–196. Springer, 2006.

[BMU08] Michael Backes, Matteo Maffei, and Dominique Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *Proceedings of 29th IEEE Symposium on Security and Privacy*, May 2008.

[BP03] Michael Backes and Birgit Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. In *Proc. 23rd Conference on Foundations of Software Technology and Theoretical Computer Science (FST–TCS'03)*, volume 2914 of *LNCS*, pages 1–12, 2003.

[BP04a] Michael Backes and Birgit Pfitzmann. A cryptographically sound security proof of the Needham-Schroeder-Lowe public-key protocol. *Journal on Selected Areas in Communications*, 22(10):2075–2086, 2004.

[BP04b] Michael Backes and Birgit Pfitzmann. Symmetric encryption in a simulatable Dolev-Yao style cryptographic library. In *Proc. 17th IEEE Computer Science Foundations Workshop (CSFW'04)*, pages 204–218, 2004.

[BP05a] Michael Backes and Birgit Pfitzmann. Limits of the cryptographic realization of Dolev-Yao-style xor. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *LNCS*, pages 336–354, 2005.

[BP05b] Michael Backes and Birgit Pfitzmann. Relating symbolic and computational secrecy. *Transactions on Dependable and Secure Computing*, 2(2):109–123, 2005.

[BP06a] Michael Backes and Birgit Pfitzmann. On the cryptographic key secrecy of the strengthened yahalom protocol. In *Proc. 21st IFIP International Information Security Conference (SEC'06)*, 2006.

[BP06b] Bruno Blanchet and David Pointcheval. Automated security proofs with sequences of games. In *CRYPTO'06*, volume 4117 of *Lecture Notes on Computer Science*, pages 537–554, Santa Barbara, CA, August 2006. Springer Verlag.

[BPS08] Michael Backes, Birgit Pfitzmann, and Andre Scedrov. Key-dependent message security under active attacks - BRSIM/UC-soundness of symbolic encryption with key cycles. *Journal of Computer Security (JCS)*, 2008.

[BPW03a] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In *Proc. 10th ACM Conference on Computer and Communications Security (CCS'03)*, 2003.

[BPW03b] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Symmetric authentication within simulatable cryptographic library. In *Proc. 8th European Symposium on Research in Computer Security (ESORICS'03)*, LNCS, pages 271–290, 2003.

[BPW06] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Limits of the reactive simulatability/uc of Dolev-Yao models with hashes. In *Proceedings of 11th European Symposium on Research in Computer Security(ESORICS)*, volume 4189 of *LNCS*, pages 404–423. Springer, September 2006. Preprint on IACR ePrint 2006/068.

[BPW07a] Michael Backes, Birgit Pfitzmann, and Michael Waidner. The reactive simulatability framework. *Information & Computation*, 205(12):1685–1720, 2007.

[BPW07b] Michael Backes, Birgit Pfitzmann, and Michael Waidner. The reactive simulatability (RSIM) framework for asynchronous systems. *Information and Computation*, 205(12):1685–1720, 2007.

[BRS02] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In *Selected Areas in Cryptography 2002 (SAC '02)*, volume 2595 of *LNCS*, pages 62–75, St. John's, Newfoundland, Canada, 2002. Springer-Verlag.

[BU08] Michael Backes and Dominique Unruh. Computational soundness of symbolic zero-knowledge proofs against active attackers. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF'08)*, pages 255–269, Pittsburgh, PA, USA, June 2008. IEEE Computer Society Press.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols (extended abstract). In *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01)*, pages 136–147, 2001.

[CCK$^+$06] Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, and Roberto Segala. Time-bounded task-pioas: A framework for analyzing security protocols. In *Proceedings of the 20th International Symposium on Distributed Computing*, pages 238–253, 2006.

[CDCEL08] Judicaël Courant, Marion Daubignard, Pascal Lafourcade Cristian Ene, and Yassine Lahknech. Towards automated proofs for asymmetric encryption schemes in the random oracle model. In *Proc. 15th ACM Conference on Computer and Communications Security, (CCS'08)*, Alexandria, USA, 2008. To appear.

[CEL07] Judicaël Courant, Cristian Ene, and Yassine Lakhnech. Computationally sound typing for non-interference: The case of deterministic encryption. In *Proceedings of the 27th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'07)*, volume 4855 of *LNCS*, pages 364–375, New Delhi, India, December 2007. Springer.

[CH06] Ran Canetti and Jonathan Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols (extended abstract). In *Proc. 3rd Theory of Cryptography Conference (TCC'06)*, volume 3876 of *LNCS*, pages 380–403. Springer, 2006.

[CHW06] Véronique Cortier, Heinrich Hördegen, and Bogdan Warinschi. Explicit randomness is not necessary when modeling probabilistic encryption. In *Workshop on Information and Computer Security (ICS 2006)*, Timisoara, Romania, September 2006.

[CKKW06] Véronique Cortier, Steve Kremer, Ralf Küsters, and Bogdan Warinschi. Computationally sound symbolic secrecy in the presence of hash functions. In *Proceedings of the 26th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'06)*, volume 4337 of *LNCS*, pages 176–187, Kolkata, India, 2006. Springer.

[CLC08] Hubert Comon-Lundh and Véronique Cortier. Computational soundness of observational equivalence. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, Alexandria, Virginia, USA, October 2008. ACM Press.

[CW05] Véronique Cortier and Bogdan Warinschi. Computationally sound, automated proofs for security protocols. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 157–171, Edinburgh, UK, 2005. Springer.

[CZ06] Véronique Cortier and Eugen Zălinescu. Deciding key cycles for security protocols. In *Proc. of the 13th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06)*, volume 4246 of *Lecture Notes in Artificial Intelligence*, pages 317–331, Phnom Penh, Cambodia, November 2006. Springer.

[DDM$^+$05] Anupam Datta, Ante Derek, John C. Mitchell, Vitaly Shmatikov, and Mathieu Turuani. Probabilistic Polynomial-time Semantics for a Protocol Security Logic. In *Proc. of 32nd International Colloquium on Automata, Languages and Programming, ICALP*, volume 3580 of *LNCS*, pages 16–29. Springer, 2005. Lisboa, Portugal.

[DDMR07] Anupam Datta, Ante Derek, John C. Mitchell, and Arnab Roy. Protocol composition logic (PCL). In *Computation, Meaning and Logic: Articles dedicated to Gordon Plotkin*, Electronic Notes in Theoretical Computer Science, 2007.

[DDMW06] Anupam Datta, Ante Derek, John C. Mitchell, and Bogdan Warinschi. Computationally sound compositional logic for key exchange protocols. In *Proceedings of 19th IEEE Computer Security Foundations Workshop*, pages 321–334, 2006.

[DKR06] Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. Coercion-resistance and receipt-freeness in electronic voting. In *Computer Security Foundations Workshop (CSFW'06)*, pages 28–39, 2006.

[FR08] Cédric Fournet and Tamara Rezk. Cryptographically sound implementations for typed information-flow security. In *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'08)*, pages 323–335. ACM, 2008.

[GGvR08] David Galindo, Flavio D. Garcia, and Peter van Rossum. Computational soundness of non-malleable commitments. In *Proc. 4th Information Security Practice and Experience Conference (ISPEC'08)*, LNCS, 2008. To appear.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GO07] Jens Groth and Rafail Ostrovsky. Cryptography in the multi-string model. In *Appears in Advances in Cryptology - CRYPTO 2007*, volume 4622 of *LNCS*, pages 323–341, 2007.

[GS05] Prateek Gupta and Vitaly Shmatikov. Towards computationally sound symbolic analysis of key exchange protocols. In *Proc. of the 2005 ACM workshop on Formal methods in security engineering (FMSE'05)*, pages 23–32. ACM, 2005.

[GS06] Prateek Gupta and Vitaly Shmatikov. Key confirmation and adaptive corruptions in the protocol security logic. In *Proc. of the Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA'06)*, pages 113–142, 2006.

[GvR06] Flavio D. Garcia and Peter van Rossum. Sound computational interpretation of symbolic hashes in the standard model. In *Proc. International Workshop on Security 2006 (IWSEC'06)*, LNCS, pages 33–47. Springer, 2006.

[GvR08] Flavio D. Garcia and Peter van Rossum. Sound and complete computational interpretation of symbolic hashes in the standard model. *Theoretical Computer Science*, 394:112–133, 2008.

[Her03] Jonathan Herzog. A computational interpretation of dolev-yao adversaries. In *Proceedings of the 3rd IFIP WG1.7 Workshop on Issues in the Theory of Security (WITS'03)*, 2003.

[Her05] Jonathan Herzog. A computational interpretation of Dolev-Yao adversaries. *Theoretical Computer Science*, 340:57–81, June 2005.

[HG03] Omer Horvitz and Virgil D. Gligor. Weak key authenticity and the computational completeness of formal encryption. In *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *LNCS*, pages 530–547. Springer, 2003.

[JLM05] Romain Janvier, Yassine Lakhnech, and Laurent Mazaré. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *European Symposium on Programming (ESOP'05)*, volume 3444 of *LNCS*, pages 172–185. Springer, 2005.

[JLM06] Romain Janvier, Yassine Lakhnech, and Laurent Mazaré. Computational soundness of symbolic analysis for protocols using hash functions. In *Proceedings of the Workshop on Information and Computer Security (ICS'06)*, Electronic Notes in Theoretical Computer Science, Timisoara, Romania, September 2006. Elsevier Science Publishers.

[KDMR08] Ralf Küsters, Anupam Datta, John C. Mitchell, and Ajith Ramanathan. On the Relationships Between Notions of Simulation-Based Security. *Journal of Cryptology*, 2008. To appear. Electronic publication http://dx.doi.org/10.1007/s00145-008-9019-9.

[KM07] Steve Kremer and Laurent Mazaré. Adaptive soundness of static equivalence. In *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS'07)*, volume 4734 of *LNCS*, pages 610–625, Dresden, Germany, September 2007. Springer.

[KM09] Steve Kremer and Laurent Mazaré. Computationally sound analysis of protocols using bilinear pairings. *Journal of Computer Security*, 2009. To appear.

[KT08] Ralf Küsters and Max Tuengerthal. Joint state theorems for public-key encryption and digital signature functionalities with local computations. In *Computer Security Foundations (CSF'08)*, 2008.

[Lau01] Peeter Laud. Semantics and program analysis of computationally secure information flow. In *Proc. 10th European Symposium on Programming (ESOP'01)*, volume 2028 of *LNCS*, pages 77–91. Springer, 2001.

[Lau02] Peeter Laud. Encryption cycles and two views of cryptography. In *Nordic Workshop on Secure IT Systems (NORDSEC'02)*, 2002.

[Lau03] Peeter Laud. Handling encryption in an analysis for secure information flow. In *Proc. 12th European Symposium on Programming (ESOP'03)*, volume 2618 of *LNCS*, pages 159–173. Springer, 2003.

[Lau04] Peeter Laud. Symmetric encryption in automatic analyses for confidentiality against active adversaries. In *Proc. IEEE Symposium on Security and Privacy (SSP'04)*, pages 71–85, 2004.

[Lau05] Peeter Laud. Secrecy types for a simulatable cryptographic library. In *Proc. 12th ACM Conference on Computer and Communications Security (CCS'05)*, 2005.

[Lau08] Peeter Laud. On the computational soundness of cryptographically masked flows. In *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'08)*, pages 337–348. ACM, 2008.

[LC04] Peeter Laud and Ricardo Corin. Sound computational interpretation of formal encryption with composed keys. In *Proc. 6th International Conference on Information Security and Cryptology (ICISC'03)*, volume 2971 of *LNCS*, pages 55–66. Springer, 2004.

[LMMS98] Patrick Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. A probabilistic poly-time framework for protocol analysis. In *Proc. 5th ACM Conference on Computer and Communications Security (CCS'98)*, pages 112–121, 1998.

[Low96] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, March 1996.

[LT05] Peeter Laud and Ilja Tsahhirov. Digital signature in automatic analyses for confidentiality against active adversaries. In *Nordic Workshop on Secure IT Systems (NORDSEC'05)*, pages 29–41, 2005.

[LV05] Peeter Laud and Varmo Vene. A type system for computationally secure information flow. In *Proc. 15th International Symposium on Fundamentals of Computation Theory (FCT'05)*, volume 3623 of *LNCS*, pages 365–377. Springer, 2005.

[Maz07] Laurent Mazaré. Computationally sound analysis of protocols using bilinear pairings. In *Proc. 7th International Workshop on Issues in the Theory of Security (WITS'07)*, pages 6–21, 2007.

[MP05] Daniele Micciancio and Saurabh Panjwani. Adaptive security of symbolic encryption. In *Proc. 2nd Theory of Cryptography Conference (TCC'05)*, volume 3378 of *LNCS*, pages 169–187. Springer, 2005.

[MRST01] John C. Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. A probabilistic polynomial-time calculus for analysis of cryptographic protocols. In *Proc. 17th Annual Conference on the Mathematical Foundations of Programming Semantics*, volume 45 of *ENTCS*, 2001.

[MS03] Gerome Miklau and Dan Suciu. Controlling access to published data using cryptography. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 898–909. VLDB Endowment, 2003.

[MW02] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the abadi-rogaway logic of encrypted expressions. In *Proceedings of the 2nd IFIP WG1.7 Workshop on Issues in the Theory of Security (WITS'02)*, 2002.

[MW04a] Daniele Micciancio and Bogdan Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004.

[MW04b] Daniele Micciancio and Bogdan Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Proc. 1st Theory of Cryptography Conference (TCC'04)*, volume 2951 of *LNCS*, pages 133–151. Springer, 2004.

[RDDM07] Arnab Roy, Anupam Datta, Ante Derek, and John C. Mitchell. Inductive proofs of computational secrecy. In *Proceedings of the 12th European Symposium on Research in Computer Security (ESORICS'07)*, volume 4734 of *LNCS*, pages 219–234. Springer, 2007.

[RDM08] Arnab Roy, Anupam Datta, and John C. Mitchell. Formal proofs of cryptographic security of Diffie-Hellman based protocols. In *Revised Selected Papers from the 3rd Symposium on Trustworthy Global Computing (TGC'07)*, volume 4912 of *LNCS*, Sophia-Antipolis, France, 2008. Springer.

[RMST04] Ajith Ramanathan, John C. Mitchell, Andre Scedrov, and Vanessa Teague. Probabilistic bisimulation and equivalence for security analysis of network protocols. In *Proc. 7th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'04)*, volume 2987 of *LNCS*, pages 468–483. Springer, 2004.

[RMST06] Ajith Ramanathan, John C. Mitchell, Andre Scedrov, and Vanessa Teague. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *Theoretical Computer Science*, 353:118–164, 2006.

[SA06] Geoffrey Smith and Rafael Alpízar. Secure information flow with random assignment and encryption. In *Proc. Workshop on Formal methods in security engineering (FMSE'06)*, pages 33–44. ACM, 2006.

[SB08a] Christoph Sprenger and David Basin. Cryptographically-sound protocol-model abstractions. In *Logic in Computer Science (LICS 08)*, pages 3–17. IEEE Computer Society, 2008.

[SB08b] Christoph Sprenger and David Basin. Cryptographically-sound protocol-model abstractions. In *Computer Security Foundations (CSF 08)*, pages 115–129. IEEE Computer Society, 2008.

[SBB$^+$06] Christoph Sprenger, Michael Backes, David Basin, Birgit Pfitzmann, and Michael Waidner. Cryptographically sound theorem proving. In *Proc. 19th IEEE Computer Science Foundations Workshop (CSFW'06)*, pages 153–166, July 2006.

[Tar05] Sabrina Tarento. Machine-checked security proofs of cryptographic signature schemes. In *Proc. 10th European Symposium on Research in Computer Security (ESORICS'05)*, volume 3679 of *LNCS*, pages 140–158, 2005.

[War05] Bogdan Warinschi. A computational analysis of the Needham-Schroeder protocol. *Journal of Computer Security*, 13:565–591, 2005.

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399