LOGICAL CONCEPTS IN CRYPTOGRAPHY

THÈSE N^{O} 3845 (2007)

PRÉSENTÉE LE 19 JANVIER 2007

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Laboratoire de modèles et théorie de calculs

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Simon KRAMER

ingénieur informaticien diplômé EPF

de nationalité suisse et originaire de Charmey (Lac) (FR)

acceptée sur proposition du jury:

Prof. E. Telatar, président du jury

Prof. U. Nestmann, Prof. T. Henzinger, directeurs de thèse

Prof. S. Artëmov, rapporteur

Prof. R. Küsters, rapporteur

Prof. L. Moss, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL

2007

To my teachers

Abstract

Subject This thesis is about a breadth-first exploration of logical concepts in cryptography and their linguistic abstraction and model-theoretic combination in a comprehensive logical system, called CPL (for *Cryptographic Protocol Logic*). We focus on two fundamental aspects of cryptography. Namely, the security of *communication* (as opposed to security of *storage*) and cryptographic *protocols* (as opposed to cryptographic *operators*). The primary logical concepts explored are the following: the *modal* concepts of belief, knowledge, norms, provability, space, and time. The distinguishing feature of CPL is that it unifies and refines a variety of existing approaches. This feature is the result of our *wholistic conception* of property-based (modal logics) and model-based (process algebra) formalisms.

Keywords applied formal logic, information security.

URL http://library.epfl.ch/en/theses/?nr=3845

Résumé

Sujet Cette thèse a comme sujet une exploration en largeur de concepts logiques en cryptographie et leur abstraction et combinaison linguistique en un système logique syncrétique, appelé CPL (pour *Cryptographic Protocol Logic*). Nous nous intéressons à deux aspects fondamentaux de la cryptographie. D'une part, à la sécurité de la *communication* (par opposition à la sécurité du *stockage*) de données, et d'autre part, aux *protocoles* cryptographiques (par opposition aux *opérateurs* cryptographiques). Les concepts logiques primaires explorés sont les suivants : les concepts *modaux* de la croyance, de la connaissance, des normes, de la prouvabilité, de l'espace, et du temps. Le trait distinctif de CPL est d'unifier et de raffiner une variété d'approches existantes. Ce trait est le résultat de notre *conception holistique* des formalismes basés sur les propriétés (logiques modales) et de ceux basés sur les modèles (algèbres de processus).

Mots-clefs logique formelle appliquée, sécurité d'information.

URL http://library.epfl.ch/en/theses/?nr=3845

Synopsis

This thesis is about a breadth-first exploration of logical concepts in cryptography and their linguistic abstraction and model-theoretic combination in a comprehensive logical system, called CPL (for *Cryptographic Protocol Logic*). We focus on two fundamental aspects of cryptography. Namely, the security of *communication* (as opposed to security of *storage*) and cryptographic *protocols* (as opposed to cryptographic *operators*). The logical concepts explored are the following. PRIMARY CONCEPTS: the *modal* concepts of belief, knowledge, norms, provability, space, and time. SECONDARY CONCEPTS: belief with error control, individual and propositional knowledge, confidentiality norms, truth-functional and relevant (in particular, intuitionistic) implication, multiple and complex truth values, and program types. The distinguishing feature of CPL is that it unifies and refines a variety of existing approaches. This feature is the result of our *wholistic conception* of property-based (modal logics) and model-based (process algebra) formalisms. We illustrate the expressiveness of CPL on representative *requirements engineering* case studies. Further, we extend (core) CPL (qualitative time) with *rational-valued time*, i.e., time stamps, timed keys, and potentially drifting local clocks, to tCPL (quantitative time). Our extension is conservative and provides further evidence for Lamport's claim that adding real time to an untimed formalism is really simple. Furthermore, we sketch an extension of (core) CPL with a notion of *probabilistic polynomial-time* (PP) computation. We illustrate the expressiveness of this extended logic (ppCPL) on tentative formalisation case studies of fundamental and applied concepts. *Fundamental concepts*: (1) one-way function, (2) hard-core predicate, (3) computational indistinguishability, (4) (n-party) interactive proof, and (5) (n-prover) zero-knowledge. *Applied concepts*: (1) security of encryption schemes, (2) unforgeability of signature schemes, (3) attacks on encryption schemes, (4) attacks on signature schemes, and (5) breaks of signature schemes. In the light of logic, adding PP to a formalism for cryptographic protocols is perhaps also simple and can be achieved with an Ockham's razor extension of an existing core logic, namely CPL.

Moreover, we define: (1) *message meaning*; (2) *message information content*; (3) *protocol meaning*; and, based on all that, (4) *protocol information content*. From the meaning of a cryptographic message, we obtain (1) an equational definition of its *context-sensitivity*, and (2) a *formalisation* of the first of Abadi and Needham's principles for prudent engineering practice for cryptographic protocols. From the meaning of a cryptographic protocol, we obtain natural definitions of the concepts of (1) a protocol *invariant*, (2) protocol *safety*, and (3) protocol *refinement*. Last but not least, we show that *protocol agents* can be conceived as evolving *Scott information systems*.

Synthèse

Cette thèse a comme sujet une exploration en largeur de concepts logiques en cryptographie et leur abstraction et combinaison linguistique en un système logique syncrétique, appelé CPL (pour *Cryptographic Protocol Logic*). Nous nous intéressons à deux aspects fondamentaux de la cryptographie. D'une part, à la sécurité de la *communication* (par opposition à la sécurité du *stockage*) de données, et d'autre part, aux *protocoles* cryptographiques (par opposition aux *opérateurs* cryptographiques). Les concepts logiques explorés sont les suivants. Concepts primaires : les concepts *modaux* de la croyance, de la connaissance, des normes, de la prouvabilité, de l'espace, et du temps. Concepts secondaires : la croyance avec contrôle d'erreur, la connaissance individuelle et propositionnelle, les normes de confidentialité, l'implication vérifonctionnelle et pertinente (en particulier, intuitionniste), les valeurs de vérité multiples et complexes, et les types de programmes. Le trait distinctif de CPL est d'unifier et de raffiner une variété d'approches existantes. Ce trait est le résultat de notre *conception holistique* des formalismes basés sur les propriétés (logiques modales) et de ceux basés sur les modèles (algèbres de processus). Nous illustrons l'expressivité de CPL à l'aide d'études de cas représentatives de type *requirements engineering*. Ensuite, nous étendons CPL (temporisée qualitativement) avec du *temps à nombres rationnels*, c'est-à-dire, avec des étiquettes temporelles, des clefs temporisées, et des horloges locales potentiellement dérivantes dans le temps, à tCPL (temporisée quantitativement). Notre extension est conservatrice et va dans le sens de la thèse de Lamport qui affirme qu'ajouter du temps réel à un formalisme est vraiment simple. En outre, nous esquissons une extension de CPL avec une notion de *calcul probabiliste et polynomial*. Nous illustrons l'expressivité de cette logique étendue (ppCPL) à l'aide d'études de cas expérimentales de concepts fondamentaux et appliqués. Concepts fondamentaux : (1) les fonctions à sens unique, (2) les prédicats de type *hard-core*, (3) l'indistinguabilité computationnelle, (4) les preuves interactives (à n entités), et (5) les preuves à divulgation nulle de connaissance (à n entités). Concepts appliqués : (1) la sécurité du et les attaques des algorithmes de chiffrage ; et (2) la non-falsifiabilité, les attaques des algorithmes de création, et les falsifications des signatures électroniques. À la lumière de la logique, le rajout d'une notion de calcul probabiliste et polynomial à un formalisme pour protocoles cryptographiques est peut-être aussi simple et peut être réalisé avec une extension selon le principe du rasoir d'Ockham d'une logique existante, à savoir CPL.

De plus, nous définissons : le *sens d'un message* ; le *contenu en information d'un tel message* ; le *sens d'un protocole* ; et sur base de tout cela, le *contenu en information d'un tel protocole*. À partir du sens d'un message cryptographique, nous obtenons (1) une définition équationnelle de sa *dépendance du contexte*, et (2) une *formalisation* du premier des principes d'ingénierie prudente pour protocoles cryptographiques d'Abadi et Needham. À partir d'un protocole cryptographique, nous obtenons des définitions naturelles des concepts (1) d'*invariant* de protocole, (2) de *sûreté* de protocole, et (3) de *raffinement* de protocole. Enfin, nous montrons que *les agents de protocole* peuvent être conçus comme *des systèmes d'information de Scott*.

Contents

List of Tables

List of Slogans[1]

[1] meant

- to connote (not to denote) scientific *content*
- to denote (not to connote) scientific *intent*

Part I

Prologue

Chapter 1

Preface

> *Studies in the foundations of mathematics divide symmetrically into two sorts, conceptual and doctrinal. The conceptual studies are concerned with meaning, the doctrinal with truth. The conceptual studies are concerned with clarifying concepts by defining them, some in terms of others. The doctrinal studies are concerned with establishing laws by proving them, some on the basis of others. Ideally the obscurer concepts would be defined in terms of the clearer ones so as to maximize clarity, and the less obvious laws would be proved from the more obvious ones so as to maximize certainty. Ideally the definitions would generate all the concepts from clear and distinct ideas, and the proofs would generate all the theorems from self-evident truths. The two ideals are linked.*[1]
>
> **Willard V. Quine**
> (cf. [Gib04, Page 259])

This thesis is the inception of a *conceptual* study in the logical foundations of cryptography. Our perspective is meta-theoretic; our concern applied; and our approach formal, and based on the following conviction[2], formulated as a slogan:

Slogan 1 *Logic is the interdisciplinary and unifying scientific discipline* par excellence.

Our perspective is meta-theoretic because such a perspective is conducive to the perception of pertinent connections between disciplines that the members of their respective communities typically claim to perceive as safely unconnected. Our concern is applied for the sake of practical relevance. And our approach is formal for the sake of scientific rigour.

Our conceptual study is concerned with the clarification of the meaning of cryptographic intuitions by reducing them to logical concepts and defining them in terms of *logical constructions*. According to Quine: "It is valuable to

[1] the link is "the duality between concept and doctrine, between knowing what a sentence means and knowing whether it is true." [Gib04, Page 273]

[2] The community-centered reader uprooted by this individual declaration of faith can be reassured that our conviction is socially well-founded — in another community, namely the one of mathematical foundationalists (though not necessarily fundamentalists).

show the reducibility of any principle to another through definition of erstwhile primitives, for every such achievement reduces the number of our presuppositions and simplifies and integrates the structure of theories." [Gib04, Page 30]. Our enterprise is clearly in the spirit of mathematical foundationalism and thus consistent with the maxim of the first quotation to define the obscurer (i.e., cryptographic) concepts in terms of the clearer (i.e., logical) ones so as to maximise clarity. Our ambition is concomitant with the maxim of the first quotation that ideally, the (logical) definitions would generate all the concepts (those relevant to cryptographic protocols) from clear and distinct ideas. Our study is conceptual because (and that is the link in the first quotation):

Slogan 2 *Proof of concept*[3] *is necessary for the concept of proof.*

And the concept of proof is well-understood, as opposed to cryptographic concepts, which according to Goldreich are not: "To summarize, the basic concepts of cryptography are indeed very natural, but they are *not* self-evident nor well understood. Hence, we do not yet understand these concepts well enough to be able to discuss them *correctly* without using precise definitions and rigorously justifying every statement made." [Gol01]. This thesis is our humble contribution to the discussion in the form of a *logical conceptualisation* of the security of communication at the level of cryptographic protocols and its crystallisation into a comprehensive *logical theory*. Logic is about correct (and hopefully, inter-communal) discourse and conversation (and hopefully, dialogue). Our attitude is curiosity in — and service to — cryptography. Our hope is to awake awareness of the relevance of formal logic applied to cryptography.

Slogan 3 *The subject-matter of applied formal logic is pragmatics of cognition. Its purpose is empowerment of the human mind in the formulation, validation, and communication of statements.*

Audience

Our target audience is the community of logically inclined engineers, computer scientists, cryptographers, logicians, and philosophers. Our (meta-)community is the one of all those who believe that thinking in terms of closed scientific communities is imprisonment of the mind and ultimately anti-scientific. The concept of a closed scientific community is a strong logical paradox, i.e., nonsense.

Acknowledgements

I would like to thank: Johannes Borgström for his contribution to our joint work, and for teaching me through that work what I know about process algebra; Mika Cohen for our stimulating discussions about crypto logics and his constructive criticism of my work; Marc De Falco for his exploratory (now superseded) contribution to my work at an experimental stage; Christoph Frei for his elder-brotherly advice on the practice of being a Ph.D. student; Henrik Imhof for his abiding Upanishads on mathematical logic; Michael Mendler for our early

[3] or at least, *evidence* for concept

discussions about crypto logics; and Jacques Zahnd for his nonpareil teachings of elementary logic [Zah03], which have gained the status of true *urelements* in my education.

Last but not least, I would like to thank the members of my thesis committee for their appreciation of my thesis, and my supervisors for their logistic support during my thesis. I am especially grateful to Johan van Benthem and Lawrence Moss (this thesis is a tribute to his manifesto for applied logic [Mos05]) for their valuable comments and encouraging opinion about my work.

The research documented in this thesis has been funded by the Swiss National Science Foundation.

Ecole Polytechnique Fédérale de Lausanne
Lausanne, Switzerland

Simon Kramer[4]
July 2007

[4] simon.kramer@a3.epfl.ch

Chapter 2

Introduction

> *Mechanisms come and go, are improved upon, rarely become popular, and are never really basic and compelling enough to bring satisfaction. Theorems are ignored or strengthened, forgotten, and hardly anyone reads their proofs or cares. What lasts, at least for a little while, are the notions of the field and that which is associated to making those notions precise: definitions.*
>
> **Phillip Rogaway**
> (cf. [Rog04])

This thesis proposes *definitions* (and a few theorems with proofs) for cryptographic notions that are *heretical* w.r.t. the established canon of modern cryptography. Our definitions are heretical in the sense that we choose to formulate them in terms of *logical constructions*, rather than in terms of the canonical Turing machines. The benediction of our heresy is to provide *linguistic abstractions* of cryptographic concepts. The benefit of linguistic abstractions is to produce *diction (idioms) for intuitions*. Additionally, our linguistic abstractions have the benefit of being *declarative* abstractions (i.e., they focus on the *what*) of operational (focusing on the *how*) aspects.[1] The benefit of logic is to enable humans to think and communicate with each other as humans rather than as machines. Humans, perhaps with the exception of computer geeks and nerds, think and communicate in the language of the mind, i.e., logic, rather than in the (programming) language of (Turing) machines.

Slogan 4 *The purpose of logic is understanding. The purpose of machines is control.*[2]

2.1 Motivation: problem

The motivation for our heresy is a certain dissatisfaction with the canonical definitions of modern cryptography. Our diagnosis of the purported problem is the following.

[1] our intention is in the spirit of *descriptive* rather than *computational* complexity theory
[2] no negative connotation intended

2.1.1 Symptom

Traditional definitions of modern cryptographic concepts yield proofs (that is the link in the first quotation of Chapter 1) that are obscure, in the sense that they are *mentally intractable*. Even cryptographers themselves have come to argue for "taming the complexity of security proofs that might otherwise become so messy, complicated, and subtle as to be nearly impossible to verify." [Sho04]. In other words, the decision problem consisting in the question of whether or not a traditional security proof is correct typically has no succinct certificate. In particular, the candidate proof itself is then no such certificate. This is an empirical fact and a dissatisfactory state of affairs. It means that traditional security proofs, which aim at establishing certain (computational) intractability results, are themselves (mentally) intractable. In other words, (modern) cryptography disproves itself with its own (traditional) proofs. An empirical corollary is the dominance of proofs (by example) of the presence of attacks on cryptographic constructions and on proofs thereof, over proofs (by argument) of the absence of such attacks.

2.1.2 Cause

The cause of this self-inflicted obscurity of modern cryptography is deep-rooted — in its *definitions*. Traditional security proofs are mentally intractable because they rely on definitions that aim at capturing extremely *high-level declarative* concerns (e.g., trust, confidentiality, identity, commitment) with extremely *low-level operational* linguistic abstractions (i.e., Turing machine instructions). The generated gap between the *what* and the *how* is abysmal. A fortiori, *formal* proofs relying on traditional definitions are mentally intractable because the *literal* programming of Turing machines, on which such proofs would depend, is. As a matter of fact, writing formal proofs is unpopular. This would involve unpopular formal logic. An empirical corollary is the dominance of indirect property statements about cryptographic constructions. That is, the statement of properties in terms of *how* they are supposed to be established, rather than just the statement of *what* they are supposed to establish. As a matter of fact, direct property statements are unpopular. Again, this would involve unpopular formal logic.

2.1.3 Remedy

A deep-rooted problem (definitions) must be administered a radical remedy: *redefinition* — at least at first sight and stage. It is preconceived usage that defines the concepts of modern cryptography in terms of Turing machines. This need not be so. In Quine's words: "Preconceived usage may lead us to stack the cards, but does not enter the rules of the game." [Gib04, Page 22]. And: "There are indefinitely many ways of framing definitions [...] choice among these ways is guided by convenience or chance." [Gib04, Page 13]. That is, definition is by definition heretical; it requires the ability to choose (a priori), and is an act of choice (a posteriori).

The heresy of modern cryptography was guided by *chance* because computational complexity theory, which it is based on, *happens* to be defined in terms of (probabilistic) Turing machines by the mainstream of computer scientists.

On account of the (proof-)technical inconvenience that Turing machines entail in cryptography, this is lamentably bad luck.

Slogan 5 *Proving with Turing machines is like programming with μ-calculus*[3].

That is, formal proofs *with* Turing machines are (doubly) inadequate because they are *mentally intractable* (too low-level) and — because they are — *inappropriate* (they use the wrong tool). Turing machines are appropriate as a model of computation, not as a means to proofs. A fortiori, *in*formal proofs with Turing machines are inadequate because the are, besides being mentally intractable and inappropriate, also *inaccurate*. In view of the theoretical subtleties and practical pitfalls of cryptography, this is pitiful bad practice.

2.2 Goal: solution

The cure for proof-technical inconvenience is, of course, *logic*, more precisely proof theory. That is our ultimate goal. Our intermediate goal and the one of this thesis must be the synthesis of the remedy, i.e., the synthesis of a logical system powerful enough to allow for the meaningful redefinition of cryptographic concepts.

2.2.1 Logic

The logical solution to a problem of meaning is a *relation of satisfaction* (or *modelling relation*). That is, a relation between a model (of a cryptographic protocol in our case) and a formula (expressing a statement about that protocol) asserting that the formula is a true statement about the considered model. Thus this thesis is about *model theory* for cryptography, although we shall also address provability but from a model-theoretic point of view.

2.2.2 Extent

The "redefinition" of the whole of modern cryptography is, of course, out of the scope of a Ph.D. thesis.

2.2.2.1 Scope

We shall focus on logical constructions for the security of *communication* (as opposed to security of *storage*) at the level of cryptographic *protocols* (as opposed to cryptographic *operators*). A cryptographic operator is to a cryptographic protocol what a brick is to a brick-house. Good bricks are necessary but not sufficient for good brick-houses.

[3] a powerful fixpoint logic famous for its computational tractability and its mental tractability at the meta-level, but infamous for its mental *in*tractability at the *object* level (frequently referred to as the machine-code-level language of program logics)

2.2.2.2 Grain

The focus on cryptographic protocols provides a macroscopic view on the security of communication. At this scale of sight, cryptographic operators are ideally perceived as black boxes with perfect functionality (the so-called Dolev-Yao abstraction). Perfect operator functionality guarantees functionality of operators in spite of *ideal* (i.e., *information*-theoretic) adversaries and discharges the protocol functionality from the assumed perfect functionality of the operators the protocol employs. In contrast, imperfect operator functionality only guarantees functionality of operators in spite of *real* (i.e., *complexity*-theoretic) adversaries. In consequence, operator functionality might infringe on protocol functionality.

We shall construct two optics of different cryptographic grain for our logical system. The first optic will have a lens to accommodate the grain of perfect cryptography, and the second a tentative lens to accommodate the grain of probabilistic polynomial-time cryptography.

2.3 Methodology

We shall derive the synthesis of our logical system from a comprehensive analysis of the cryptographic concepts that are relevant to the security of communication. Preceding analysis is necessary for successful synthesis. For example, the synthesis should respect conceptual orthogonality and hierarchy.

2.3.1 Approach

Our conceptual analysis resembles Leibniz's approach described in his *magnus opus*, namely the *Monadology*. There, Leibniz describes the analysis of concepts as a breaking down into their atomic constituents (the so-called *monads*) comparable to the factorisation of natural numbers into their primes. Proceeding this way, Leibniz ultimately arrives at primitive concepts.

Our approach is *goal-oriened* in the sense that we discover logical concepts in natural-language formulations of goals for cryptographic protocols. We then cast these natural-language formulations (in mental step-wise refinement of precision) into formulae of an imaginary logical language and invent an adequate semantics (the relation of satisfaction) for these formulae.

The challenge in this enterprise is to find the right primitive linguistic abstractions and the right method of their logical combination. Metaphorically speaking, our task is to find the right "cryptographic" prime factors and the right notion of — and algorithm for — their factorisation. This is a non-trivial task.

2.3.2 Formalism

A distinguishing feature of our approach is that we use formal logic in a doubly *unifying* sense. That is, in the sense of the unification of different (modal) logics and in the sense of the integration of logic with (formal) program semantics.

Manifesto There is a *fictive schism* between the so-called formal-methods community and the hard-core cryptography community on the subject of formalism. The formal-methods community professes that formalism is good be-

cause formalism means abstraction from bit-strings and thus automation. The hard-core cryptography community professes that formalism is bad because formalism means abstraction from bit-strings and thus imprecision. The schism is fictive because both professions proclaim a mirage, namely the one of imagining that possibility be necessity. Formalism possibly, but not necessarily, means abstraction. When it means abstraction, it effectively means automation and possibly imprecision; otherwise it means neither of both. It is a misconception to believe that bit-strings cannot be reasoned about formally. A bit-string can be simply and faithfully modelled as a syntactic term, i.e., as a string of symbols over the alphabet $\{0, 1\}$. And that bit of syntax does not require big bites of formalism. Both professions need confession: this schism is nonsense[4].[5]

2.3.2.1 Unifying modal logics for cryptography

As our imaginary logical language we shall "use the language of modal logic as an aid to precision" [BvBW07, Page xvi]. Modal logics are logics with *modal* operators (*modalities*), i.e., operators that are *not truth-functional*. A non-truth-functional operator is an operator that, when applied to operands, yields a formula whose truth value cannot be established as a mere function of the truth values of its operands. Many modal logics are relevant to the purpose of our conceptual investigation. Thereof, we shall demonstrate the relevance of the modal logics of knowledge, norms, provability, space, and time. Unifying these logics in a single many-dimensional modal logic for that purpose is clearly desirable.

2.3.2.2 Integrating modal logic and program semantics

As a complement to our logical language, we shall introduce a programming language, with adequate semantics, for cryptographic protocols. The programs of this programming language are the models for our logical formulae.

Slogan 6 *The subject-matter of programming languages is pragmatics of control (of action for effect). Their purpose is empowerment of the human mind in the formulation of instructions for machines.*

The complementary nature of our two languages is explained by five natural, but notwithstanding novel (except for Item 4 and 5), integrating design decisions: (1) define the meaning of a cryptographic protocol (its denotational semantics) in terms of the meaning of the cryptographic messages it produces during its execution, i.e., define the *what* (denotation) in terms of the *how* (operation); (2) define the meaning of a cryptographic message in terms of the *propositional* knowledge a protocol agent *would* acquire from the (*individual*) knowledge of that message; (3) define *dynamic* observational equivalence (indistinguishability of execution *paths*) in terms of *static* observational equivalence

[4] we mean *scientific* (as opposed to *political*) nonsense

[5] The executive reader might object that both sides have found a *terrain d'entente* in soundness (and incompleteness) results w.r.t. the "formal view" on cryptography in the sense of Dolev-Yao (so cryptographers still do the real stuff). That reader is urged to consider that that view is not necessarily a limit to the formal view in the sense of Gödel. The question thus is: "Is cryptography completely axiomatisable at the level of cryptographic protocols?" (At the level of cryptographic operators it is not due to the involved number theory.)

(indistinguishability of execution *states*); (4) identify *static observational equivalence* with *epistemic accessibility* (the semantics of propositional knowledge); and (5) identify the *operational semantics* (protocol execution) with *temporal accessibility* (the semantics of temporal propositions).

2.4 Prerequisites

The prerequisites for this thesis are, by the very nature of our exposition, basic knowledge of logic, program semantics, and cryptography. We shall not restate that basic knowledge here, but rather confine ourselves to pointing the needful reader to the relevant literature here and to introducing more advanced knowledge on the fly in our exposition ahead.

2.4.1 Logic and program semantics

A classical, comprehensive reference for general mathematical logic is [Bar99]. The relevant chapters for our exposition are: A.1 (first-order logic, by J. Barwise), B.1 (set theory, by J. R. Shoenfield), and C.7 (inductive definitions, by P. Aczel). A recent, comprehensive reference for modal logic is [BvBW07]. The relevant chapters for our exposition are: 1 (semantics, by P. Blackburn and J. van Benthem), 9 (first-order modal logic, by T. Braüner and S. Ghilardi), 11 (temporal logic, by I. Hodkinson and M. Reynolds), 16 (provability and spatial logic, by S. Artëmov), and 18 (deontic, doxastic, and epistemic logic, by J.-J. Meyer and F. Veltnam). A classical, comprehensive reference for program semantics is [vL90]. The relevant chapters for our exposition are: Denotational Semantics by P. D. Mosses, and Operational and Algebraic Semantics of Concurrent Processes by R. Milner.

2.4.2 Cryptography

Slogan 7 *Cryptography is a technological means to information security.*

A two-volume foundational reference for modern cryptography is [Gol01, Gol04]. A comprehensive reference for applied cryptography is [MvOV96] (where from we shall often quote in this thesis). A comprehensive reference for cryptographic protocols is [BM03]. And an encyclopaedic reference for the whole field is [vT05].

2.4.2.1 Prehistory

The prehistory of cryptography (i.e., message obfuscation) is *steganography* (i.e., message hiding). In general, a steganographic message is hidden in another, apparent, message, whereas a cryptographic message is apparent as such. It is possible to combine steganography with cryptography, e.g., by encrypting the steganographic message. With the advent of redundant, digital representation of information, steganography has regained relevance in modern times. Data compression on the contrary makes steganography more difficult.

2.4.2.2 Classical cryptography

Classical cryptography is based on *information theory*[6]. Its first published, mathematical treatment appeared in 1949 under the title "Communication Theory of Secrecy Systems" by C. Shannon. Classical cryptography guarantees *perfect secrecy* of the encrypted plaintext $\{|M|\}_k$ under the assumptions of the perfect secrecy of the encrypting key k required to be of the same length as the plaintext M. A corollary of the equal-length assumption is that the key be used only once. That is, using the key a second time divides its entropy (i.e., information indeterminacy) by two. Whence the name *one-time pad* of this (XOR-based) encryption scheme. A property of classical encryption schemes is that they are *symmetric*, i.e., the same key is used for en- and decryption.

2.4.2.3 Modern cryptography

Modern cryptography is based on (computational) *complexity theory*. It does not guarantee perfect secrecy, but, besides "computational" secrecy, aims at guaranteeing many more desirable properties related to information security. The assumptions of modern cryptography are the existence of *one-way functions* and of *true randomness*. One-way functions are functions whose inversion is computationally intractable. An important result of modern cryptography is that true randomness can be arbitrarily well approximated by pseudo randomness, i.e., the randomness furnished by classical (as opposed to quantum) computers. Security of cryptographic schemes is demonstrated by reduction to computational problems whose hardness is an empirical fact. A property of modern encryption schemes is that they are possibly *asymmetric*, i.e., different keys are used for en- and decryption. The first published treatment of asymmetric schemes appeared in 1976 under the title "New Directions in Cryptography" by W. Diffie and M. Hellman. The first published implementation of an asymmetric scheme appeared in 1978 and is due to R. Rivest, A. Shamir, and L. Adleman.

Cryptographic operators The traditional occupation of crypto*graphers* is the *con*struction of *operators* for cryptographic *tasks* such as en- and decryption, electronic signature generation and verification, and irreversible obfuscation and destructive compression of data (data hashing). The traditional occupation of crypt*analysts* is the "*de*struction" of those operators, i.e., the breaking of their intended functionality.

Cryptographic protocols A more modern occupation of cryptographers is the construction of *protocols* for cryptographic *concerns* (e.g., trust, confidentiality, identity, and commitment) by employing cryptographic operators. Such concerns arise in the context of communication in hostile environment. The occupation of hostile communicators (so-called *adversaries*) is the "destruction" of those protocols, i.e., the breaking of their intended functionality. Adversaries can be passive and active. A *passive adversary* may *eavesdrop*, i.e., (1)

[6] there is also a relation to *coding theory*: "Cryptography distinguishes itself from coding theory in the sense that the presence of random noise in the latter is replaced by malicious adversaries in the former." [Vau05, Page 1]

read any message, (2) decompose a message into parts (*analysis*) and remember them. An *active adversary* may also (3) block message transmission, (4) generate fresh data as needed, and (5) compose new messages from known data (*synthesis*) and send messages. That is, eavesdropping is unaltered (*no tampering*) eventual forwarding (at most *temporary blocking*) of intercepted messages. It is common, theoretical practice to assume the existence of one, archetypical adversary, and to identify that adversary with the communication network. It has been reasoned that this is reasonable practice [CLC04].

Our interest with cryptographic protocols is the *declarative* description (the *what*) of their intended functionality. Such declarative statements have the advantage of being mentally tractable (and hopefully, compelling to bring satisfaction eventually) as we will show. In contrast, cryptographic protocols themselves, i.e., their *operational* descriptions (the *how*), despite their small size, are mentally *in*tractable, as experience has amply shown.

2.5 Preview

We produce the following declarative descriptions of cryptographic states of affairs and cryptographic concepts that are relevant to the functionality of cryptographic protocols.

2.5.1 Cryptographic states of affairs

Trust-related affairs maliciousness, honesty, faultiness, prudency, and trustworthiness of protocol agents (cf. Section 3.3.1).

Confidentiality-related affairs shared secret, secrecy, anonymity, data derivation, non-interaction, perfect forward secrecy, known-key attack, and agent corruption (cf. Section 3.3.2).

Authentication-related affairs key confirmation, key authentication (implicit and explicit), message integrity, message authorship, message authentication (authenticity), key transport (unacknowledged and acknowledged), key agreement (unacknowledged and acknowledged), entity authentication (identification) (unilateral, weakly mutual, and strongly mutual) (cf. Section 3.3.3).

Commitment-related affairs cryptographic proof, cryptographic evidence, provability, non-repudiation, contract signing, (optimism, completion, accountability, and abuse-freeness) (cf. Section 3.3.4).

Compositionality-related affairs key separation, compositional correctness (existential composability, conditional composability, and universal composability), and attack scenario (cf. Section 3.3.5).

2.5.2 Cryptographic concepts

Fundamental concepts one-way function, hard-core predicate, computational indistinguishability, (n-party) interactive proof, interactive provability, proof of knowledge, and (n-prover) zero-knowledge (cf. Section 5.3.1).

Applied concepts security of encryption schemes (standard, semantic, via indistinguishability, and via non-malleability), unforgeability of signature schemes, attacks on encryption schemes (ciphertext-only attack, known-plaintext attack, chosen-plaintext attack, adaptive chosen-plaintext attack, chosen-ciphertext attack, and adaptive chosen-ciphertext attack), attacks on signature schemes (key-only attack, known-message attack, chosen-message attack, and adaptive chosen-message attack), and breaks of signature schemes (existential forgery, selective forgery, universal forgery, and total break) (cf. Section 5.3.2).

2.5.3 Research highlight

Our research highlight is having demonstrated the macro-definability of a *Gödel-style provability modality* within the spatio-epistemic fragment of CPL (cf. Theorem 2). With this modality, CPL can capture the provability meaning of intuitionistic implication, and provability is shown to be the key to the formalisation of *commitment* and related cryptographic states of affairs (cf. Section 3.3.4).

2.5.4 Peer-review

The content of this thesis has been validated through the publication of three peer-reviewed short papers [Kra03], [Kra06a], and [Kra06b]; four workshop papers [Kra04], [BKN06], [BGK06], and [Kra07a, Kra07b]; and one journal paper [Kraar].

Part II

Cryptographic Protocol Logic

Chapter 3

Dolev-Yao cryptography

3.1 Introduction

We give a comprehensive motivation for our approach to the correctness of cryptographic protocols by placing the approach in its historical and topical context. The length of the introduction reflects our desire to expose a wide and deep perspective on the highly interdisciplinary field of cryptographic protocols.

3.1.1 Historical context

"A cryptographic protocol [...] is a distributed algorithm defined by a sequence of steps precisely specifying the actions required of two or more entities to achieve a specific security objective." [MvOV96, Page 33]. Principal security objectives are secrecy of confidential information, authenticity of received messages w.r.t. their origin, and non-repudiation of message authorship. Our slogan is:

Slogan 8 *The purpose of a cryptographic protocol is to interactively compute, via message passing*[1]*, knowledge of the truth of desired — and, dually, knowledge of the falsehood of undesired — cryptographic states of affairs.*

In 1996, Anderson and Needham assert that cryptographic protocols typically "involve the exchange of about 2–5 messages, and one might think that a program of this size would be fairly easy to get right. However, this is absolutely not the case: bugs are routinely found in well known protocols, and years after they were first published. The problem is the presence of a hostile opponent, who can alter messages at will. In effect, our task is to program a computer which gives answers which are subtly and maliciously wrong at the most inconvenient possible moment." [AN96b]. Indeed, designing a correct cryptographic protocol (i.e., "programming Satan's computer" [AN96b]), is extremely more difficult than designing a correct, ordinary computer program (i.e., "programming Murphy's [computer]" [AN96b]) of the same size. In fact, at the end of the 1980s, i.e., 20 years after the surge of the software crisis in the software-engineering community, the communication-security community was also shaken by a software crisis, though a different one. The first software crisis was provoked by the

[1] rather than shared memory

(increasing) *size* of computer programs [Dij72], whereas the second crisis was triggered by the (sudden, e.g., [BAN90]) awareness about the complexity of the *structure* of a certain class of such programs, namely cryptographic protocols. Our slogan, especially applying to cryptographic protocols, is:

Slogan 9 *In theory, it is possible to construct a correct computer program without knowing a theory of program correctness; in practice, it rarely is.*

The answer to both software crises has really been the formal-methods movement. In 1999, McLean affirms that "[o]ne of the biggest success stories of formal methods in the computer security community is the application of them to cryptographic protocols. Cryptographic protocols are small enough to be susceptible to complete formal analysis, and such analyses have turned up flaws that would have, otherwise, gone undetected." [McL99]. However, McLean also points out "the need for more research in the specification arena." in the same paper. In 2003, Meadows reaffirms and strengthens the importance of that issue by observing that "[...] although it is difficult to get cryptographic protocols right, what is really difficult is not the design of the protocol itself, but of the requirements. Many problems with security protocols arise, not because the protocol as designed did not satisfy its requirements, but because the requirements were not well understood in the first place." [Mea03] (consider also, more generally, [Rog04]). Our slogan is:

Slogan 10 *Cryptographic protocol correctness: a killer application for formal methods.*

3.1.2 Topical context

3.1.2.1 Requirements engineering—ideally

Indeed, the construction of a cryptographic protocol begins (and "ends" if this stage is not mastered) with *requirements engineering*, i.e., the definition of the requirements (global properties) the protocol is supposed to meet. In particular, understanding protocol requirements is necessary for understanding protocol attacks, which can be looked at as falsifications of necessary conditions for the requirements to hold. Protocol specification (requirements engineering), design (modelling), verification, and implementation (programming) are engineering tasks (the spirit of [MvOV96]). In contrast, the construction of a cryptographic operator (for encryption, signing, and hashing) is a scientific task (the spirit of [Gol01, Gol04]) requiring profound expertise from different fields of discrete mathematics.[2] Protocol engineers do (and should) not have (to have) this expertise. For example, it is legitimate for a protocol engineer to "abstract" negligible probabilities and consider them as what they are — negligible. Ideally, engineers should only have to master a single, common, and formal language for requirements engineering that adequately abstracts "hard-core" mathematical concepts.

[2] consider also [Riv90]: "The design of protocols and the design of operators are rather independent [...]. The protocol designer creates protocols assuming the existence of operators with certain security properties. The operator designer proposes implementations of those operators, and tries to prove that the proposed operators have the desired properties."

Since logic is what all sciences have in common, it is natural to stipulate that such a *lingua franca* for requirements-engineering cryptographic protocols be an appropriate logical language.

Program statement We argue that a good candidate language is a candidate that is *technically adequate* and *socially acceptable*. By a technically adequate candidate we mean a candidate that (1) is semantically and pragmatically *sufficiently expressive*, i.e., *versatile* and yielding *intuitive* specifications, respectively; (2) has a cryptographically *intuitive semantics*; (3) is *completely axiomatisable*; and (4) has important *decidable fragments* (e.g., the temporal fragment). By a versatile candidate we mean a candidate that allows all desirable specifications to be directly expressed, or else defined, in terms of the primitives of the candidate. By intuitive specifications we mean that the conceptual dimensions of a specification are apparent in distinctive forms in the formula that expresses the specification — succinctly. By a socially acceptable candidate we mean a candidate that *unifies* and possibly *transcends* previous specification languages.

Our task shall be to synthesise the relevant logical concepts in cryptography into a cryptographic protocol logic with a temporal-logic skeleton. Our preference of temporal logic over program logics such as Hoare and dynamic logic is motivated by the success of temporal logic as a specification language for (non-cryptographic) interactive systems. We will validate our language, at least at a first stage, on specification (stress on different requirements) rather than verification (stress on different protocols) case studies, since program specification must in theory, and should in practice—where it unfortunately rarely does—precede program verification. Nonetheless, the existence of verification examples is guaranteed by *subsumption* under CPL of other logics from authors with the opposite focus.

3.1.2.2 Requirements engineering—really

We briefly survey requirements engineering (the practice of the specification) of cryptographic protocols. Protocol designers commonly define a cryptographic protocol jointly by a semi-formal description of its *behaviour* (or *local* properties) in terms of *protocol narrations*, and by an informal prescription of its *requirements* (or *global* properties) in *natural language* [BM03]. Informal specifications present two major drawbacks: they do not have a well-defined, and thus a well-understood *meaning*, and, therefore, they do not allow for verification of correctness. In *formal* specifications of cryptographic protocols, local and global properties are expressed either explicitly *as such* in a logical (or *property*-based) language, or implicitly *as code*, resp. *as encodings* in a programming (or *model*-based) language (e.g., applied λ-Calculus [SP03]; process calculi: CSP [RSG+00], applied π-Calculus [AF01], Spi-Calculus [AG99], and [MRST06]).

Model-based languages The most popular examples of such encodings are equations between protocol instantiations [Aba00]. However, such encodings present four major drawbacks: (1) they have to be found for each protocol anew; worse, (2) they may not even exist; (3) they are neither directly comparable with other encodings in the same or in other programming languages, nor with properties expressed explicitly in logical languages; and (4) they are not easy to

understand because the intuition of the encoded property is not explicit in the encoding; yet "[r]obust security is about explicitness." [AN96b]! On the other hand, process calculi are ideal *design formalisms*. That is, they offer — due to their minimalist, linguistic abstractions of modelling concepts (syntax) and their mathematical, operational notion of execution (semantics) — a win-win situation between the (pedantic) rigour of (Turing) machine models and the (practical) usability of mainstream programming languages.

Property-based languages Still, informal language and programming (or *effect*) languages are inadequate for expressing and comparing cryptographic properties. It is our belief that only a *logical* (or *truth*) language equipped with an appropriate notion of truth, i.e., a cryptographic *logic*, will produce the necessary adequacy. A number of logics have been proposed in this aim so far, ranging from *special-purpose*, cryptographic logics: the pioneering BAN-logic [BAN90], a unification of several variants of BAN-logic [SvO96], and a recent reworking of BAN-logic [KS06]; over general-purpose propositional, modal, program, and first- and higher-order logics used for the special purpose of cryptographic protocol analysis: *propositional* ("logic programming") [AM01, AB05]; *modal*: deontic [BC93], doxastic [ABV01, ZV01], epistemic [SS99, HO02], linear [BD04], temporal [GM95]; *program*: dynamic [FHJ02, ADM03], Hoare-style [DMP03, LA05]; *first-order* [CJM98, Sel01, Coh03]; *higher-order* [Pau98, IK06]; to combinations thereof: doxastic-epistemic [CS97], doxastic-temporal [BGPS00], distributed temporal [CVB05], dynamic-epistemic [Bal01], epistemic-temporal [DGMF04, LW06] first-order-temporal [GLL01], dynamic-epistemic-temporal [Bie90], and deontic-epistemic-temporal [GMP92].

All these logics have elucidated important concerns of the security of communication and proved the relevance of logical concepts to that security. In particular, mere enunciation of maybe the three most fundamental protocol requirements, namely secrecy, authenticity, and non-repudiation, reveals the paramount importance of the concept of *knowledge*, both in its *propositional* (so-called knowledge *de dicto*) and in its *individual* (so-called knowledge *de re*) manifestation. Possible[3] enunciations in natural language of these requirements are the following (cf. Section 3.3 for their formalisation in CPL). Secrecy for a protocol: "Always and for all messages m, if it is forbidden that the adversary (Eve) *know* m then Eve does not *know* m." (knowledge de re in the present subjunctive and the present indicative mode, respectively). Authenticity of a message m from the viewpoint of agent a w.r.t. agent b: "a *knows that* once only b *knew* m." (knowledge de dicto in the present and knowledge de re in the past indicative mode). Non-repudiation of authorship of a message m' by b w.r.t. a, corroborated by a proof m (m is a proof for a that b is the author of m'): "If a *knew* m then a *would know that* once only b *knew* m'." (knowledge de re in the past subjunctive and then in the past indicative mode, and knowledge de dicto in the conditional mode). However, general-purpose/standard epistemic logic is inadequate in a cryptographic setting due to weak paradoxes, as is, for the same reason, (standard) deontic logic (cf. Section 3.2.3). (We recall that a *weak* paradox in a logic is a *counter-intuitive* statement in the logic, whereas a *strong* paradox is an *inconsistency* in the logic.) And doxastic logic is inadequate because the above requirements are ineffable in it, as these crucially

[3] as a matter of fact unique, canonical formulations of these requirements do not exist (yet)

rely on knowledge, i.e., necessarily true, and not possibly false, belief (no error control!). Our slogan, and pun[4], is:

Slogan 11 *Belief (without error control) can be used to show the presence of attacks, but, as opposed to knowledge, never to show their absence.*[5]

Further, linear logic has, for our approach, a flavour that is too operational to the extent that it is possible that "the combinators of a process calculus are mapped to [linear] logical connectives" [Mil06]. Our approach is diametric, i.e., we aim at providing declarative abstractions (the *what*) of operational aspects (the *how*). Finally, special-purpose logics have been limited in their adequacy due to their choice of primitive concepts, e.g., belief, no negation/quantification, too specific primitive concepts at the price of high extension costs.

Logical limitations originate in *design decisions* of syntactic (language-defining *operators*) and/or semantic (meaning-defining *notion of truth*) nature. The advantages (or disadvantages) of the cited logics are corollaries of the respective advantages (or disadvantages) of capturing (or not) the discussed and to-be-discussed concepts. In particular, crucial advantages are to capture: (1) individual and propositional knowledge, with a treatment of weak paradoxes; (2) permission and prohibition, with a treatment of weak paradoxes; (3) proof and provability; (4) protocol composition (either with dynamic/Hoare-logic constructs, or with spatial-logic constructs as in CPL); and (5) time (both qualitative and quantitative).

3.1.2.3 Requirements engineering—CPL

Our goal is to supply a formal *synthesis* of (mono-dimensional) concepts in a single, poly-dimensional[6] modal logic, namely CPL, that yields requirements that are *intuitive* but (syntactically) *abstract* w.r.t. particular conceptions of cryptography[7]. First, our belief, expressed as a slogan, is:

Slogan 12 *The formal method for any science is, ultimately, logic.*

Logic, as defined by a relation of satisfaction (*model*-theoretic approach[8], effectuated via *model-checking* [CGP99]) or a relation of deduction (*proof*-theoretic approach, effectuated via *automated theorem-proving* [Fit96]). Second, given that requirements engineering is mainly about meaning, i.e., understanding and formalising properties, we believe that a model-theoretic approach is, at least at a first stage, more suitable than a proof-theoretic approach. We argue that propositional and higher-order (at least beyond second order) logic, and set theory are unsuitable as front-end formalisms for requirements engineering. Propositional logic is simply too weak as a specification language but is well-suited for

[4] on the slogan "Program testing can be used to show the presence of bugs, but never to show their absence!" by Dijkstra

[5] this is the deeper reason for the well-known limitations of BAN-logic

[6] cf. [GKWZ03] for a research monograph on poly-dimensional modal logic, characterised in [BdRV01] as "... a branch of modal logic dealing with special relational structures in which the states, rather than being abstract entities, have some inner structure. ... Furthermore, the accessibility relations between these states are (partly) determined by this inner structure of the states."

[7] such logics are called *endogenous* (or *mono-modal*), as opposed to *exogenous* (or *poly-modal*)

[8] not to be confused with a model-*based formalism*

fully-automated, approximative verification. Higher-order logic and set theory may well be semantically sufficiently expressive; however, we opine that they are unsuitable for engineers in charge of capturing meaning of protocol requirements within an acceptable amount of time (i.e., financial cost per specification) and space (i.e., intelligibility of specifications). The intuitiveness of the specifications that a formalism yields is not just luxury, but the very (and difficult to distil) essence and a measure of its *pragmatics*, i.e., practical usefulness. Our slogan[9] is:

Slogan 13 *Logic for engineering necessarily is, possibly first-order, modal logic.*

Modal operators (*modalities*) are object-level abstractions of meta-level quantifiers. In effect, they eliminate variables (and the quantifiers that bind them) in logical (truth) languages as combinators do in programming (effect) languages, and delimit quantification to the relevant (i.e., accessible) parts of the interpretation structure. Their benefits are *intelligibility* of the expressed statement, and *effectiveness* and relative efficiency of truth establishment, respectively. The concept of a cryptographic protocol is very rich. A suitable formalism must organise and hard-wire/pre-compile this conceptual variety in its semantics and provide succinct and intuitive linguistic abstractions (syntax) for them. The resulting added value of such a formalism is empowerment of the engineer (speed-up of the mental process of requirements formalisation)[10], and more powerful tools (speed-up of model-checking and automated theorem-proving). Higher-order logic and set theory, having been conceived as general-purpose formalisms, obviously lack this special-purpose semantics and syntax. However, they are well-suited as logical frameworks (meta-logics/back-ends) for such special-purpose formalisms (object logics/front-ends). For example, our candidate language has a model-theoretic (i.e., relying on set theory) semantics.

CPL has a *first-order* fragment for making statements about protocol events and about the (individual) knowledge ("knows") and the structure of cryptographic messages induced by those events; and four *modal* fragments for making statements about confidentiality *norms* (cf. deontic logic [BC93]); propositional *knowledge* ("knows that"), i.e., knowledge of cryptographic states of affairs, (cf. epistemic logic [FHMV95]); execution *space* (cf. spatial logic [Dam89]); and execution *time* (cf. temporal logic [MP84]). That is, CPL *unifies* first-order and four modal logics in a single, first-order, poly-dimensional modal logic. Further, CPL *refines* standard epistemic and deontic logic in the sense that it resolves the long-standing problem of weak paradoxes (caused by logical omniscience and conflicting obligations, respectively) that these logics exhibit when applied in a cryptographic setting (cf. Section 3.2.3). Yet CPL (a property-based formalism) goes even further in its *wholistic ambition* in that it integrates the perhaps most important model-based framework, namely process algebra [BPS01], in a novel *co-design*. First, CPL's temporal accessibility relation (the semantics of its temporal modalities) *can* be defined by an event-trace generating process (reduction) calculus, *for example* C^3 (cf. Chapter 4) whose reduction constraints *can* moreover be checked via CPL-satisfaction; and second[11], CPL's epistemic

[9] and pun on the two cornerstones of modal logic, namely *possibility* and *necessity*

[10] in analogy with high-level programming languages versus machine-code languages

[11] This idea seems to have been published first in [HS04b]. However, the authors adopt a very different approach based on so-called function views.

accessibility relation (the semantics of its epistemic modality "knows that") is the definitional basis for C^3's observational equivalence, which can be used for the model-based (process-algebraic and complementary to property-based) formulation of protocol requirements. We believe that this co-design is also the key to a genuine modal model theory for cryptography.

Justification A cryptographic protocol involves the concurrent interaction of agents that are physically separated by — and exchange messages across — an unreliable and insecure transmission medium. Expressing properties of concurrent interaction (i.e., *interactive* computation) requires temporal modalities [MP84]. The physical separation by an unreliable and insecure transmission medium (i.e., *unreliable* computation) in turn demands the epistemic and deontic modalities. To see why, consider that the existence of such a separating medium introduces an *uncertainty* among agents about the *trustworthiness* of the execution of protocol actions (sending, receiving) and the contents of exchanged messages, both w.r.t. *actuality* (an epistemic concern) and *legitimacy* (a deontic concern).

Slogan 14 *Trustworthiness = Actuality + Legitimacy*

The purpose of a cryptographic protocol is to reëstablish this trustworthiness through the judicious use of *cryptographic evidence*, i.e., essential information (e.g., ciphers, signatures and hash values) for the knowledge of other information (e.g., messages or truth of formulae), bred in a *crypto system* (e.g., a shared-key or public-key system) from *cryptographic germs* such as keys and nonces, themselves generated from *cryptographic seeds* (or seed values). However, any use of keys (as opposed to hash values and nonces) requires that the knowledge of those keys be shared a priori. This sharing of key knowledge is established by cryptographic protocols called *key-establishment* protocols (comprising *key-transport* and *key-agreement* protocols) [MvOV96, Chapter 12], which are executed before any cryptographic protocol that may then subsequently use those keys. Thus certain cryptographic protocols must be considered interrelated by a notion of *composition* in a common execution *space*; hence the need of spatial operators. Another argument for spatial operators comes from the fact that a correct protocol should conserve its sole correctness even when composed with other protocols, i.e., a compositionally correct protocol should be *stable in different execution contexts* [Can01, BPW03].

3.2 Logic

3.2.1 Syntax

The language $\mathcal{F}$ of CPL is parametric in the language $\mathcal{M}$ of its individuals, i.e., cryptographic messages. It is chiefly relational, and functional in exactly the language $\mathcal{M}$ of cryptographic messages it may be instantiated with. The temporal fragment of $\mathcal{F}$ coincides with the syntax of LTLP (linear temporal logic with past). We shall fix our mind on the following, comprehensive language $\mathcal{M}$.

Definition 1 (Cryptographic messages) *We form messages $M \in \mathcal{M}$ with the term constructors displayed in Table 3.1. There, names $n \in \boldsymbol{\mathcal{N}}$ denote agent*

names $a, b, c \in \boldsymbol{\mathcal{A}}$*, the (for the moment Dolev-Yao [DY83]) adversary's name* `Eve`*, symmetric short-term (session)* ($\boldsymbol{\mathcal{K}}^1$) *and long-term* ($\boldsymbol{\mathcal{K}}^\infty$) *keys* $k \in \boldsymbol{\mathcal{K}}$*, (asymmetric) private keys* $p \in \boldsymbol{\mathcal{K}}^-$*, and nonces* $x \in \boldsymbol{\mathcal{X}}$ *(also used as session identifiers).*

We assume that given a private key p*, one can compute the corresponding public key* p^+*, as in DSA and Elgamal. Shared and private keys shall be referred to as* confidential keys $\boldsymbol{\mathcal{CK}}$*, i.e., keys that must remain secret. Symmetric keys may be* compound *for key* agreement *(as opposed to mere key* transport*). Message forms (open messages)* F *are messages with variables* $v \in \mathcal{V}$*.*

Table 3.1: Message language

M	::=	n	(names, i.e., logical constants)
	|	■	(the abstract message)
	|	p^+	(public keys)
	|	$\lceil M \rceil$	(message hashes)
	|	$\{\!\lvert M \rvert\!\}_M$	(symmetric message ciphers)
	|	$\{\!\lvert M \rvert\!\}^+_{p^+}$	(asymmmetric message ciphers)
	|	$\{\!\lvert M \rvert\!\}^-_p$	(signed messages)
	|	(M, M)	(message tuples)

We use the terms *privacy*, *confidentiality*, and *secrecy* to qualify cryptographic information w.r.t. the *legitimacy*, the *intention*, resp. the *actuality* of the knowledge of that information (status of discreetness). For example, in asymmetric-key cryptography, the knowledge of a key for decrypting or signing cryptographic information is limited to the discretion of a single entity, say a. Thus, such a key qualifies as the *private* key of entity a; and to-be-encrypted plaintext is by definition cryptographic information whose knowledge is intended to be limited to the discretion of the sender and the recipient(s), i.e., it is qualified *confidential* a priori, and may be qualified *secret* by vigour of verification a posteriori.

The abstract message is a computational artifice to represent the absence of *intelligibility*, just as the number zero is a computational artifice to represent the absence of *quantity*. The abstract message is very useful for doing knowledge-based calculations (cf. Definition 5), just as the number zero is very useful (to say the least) for doing number-based calculations.

The focus on cryptographic protocols rather than cryptographic operators leads us (for the moment) to (1) making abstraction from the exact representation of messages, e.g., bit strings; and assuming (2.1) *perfect hashing*, i.e., collision resistance (hash functions are injective) and strong pre-image resistance (hash functions are not invertible, or given $\lceil M \rceil$, it is infeasible to compute M), and (2.2) *perfect encryption* (given $\{\!\lvert M \rvert\!\}_k$ but not the shared key k or given $\{\!\lvert M \rvert\!\}^+_{p^+}$ but not the private key p corresponding to the public key p^+, it is infeasible to compute M).

We introduce a type language for messages to increase the succinctness of statements about the structure of messages.

Definition 2 (Message types) *Message types τ have the following structure.*

$$\begin{array}{lcl} \tau, \tau' & ::= & \emptyset \mid \sigma \mid \mathtt{H}[\tau] \mid \mathtt{SC}_M[\tau] \mid \mathtt{AC}_{p^+}[\tau] \mid \mathtt{S}_p[\tau] \mid \mathtt{T}[\tau, \tau'] \mid \tau \cup \tau' \mid \tau \cap \tau' \mid \tau \setminus \tau' \mid \mathtt{M} \\ \sigma, \sigma' & ::= & \mathtt{A} \mid \mathtt{Adv} \mid \varsigma \mid \mathtt{K}^+ \\ \varsigma, \varsigma' & ::= & \mathtt{K}^1 \mid \mathtt{K}^\infty \mid \mathtt{K}^- \mid \mathtt{X} \end{array}$$

Message type forms θ shall be message types with variables in key position.

Observe that (1) for each kind of message there is a corresponding type (e.g., $\mathtt{H}[\tau]$ for hashes, $\mathtt{SC}_M[\tau]$ for symmetric and $\mathtt{AC}_{p^+}[\tau]$ for asymmetric ciphers, $\mathtt{S}_p[\tau]$ for signatures, and $\mathtt{T}[\tau, \tau']$ for tuples); (2) encryption and signature types are parametric; and (3) the union, intersection, and difference of two message types is again a message type. In short, message types are structure-describing *dependent types* closed under union, intersection, and difference. ς and ς' denote types of dynamically generable names. We macro-define $\mathtt{A}_{\mathtt{Adv}} := \mathtt{A} \cup \mathtt{Adv}$, $\mathtt{K} := \mathtt{K}^1 \cup \mathtt{K}^\infty$, $\mathtt{CK} := \mathtt{K} \cup \mathtt{K}^-$, $\mathtt{K}^* := \mathtt{CK} \cup \mathtt{K}^+$, and $\mathtt{N} := \mathtt{A}_{\mathtt{Adv}} \cup \mathtt{K}^* \cup \mathtt{X}$.

Definition 3 (Logical formulae) *The set of formulae $\mathcal{F}$ contains precisely those propositions that are the closed predicates formed with the sentence constructors displayed in Table 3.2. There, β denotes basic, α action, and δ data formulae; and o denotes tuples of agent names (key owners).*

Table 3.2: Predicate language

$$\begin{array}{lcl} \phi, \phi' & ::= & \beta \mid \neg\phi \mid \phi \wedge \phi' \mid \forall v(\phi) \\ & \mid & \underbrace{\mathsf{P}\phi}_{\text{norms}} \mid \underbrace{\mathsf{K}_a(\phi) \mid \phi \sqsupseteq \phi'}_{\text{knowledge}} \mid \underbrace{\phi \otimes \phi' \mid \phi \rhd \phi'}_{\text{space}} \mid \underbrace{\phi \,\mathsf{S}\, \phi' \mid \ominus\phi \mid \oplus\phi \mid \phi \,\mathsf{U}\, \phi'}_{\text{time}} \\ \beta, \beta' & ::= & \alpha \mid \delta \\ \alpha, \alpha' & ::= & a \circlearrowleft n.o \mid \underbrace{a \xrightarrow[\text{Eve}]{F} b \mid a \xleftarrow[\text{Eve}]{F} b}_{\text{private comm.}} \mid \underbrace{a \xrightarrow[\text{Eve}]{F} b \mid a \xleftarrow[\text{Eve}]{F} F}_{\text{public comm.}} \\ \delta, \delta' & ::= & n : \sigma \mid a \,\mathsf{k}\, F \mid F \preccurlyeq F' \mid a@x \end{array}$$

Predicates can be transformed into propositions either via binding of free variables, i.e., universal (*generalisation*) or existential (*abstraction*) quantification, or via substitution of individuals for free variables (*individuation*). In accordance with standard logical methodology, basic predicates express *elementary facts*[12].

Our symbols are — and their intuitive meaning is as they are — pronounced $\neg$ "not", $\wedge$ "and", $\forall v$ "for all v", P "it is permitted that", K_a "a knows that", $\sqsupseteq$ "epistemically implies", $\otimes$ "conjunctively separates", $\rhd$ "assume—guarantee", S "since", $\ominus$ "previous", $\oplus$ "next", U "until", $a \circlearrowleft n.o$ "a freshly generated the name n for owner(s) o", $a \xrightarrow[\text{Eve}]{F} b$ "a securely (i.e., over some private channel) sent F *as such* (i.e., not only as a strict sub-term of another message) to b",

[12] a fact is a contingent (particular) truth as opposed to a logical (universal) truth

$a \underset{\not\text{Eve}}{\overset{F}{\leftarrow}} b$ "a securely received F as such from b", $a \underset{\text{Eve}}{\overset{F}{\rightarrow}} b$ "a insecurely (i.e., over some public channel) sent off F as such to b", $a \underset{\text{Eve}}{\leftarrow} F$ "a insecurely received F as such", : "has type", k "knows", $\preccurlyeq$ "is a subterm of", and @ "is in protocol run/session".

Our predicate language is *1-sorted* thanks to the standard technique of *sort reduction*[13] and to the fact that agents are referred to by their name and names are transmittable data, i.e., messages.

The modality K expresses *propositional* knowledge, i.e., the knowledge that a certain proposition is true. In contrast, the relational symbol k expresses *individual* knowledge. Individual knowledge conveys understanding of the purpose and possession of a certain piece of cryptographic information up to cryptographically irreducible parts. It is established based on the capability of agents to synthesise those pieces from previously analysed pieces. By 'understanding of the purpose' we mean (1) knowledge of the *structure* for compound, and (2) knowledge of the *identity* for atomic (names) information. Note that such understanding requires that there be a *minimal redundancy* in that information. The conditional $\phi \sqsupseteq \phi'$ is epistemic in the sense that the set of evidence corroborating truth of the consequent ϕ' (e.g., the knowledge of a key) is *included* in the set of evidence corroborating truth of the antecedent ϕ (e.g., the knowledge of a plaintext *derived* from that key). The epistemic conditional captures the epistemic dependence of the truth of the antecedent on the truth of the consequent.

The formula $\phi \otimes \phi'$ is satisfied by a (protocol) model if and only if the model can be separated in exactly two parts such that one part satisfies ϕ (e.g., key establishment/production) and the other satisfies ϕ' (e.g., key use/consumption). The spatial conditional $\phi \rhd \phi'$ is satisfied by a model if and only if for all models that satisfy ϕ the adjunction of the second to the first model satisfies ϕ' (cf. compositional correctness of a protocol, as mentioned earlier).

Typing formulae $F : \theta$ have an essential and a pragmatic purpose. Typing of *atomic* data, i.e., when F designates a name n and θ an atomic type σ, is a linguistic abstraction for the above-mentioned essential modelling hypothesis of minimal redundancy. Typing of *compound* data simply increases succinctness of statements about message structure. It is actually macro-definable in terms of typing of atomic data, equality (itself macro-definable), and existential quantification (cf. Appendix B).

3.2.2 Semantics

Our definition of satisfaction[14] is *anchored* (or *rooted*) and defined on protocol states, i.e., tuples $(\mathfrak{h}, P) \in \mathcal{H} \times \mathcal{P}$ of a protocol model P (i.e., a process term of parallel-composable, located threads $a.x[\,T\,]$) and a protocol history $\mathfrak{h}$ (i.e., a trace of past protocol events). Note that *history-dependency* is characteristic of interactive computation [GSW06].

The logically-inclined reader will notice that CPL has a *Herbrand-semantics*, i.e., logical constants and functional symbols are self-interpreted rather than interpreted in terms of (other, semantic) constants and functions.

[13] introduction of unary relational symbols ($\cdot : \sigma$ in our case) emulating the different sorts

[14] the concept was invented by Tarski

Slogan 15 (Symbolic Foundationalism) *Meaning, when communicable, is symbolic. Semantics is interpretation of syntax via rewriting.*

For the present purpose, we presuppose a notion of execution, *for example* the one of Section 4.1, $\longrightarrow \subseteq (\mathcal{H} \times \mathcal{P})^2$ (or relation of *temporal accessibility* in the jargon of modal logic) producing protocol events of a certain kind and chaining them up to form protocol histories. We stress that the locality and parallel-composability of processes (denoted $P \,|||\, P'$), and the kind of protocol events are the only particularities of $\longrightarrow$ that we presuppose.

Protocol events are of the following kind: generation of a name n for owners o (recall that o is a tuple of agent names) in session x by a, written $\mathtt{N}(a, x, n, o)$; insecure input of M by a, written $\mathtt{I}(a, x, M)$; secure input of M from b by a, written $\mathtt{sI}(a, x, M, b)$; insecure output of M to b by a, written $\mathtt{O}(a, x, M, b)$; and secure output of M to b by a, written $\mathtt{sO}(a, x, M, b)$. By definition, an event ε is secure if and only if ε is unobservable by the adversary `Eve`. By convention, name generation is a secure event. We write $\varepsilon(a)$ for any of the above protocol events, $\varepsilon(a, n)$ for any of the above name-generation events, $\varepsilon(a, M)$ for any of the above communication events, and $\hat{\varepsilon}(a)$ for any of the above secure events. Protocol histories $\mathfrak{h} \in \mathcal{H}$ are simply *finite words* of protocol events ε, i.e., event traces $\mathfrak{h} ::= \epsilon \mid \mathfrak{h} \cdot \varepsilon$, where ϵ designates the empty protocol history.

We define satisfaction in a functional style (with a function of truth denotation) on the structure of formulae. Satisfaction employs *complex* (and thus multiple[15]) truth values. Truth values are complex in the sense that they are tuples of a simple truth value (i.e., 'true' or 'false') and a set of those events (the evidence/witnesses) that corroborate that simple truth.

Note that the definition employs macro-defined predicates (cf. Appendix B; the reader is urged to consult it).

Definition 4 (Satisfaction) *Let $\models \subseteq (\mathcal{H} \times \mathcal{P}) \times \mathcal{F}$ designate satisfaction of a formula $\phi \in \mathcal{F}$ by a protocol state $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$ (the anchor/root of an implicit execution path model[16] for ϕ):*

$$\begin{array}{rll} \mathfrak{s} \models \phi & \textit{:iff} & \textit{there is a set } \mathcal{E} \textit{ of protocol events s.t. } \mathfrak{s} \models_{\mathcal{E}} \phi \\ \mathfrak{s} \models_{\mathcal{E}} \phi & \textit{:iff} & \textit{for all } \mathfrak{p} \in \mathrm{paths}(\mathfrak{s}),\ [\![\phi]\!]^0_{\mathfrak{p}} = (\textit{true}, \mathcal{E}) \end{array}$$

where $\mathrm{paths}(\mathfrak{s}) := \{\ \mathfrak{p} \mid \mathfrak{p}@0 = \mathfrak{s}$ *and for all* $i < |\mathfrak{p}|$, $\mathfrak{p}@0 \longrightarrow^* \mathfrak{p}@i\ \}$ *designates the set of paths* $\mathfrak{p}$ *achored/rooted in* $\mathfrak{s}$ *and induced by* $\longrightarrow$, *and* $[\![\cdot]\!]$ *designates our function of truth denotation from formulae to complex truth values (cf. Table 3.4). There,*

- $\mathfrak{p}@i$ *designates the state, say—please memorise—*$(\mathfrak{h}, P)$, *at position* i *in* $\mathfrak{p}$
- $\dot{\mathfrak{h}}$ *designates the* set *of events derived from the* trace *of events* $\mathfrak{h}$
- $\mathfrak{h} \vdash^{\mathcal{E}}_{a} M$ *designates derivation of* M *by* a *from—this is a novel idea—the set* $\mathcal{E}$ *of events in* a*'s view on* $\mathfrak{h}$, *i.e., the extraction, analysis, and synthesis of the data that* a *has generated, received, or sent in* $\mathfrak{h}$ *(cf. Table 3.3)*[17]

[15] multi-valued logic was invented by Post

[16] notice the two notions of a model: namely, the one of a model *for* a logical formula (i.e., a protocol state $(\mathfrak{h}, P)$), and the one of a model *of* a cryptographic protocol (i.e., a process term P)

[17] We could easily account for individual knowledge *modulo an equational theory* of cryp-

Table 3.3: Derivation of individual knowledge

Data extraction

$$\frac{}{\mathfrak{h} \cdot \varepsilon(a, \overline{M}) \vdash_a^{\{\varepsilon(a,\overline{M})\}} (a, \overline{M})} \qquad \frac{\mathfrak{h} \vdash_a^{\mathcal{E}} M}{\mathfrak{h} \cdot \varepsilon \vdash_a^{\mathcal{E}} M}$$

Data synthesis ***Data analysis***

$$\frac{\mathfrak{h} \vdash_a^{\mathcal{E}} M \quad \mathfrak{h} \vdash_a^{\mathcal{E}'} M'}{\mathfrak{h} \vdash_a^{\mathcal{E}\cup\mathcal{E}'} (M, M')} \qquad \frac{\mathfrak{h} \vdash_a^{\mathcal{E}} (M, M')}{\mathfrak{h} \vdash_a^{\mathcal{E}} M} \qquad \frac{\mathfrak{h} \vdash_a^{\mathcal{E}} (M, M')}{\mathfrak{h} \vdash_a^{\mathcal{E}} M'}$$

$$\frac{\mathfrak{h} \vdash_a^{\mathcal{E}} p}{\mathfrak{h} \vdash_a^{\mathcal{E}} p^+} \qquad \frac{\mathfrak{h} \vdash_a^{\mathcal{E}} M}{\mathfrak{h} \vdash_a^{\mathcal{E}} \lceil M \rceil}$$

$$\frac{\mathfrak{h} \vdash_a^{\mathcal{E}} M \quad \mathfrak{h} \vdash_a^{\mathcal{E}'} M'}{\mathfrak{h} \vdash_a^{\mathcal{E}\cup\mathcal{E}'} \{\!|M|\!\}_{M'}} \qquad \frac{\mathfrak{h} \vdash_a^{\mathcal{E}} \{\!|M|\!\}_{M'} \quad \mathfrak{h} \vdash_a^{\mathcal{E}'} M'}{\mathfrak{h} \vdash_a^{\mathcal{E}\cup\mathcal{E}'} M}$$

$$\frac{\mathfrak{h} \vdash_a^{\mathcal{E}} M \quad \mathfrak{h} \vdash_a^{\mathcal{E}'} p^+}{\mathfrak{h} \vdash_a^{\mathcal{E}\cup\mathcal{E}'} \{\!|M|\!\}^+_{p^+}} \qquad \frac{\mathfrak{h} \vdash_a^{\mathcal{E}} \{\!|M|\!\}^+_{p^+} \quad \mathfrak{h} \vdash_a^{\mathcal{E}'} p}{\mathfrak{h} \vdash_a^{\mathcal{E}\cup\mathcal{E}'} M}$$

$$\frac{\mathfrak{h} \vdash_a^{\mathcal{E}} M \quad \mathfrak{h} \vdash_a^{\mathcal{E}'} p}{\mathfrak{h} \vdash_a^{\mathcal{E}\cup\mathcal{E}'} \{\!|M|\!\}^-_{p}} \qquad \frac{\mathfrak{h} \vdash_a^{\mathcal{E}} \{\!|M|\!\}^-_{p} \quad \mathfrak{h} \vdash_a^{\mathcal{E}'} p^+}{\mathfrak{h} \vdash_a^{\mathcal{E}\cup\mathcal{E}'} M}$$

- $\circ$ *designates concatenation of histories conserving uniqueness of events*
- $\Sigma := \exists(k : \mathtt{CK})(\mathtt{Eve}\ \mathsf{k}\ k \wedge \neg\, k\ \mathsf{ck}\ \mathtt{Eve})$ *designates a state formula expressing the* state of violation *in a Dolev-Yao adversarial setting, namely the one where the adversary has come to know a confidential key not of her own*
- $\approx_a \subseteq (\mathcal{H} \times \mathcal{P})^2$ *designates the relation of* epistemic accessibility *associated with the modality* K_a*; it is defined hereafter*
- $(\!|\cdot|\!)^{\mathfrak{h}}_a$ *designates a unary function (inspired by [AR02]) of* cryptographic parsing *defined on protocol states and on logical formulae; it is defined hereafter on messages and tacitly lifted onto protocol states and logical formulae*
- $\equiv$ *designates a relation of* structural equivalence *defined on process terms and on event traces. On process terms, it designates the smallest equivalence relation expressing associativity and commutativity of processes. On event traces, it designates* shuffling.

The permission modality is included in the logic because we want to highlight that each new notion of state of violation will give rise to a new notion of permission, such as the one for real time (cf. Section 3.4.2) or the ones for

---

tographic messages, i.e., a set of algebraic properties of cryptographic operators expressed with an equivalence relation $\equiv \subseteq \mathcal{M} \times \mathcal{M}$, by adding a rule $\dfrac{\mathfrak{h} \vdash_a^{\mathcal{E}} M}{\mathfrak{h} \vdash_a^{\mathcal{E}} M'}\ M \equiv M'$. Further, *agent-name guessing* could be modelled by adding an axiom $\dfrac{}{\mathfrak{h} \vdash_a^{\mathcal{E}} b}\ b \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$.

Table 3.4: Truth denotation

$$
\begin{array}{rcl}
[\![a \circlearrowright n.o]\!]^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{\mathcal{X}}} \{\mathtt{N}(a,x,n,o)\} \cap \dot{\mathfrak{h}} \\
[\![a \xrightarrow[\text{Eve}]{M} b]\!]^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{\mathcal{X}}} \{\mathtt{sO}(a,x,M,b)\} \cap \dot{\mathfrak{h}} \\
[\![a \xleftarrow[\text{Eve}]{M} b]\!]^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{\mathcal{X}}} \{\mathtt{sI}(a,x,M,b)\} \cap \dot{\mathfrak{h}} \\
[\![a \xrightarrow[\text{Eve}]{M} b]\!]^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{\mathcal{X}}} \{\mathtt{O}(a,x,M,b)\} \cap \dot{\mathfrak{h}} \\
[\![a \xleftarrow[\text{Eve}]{} M]\!]^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{\mathcal{X}}} \{\mathtt{I}(a,x,M)\} \cap \dot{\mathfrak{h}} \\
[\![n : \sigma]\!]^i_{\mathfrak{p}} & := & (n \text{ has type } \sigma, \emptyset) \\
[\![a \mathsf{k} M]\!]^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup\{\ \mathcal{E}' \mid \mathfrak{h} \vdash^{\mathcal{E}'}_a M\ \} \\
[\![M \preccurlyeq M']\!]^i_{\mathfrak{p}} & := & (M \text{ is a subterm of } M', \emptyset) \\
[\![a@x]\!]^i_{\mathfrak{p}} & := & (\text{there is a thread } T \text{ s.t. } P = a.x[\,T\,] \text{ and } \mathfrak{h} = \mathfrak{h}|_{a.x}, \emptyset) \\
[\![\neg\phi]\!]^i_{\mathfrak{p}} & := & (\text{not } \mathsf{v}_\phi, \dot{\mathfrak{h}} \setminus \mathcal{E}_\phi) \quad \text{where } [\![\phi]\!]^i_{\mathfrak{p}} = (\mathsf{v}_\phi, \mathcal{E}_\phi) \\
[\![\phi \wedge \phi']\!]^i_{\mathfrak{p}} & := & (\mathsf{v}_\phi \text{ and } \mathsf{v}_{\phi'}, \mathcal{E}_\phi \cup \mathcal{E}_{\phi'}) \quad \text{where } \begin{array}{l}[\![\phi]\!]^i_{\mathfrak{p}} = (\mathsf{v}_\phi, \mathcal{E}_\phi) \text{ and} \\ [\![\phi']\!]^i_{\mathfrak{p}} = (\mathsf{v}_{\phi'}, \mathcal{E}_{\phi'})\end{array} \\
[\![\forall v(\phi)]\!]^i_{\mathfrak{p}} & := & (\text{for all } M \in \mathcal{M},\ \mathsf{v}_M, \bigcup_{M \in \mathcal{M}} \mathcal{E}_M) \quad \text{where } [\![\{^M/_v\}\phi]\!]^i_{\mathfrak{p}} = (\mathsf{v}_M, \mathcal{E}_M) \\
[\![\mathsf{P}\phi]\!]^i_{\mathfrak{p}} & := & [\![(\phi \wedge \neg \diamondplus \Sigma) \rhd \boxplus(\Sigma \to (\Sigma \not\trianglerighteq \phi))]\!]^i_{\mathfrak{p}} \\
[\![\mathsf{K}_a(\phi)]\!]^i_{\mathfrak{p}} & := & (\text{for all } \mathfrak{s}, \text{ if } \ \mathfrak{p}@0 \longrightarrow^* \mathfrak{s} \text{ and } \mathfrak{s} \approx_a \mathfrak{p}@i \ \text{ then } \ \mathfrak{s}' \models_{\mathcal{E}'} \phi', \mathcal{E}'_{(\mathfrak{s},\phi)}) \\
 & & \text{where } (\mathfrak{s}', \phi') := \begin{cases} (\mathfrak{s}, \phi) & \text{if } \mathfrak{s} = \mathfrak{p}@i, \text{ and} \\ (\langle\!|\, \mathfrak{s} \,|\!\rangle^{\mathfrak{h}}_a, \langle\!|\, \phi \,|\!\rangle^{\mathfrak{h}}_a) & \text{otherwise.} \end{cases} \\
[\![\phi \sqsupseteq \phi']\!]^i_{\mathfrak{p}} & := & (\underline{\text{if } \mathsf{v}_\phi \text{ then } \mathsf{v}_{\phi'}} \text{ and } \underline{\emptyset \neq \mathcal{E}_{\phi'} \subseteq \mathcal{E}_\phi}, \mathcal{E}_\phi) \quad \text{where } \begin{array}{l}[\![\phi]\!]^i_{\mathfrak{p}} = (\mathsf{v}_\phi, \mathcal{E}_\phi) \text{ and} \\ [\![\phi']\!]^i_{\mathfrak{p}} = (\mathsf{v}_{\phi'}, \mathcal{E}_{\phi'})\end{array} \\
[\![\phi \otimes \phi']\!]^i_{\mathfrak{p}} & := & (\text{there are } Q, Q' \in \mathcal{P} \text{ and } \mathfrak{h}', \mathfrak{h}'' \in \mathcal{H} \text{ s.t. } P \equiv Q \,|||\, Q' \text{ and } \mathfrak{h} \equiv \mathfrak{h}' \circ \mathfrak{h}'' \\
 & & \text{and } (\mathfrak{h}', Q) \models_{\mathcal{E}_\phi} \phi \text{ and } (\mathfrak{h}'', Q') \models_{\mathcal{E}_{\phi'}} \phi',\ \mathcal{E}_\phi \cup \mathcal{E}_{\phi'}) \\
[\![\phi \rhd \phi']\!]^i_{\mathfrak{p}} & := & (\text{for all } (\mathfrak{h}', Q) \in \mathcal{H} \times \mathcal{P} \text{ and } \mathfrak{h}'' \equiv \mathfrak{h}' \circ \mathfrak{h}, \text{ if } \ (\mathfrak{h}', Q) \models_{\mathcal{E}'} \phi \ \text{ then} \\
 & & (\mathfrak{h}'', Q \,|||\, P) \models_{\mathcal{E}''} \phi', \bigcup \mathcal{E}'' \cup \bigcup \mathcal{E}') \\
[\![\phi \,\mathsf{S}\, \phi']\!]^i_{\mathfrak{p}} & := & (\text{there is } k \text{ s.t. } 0 \leq k \leq i \text{ and } \mathsf{v}_k \text{ and for all } j, \text{ if } k < j \leq i \text{ then } \mathsf{v}_j, \\
 & & \textstyle\bigcup_j \mathcal{E}_j \cup \mathcal{E}_k) \quad \text{where } [\![\phi]\!]^j_{\mathfrak{p}} = (\mathsf{v}_j, \mathcal{E}_j) \text{ and } [\![\phi']\!]^k_{\mathfrak{p}} = (\mathsf{v}_k, \mathcal{E}_k) \\
[\![\ominus\phi]\!]^i_{\mathfrak{p}} & := & \begin{cases} [\![\phi]\!]^{i-1}_{\mathfrak{p}} & \text{if } i > 0, \text{ and} \\ (\text{false}, \emptyset) & \text{otherwise.} \end{cases} \\
[\![\oplus\phi]\!]^i_{\mathfrak{p}} & := & \begin{cases} [\![\phi]\!]^{i+1}_{\mathfrak{p}} & \text{if } i < |\mathfrak{p}| - 1, \text{ and} \\ (\text{false}, \emptyset) & \text{otherwise.} \end{cases} \\
[\![\phi \,\mathsf{U}\, \phi']\!]^i_{\mathfrak{p}} & := & (\text{there is } k \text{ s.t. } i \leq k \text{ and } \mathsf{v}_k \text{ and for all } j, \text{ if } i \leq j < k \text{ then } \mathsf{v}_j, \\
 & & \textstyle\bigcup_j \mathcal{E}_j \cup \mathcal{E}_k) \quad \text{where } [\![\phi]\!]^j_{\mathfrak{p}} = (\mathsf{v}_j, \mathcal{E}_j) \text{ and } [\![\phi']\!]^k_{\mathfrak{p}} = (\mathsf{v}_k, \mathcal{E}_k)
\end{array}
$$

probabilistic polynomial-time settings (cf. Section 5.2.2). That is, we look at the state formula Σ as a parameter of the logic.

The epistemic accessibility relation has, as previously mentioned, a double use. It not only serves as the definitional basis for the epistemic modality of CPL, but also as the definitional basis for the observational equivalence of C^3 (cf. Section 4.1.3).

Notice that the spatial conditional is monotonic w.r.t. positive antecedents, e.g., $\models a \mathsf{k} M \rhd a \mathsf{k} M$, but $\not\models \neg a \mathsf{k} M \rhd \neg a \mathsf{k} M$ due to the possibility for a of learning additional information from the adjunction.

Cryptographic parsing captures an agent's capability to understand the structure of a cryptographically obfuscated message. It allows the definition of a cryptographically meaningful notion of epistemic accessibility via the intermediate concept of structurally indistinguishable protocol histories. The idea is to parse unintelligible messages to the abstract message $\blacksquare$.

Definition 5 (Cryptographic parsing) *The cryptographic parsing function $(\!|\cdot|\!)^{\mathfrak{h}}_a$ associated with an agent $a \in \mathcal{P}$ and a protocol history $\mathfrak{h} \in \mathcal{H}$ (and complying with the assumptions of perfect cryptography) is an identity on names, the abstract message, and public keys; and otherwise acts as defined in Table 3.5.*

Table 3.5: Parsing cryptographic messages

$$
\begin{array}{rcl}
(\!|\ulcorner M \urcorner|\!)^{\mathfrak{h}}_a & := & \begin{cases} \ulcorner (\!|M|\!)^{\mathfrak{h}}_a \urcorner & \text{if } \mathfrak{h} \models a \mathsf{k} M, \text{ and} \\ \blacksquare & \text{otherwise.} \end{cases} \\
(\!|\{\!|M|\!\}_{M'}|\!)^{\mathfrak{h}}_a & := & \begin{cases} \{\!|(\!|M|\!)^{\mathfrak{h}}_a|\!\}_{(\!|M'|\!)^{\mathfrak{h}}_a} & \text{if } \mathfrak{h} \models a \mathsf{k} M', \text{ and} \\ \blacksquare & \text{otherwise.} \end{cases} \\
(\!|\{\!|M|\!\}^{+}_{p^+}|\!)^{\mathfrak{h}}_a & := & \begin{cases} \{\!|(\!|M|\!)^{\mathfrak{h}}_a|\!\}^{+}_{p^+} & \text{if } \mathfrak{h} \models a \mathsf{k} p \vee (a \mathsf{k} M \wedge a \mathsf{k} p^+), \text{ and} \\ \blacksquare & \text{otherwise.} \end{cases} \\
(\!|\{\!|M|\!\}^{-}_{p}|\!)^{\mathfrak{h}}_a & := & \begin{cases} \{\!|(\!|M|\!)^{\mathfrak{h}}_a|\!\}^{-}_{p} & \text{if } \mathfrak{h} \models a \mathsf{k} p^+, \text{ and} \\ \blacksquare & \text{otherwise.} \end{cases} \\
(\!|(M, M')|\!)^{\mathfrak{h}}_a & := & ((\!|M|\!)^{\mathfrak{h}}_a, (\!|M'|\!)^{\mathfrak{h}}_a)
\end{array}
$$

A particularity of this notion of cryptographic parsing is that if $\mathfrak{h} \not\models a \mathsf{k} k$ and $\mathfrak{h}' \not\models a \mathsf{k} k$ then $(\!|\{\!|M|\!\}_k|\!)^{\mathfrak{h}}_a = \blacksquare = (\!|\{\!|M'|\!\}_k|\!)^{\mathfrak{h}'}_a$. That is, two different plaintexts (M and M') encrypted under the same symmetric key (k) are parsed to the same (abstract) message ($\blacksquare$), when the parsing agent does not know the decrypting key. This is justified by the fact that in reality, and in an extension of CPL with a notion of probabilistic (polynomial-time) computation (cf. Chapter 5), encryption is probabilistic anyway, which has precisely the effect of rendering the above ciphers (computationally) indistinguishable to a parsing agent.

Definition 6 (Structurally indistinguishable protocol histories) *Two protocol histories $\mathfrak{h}$ and $\mathfrak{h}'$ are* structurally indistinguishable from the viewpoint of *an agent a, written $\mathfrak{h} \approx_a \mathfrak{h}'$, :iff a observes the same* event pattern *and the same* data patterns *in $\mathfrak{h}$ and $\mathfrak{h}'$. Formally, for all $\mathfrak{h}, \mathfrak{h}' \in \mathcal{H}$, $\mathfrak{h} \approx_a \mathfrak{h}'$:iff $\mathfrak{h} \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}'$ where,*

- *given that a is a legitimate agent or the adversary* `Eve`*,*

  1. $\dfrac{}{\epsilon \approx_a^{(\mathfrak{h},\mathfrak{h}')} \epsilon}$

  2. $\dfrac{\mathfrak{h}_l \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \varepsilon(a,n) \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \varepsilon(a,n)}$

  3. $\dfrac{\mathfrak{h}_l \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \varepsilon(a,M) \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \varepsilon(a,M')}\ (\!|\, M \,|\!)_a^{\mathfrak{h}} = (\!|\, M' \,|\!)_a^{\mathfrak{h}'}$

- *given that a is a legitimate agent,*

  4. $\dfrac{\mathfrak{h}_l \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \varepsilon(b) \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}\ a \neq b \qquad \dfrac{\mathfrak{h}_l \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \varepsilon(b)}\ a \neq b$

- *given that a is the adversary* `Eve`*,*

  5. $\dfrac{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \hat{\varepsilon}(b) \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}\ \mathtt{Eve} \neq b \qquad \dfrac{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \hat{\varepsilon}(b)}\ \mathtt{Eve} \neq b$

  6. $\dfrac{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \mathtt{I}(b,x,M) \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \mathtt{I}(b,x,M')}\ (\!|\, M \,|\!)_{\mathtt{Eve}}^{\mathfrak{h}} = (\!|\, M' \,|\!)_{\mathtt{Eve}}^{\mathfrak{h}'}$

  7. $\dfrac{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \mathtt{O}(b,x,M,c) \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \mathtt{O}(b,x,M',c)}\ (\!|\, M \,|\!)_{\mathtt{Eve}}^{\mathfrak{h}} = (\!|\, M' \,|\!)_{\mathtt{Eve}}^{\mathfrak{h}'}$

Note that the observations at the different (past) stages $\mathfrak{h}_l$ and $\mathfrak{h}_r$ in $\mathfrak{h}$ and $\mathfrak{h}'$, respectively, must be made with the whole (present) knowledge of $\mathfrak{h}$ and $\mathfrak{h}'$ (cf. $\mathfrak{h}_l \approx^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r$). Learning new keys may render intelligible past messages to an agent a in the present that were not to her before.

Remark 1 *For all $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, $\approx_a \subseteq \mathcal{H} \times \mathcal{H}$ is*

1. *an equivalence with an infinite index due to fresh-name generation*
2. *not a right-congruence due to the possibility of learning new keys*
3. *a refinement on the projection $\mathcal{H}|_a$ of $\mathcal{H}$ onto a's view [FHMV95]*
4. *decidable.*

We lift structural indistinguishability from protocol histories to protocol states, i.e., tuples of a protocol term and a protocol history, and finally obtain our relation of epistemic accessibility.

Definition 7 (Observationally equivalent protocol states) *Let P_1 and P_2 designate two cryptographic processes, i.e., models of cryptographic protocols, of some set $\mathcal{P}$. Then two protocol states $(\mathfrak{h}_1, P_1)$ and $(\mathfrak{h}_2, P_2)$ are* observationally equivalent from the viewpoint of *an agent a, written $(\mathfrak{h}_1, P_1) \approx_a (\mathfrak{h}_2, P_2)$, :iff $\mathfrak{h}_1 \approx_a \mathfrak{h}_2$.*

Proposition 1 *Let for all $\phi, \phi' \in \mathcal{F}$,*

- *$[\![\phi]\!] = [\![\phi']\!]$:iff for all $\mathfrak{p}$ and i, $[\![\phi]\!]^i_{\mathfrak{p}} = [\![\phi']\!]^i_{\mathfrak{p}}$*
- *$\models \phi$, pronounced "ϕ is a* logical truth *(or* tautology*) in CPL", :iff for all $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$, $\mathfrak{s} \models \phi$.*

Then for all $\phi, \phi' \in \mathcal{F}$,

1. *if $\models \phi \equiv \phi'$ then $[\![\phi]\!] = [\![\phi']\!]$*
2. *if $[\![\phi]\!] = [\![\phi']\!]$ then $\models \phi \leftrightarrow \phi'$.*

Proof. almost by definition

Definition 8 (Logical consequence and equivalence) *Let $\phi, \phi' \in \mathcal{F}$. Then,*

- *ϕ' is a* logical consequence *of ϕ, written $\phi \Rightarrow \phi'$, :iff for all $\mathfrak{s} \in \mathcal{P} \times \mathcal{H}$, if $\mathfrak{s} \models \phi$ then $\mathfrak{s} \models \phi'$.*
- *ϕ' is* logically equivalent *to ϕ, written $\phi \Leftrightarrow \phi'$, :iff $\phi \Rightarrow \phi'$ and $\phi' \Rightarrow \phi$.*

Remark 2 *$\phi \supseteq \phi' \Rightarrow \phi \rightarrow \phi'$ but $\phi \rightarrow \phi' \not\Rightarrow \phi \supseteq \phi'$.*

Definition 9 (CPL: logic and logical theory)

- logic *(a body of truth):*

$$\mathcal{CPL} := \{\ \phi \mid \models \phi\ \}$$

- logical theory *(a system of inference):*

$$\mathfrak{CPL} := \{\ \phi \mid \textit{there is } \mathfrak{s} \in \mathcal{H} \times \mathcal{P} \textit{ s.t. } \mathfrak{s} \models \phi\ \}$$

(For all $\phi \in \mathfrak{CPL}$ and $\phi' \in \mathcal{F}$, if $\phi \Rightarrow \phi'$ then $\phi' \in \mathfrak{CPL}$.)

Theorem 1 (Barcan and relativised Co-Barcan)

Barcan $\models \forall m(\mathsf{K}_a(\phi)) \rightarrow \mathsf{K}_a(\forall m(\phi))$

Co-Barcan $\models \mathsf{K}_a(\forall m(a\ \mathsf{k}\ m \rightarrow \phi)) \rightarrow \forall m(a\ \mathsf{k}\ m \rightarrow \mathsf{K}_a(\phi))$

Proof. see Appendix A

The Barcan property w.r.t. propositional knowledge (K_a) is quite standard. However, the *relativisation to individual knowledge* (k) to obtain the converse Barcan property is novel. (The plain converse Barcan property obviously does not hold in a cryptographic context.)

Corollary 1 $\models \mathsf{K}_a(\forall m(a\ \mathsf{k}\ m \rightarrow \phi)) \leftrightarrow \forall m(a\ \mathsf{k}\ m \rightarrow \mathsf{K}_a(\phi))$

In words: propositional knowledge commutes with universal (and analogously with existential) quantification when that quantification is relativised to (or: guarded by) individual knowledge.

Remark 3 *CPL-satisfaction ("model-checking") is undecidable, as secrecy, being CPL-definable (cf. Section 3.3.2), is.*

3.2.3 Discussion

3.2.3.1 Expressiveness

The undecidability of the model-checking problem of CPL is intriguing because CPL is overtly first-order and the model-checking problem of plain first-order logic *is* decidable, in fact PSPACE-complete[18] [BvBW07]. The deeper reason for this intriguing state of affairs is that CPL is actually covertly *weak* second-order! To see why, consider that the truth condition of the spatial conditional ($\triangleright$) involves universal quantification over (adjoint) protocols, which, and that is the reason, generate via their execution *finite sets* of messages (CPL's official first-order individuals). The implicit (at the meta-level) and indirect (via the spatial conditional and protocols) universal quantification over finite sets of individuals *induces* weak second-order expressiveness.[19] Regarding secrecy, we will see in Section 3.3.2 that the source of its undecidability is nicely pinpointed by the prohibition (negated permission) modality, which employs the (negated) spatial conditional, required for its formalisation. In CPL, weak second-order expressiveness is available on demand of the spatial conditional and remains nicely confined to the use of that conditional.

3.2.3.2 Relevant implication

In the terminology of relevant logics, both the spatial conditional $\triangleright$ and the epistemic conditional $\supseteq$ are *relevant* (as opposed to the *truth-functional* material conditional $\rightarrow$) in the sense that information based on which the antecedent is evaluated is relevant to the information based on which the consequent is evaluated. In $\triangleright$, the relevant (and *potential*) information is represented by the adjoint state $(\mathfrak{h}', Q)$. In $\supseteq$, the relevant (and *actual*) information is represented by the event subset $\mathcal{E}_{\phi'}$.

As an example, consider (session identifier and process term omitted) the assertion

$$\epsilon \cdot \mathtt{I}(\mathtt{Eve}, \{\!|M|\!\}_k) \models \mathtt{Eve}\ \mathsf{k}\ k \triangleright \mathtt{Eve}\ \mathsf{k}\ M$$

which states *what* primary knowledge, namely k, Eve *requires to derive* the (secondary) knowledge M in the given model. In other words, if Eve *knew* k then Eve *would know* M in the given model. (Notice the *conditional* mode!) This is a property of Eve's cryptographic *knowledge* w.r.t. its *potentiality*. That is, the addition of information potentially leads to multiplication of knowledge. In comparison, consider the assertion

$$\epsilon \cdot \mathtt{I}(\mathtt{Eve}, \{\!|M|\!\}_k) \cdot \mathtt{I}(\mathtt{Eve}, k) \models \mathtt{Eve}\ \mathsf{k}\ M \supseteq \mathtt{Eve}\ \mathsf{k}\ k$$

which states *how* Eve *actually derives* the secondary knowledge M from the primary knowledge in the given model (cf. Table 3.6, where it becomes evident that if Eve knows the plaintext M then *possibly because* Eve knows the key k, because: first, Eve can derive M from the set $\{\mathtt{I}(\mathtt{Eve}, k), \mathtt{I}(\mathtt{Eve}, \{\!|M|\!\}_k)\}$ of events and k from the set $\{\mathtt{I}(\mathtt{Eve}, k)\}$ of events in her view; and second, $\{\mathtt{I}(\mathtt{Eve}, k)\} \subseteq \{\mathtt{I}(\mathtt{Eve}, k), \mathtt{I}(\mathtt{Eve}, \{\!|M|\!\}_k)\}$). In other words, if Eve *knows* M then possibly (not only probably) because Eve *knows* k in the given model. (Notice

[18] however, FOL-*satisfiability* is *un*decidable

[19] That a certain form of spatial *conjunction* (conjunctive separation) also yields second-order expressiveness has been argued in [KR04].

Table 3.6: Deriving a plaintext

$\epsilon \cdot \mathtt{I}(\mathtt{Eve}, \{\!\lvert M \rvert\!\}_k) \vdash^{\{\mathtt{I}(\mathtt{Eve},\{\lvert M \rvert\!\}_k)\}}_{\mathtt{Eve}} (\mathtt{Eve}, \{\!\lvert M \rvert\!\}_k)$	
$\epsilon \cdot \mathtt{I}(\mathtt{Eve}, \{\!\lvert M \rvert\!\}_k) \vdash^{\{\mathtt{I}(\mathtt{Eve},\{\lvert M \rvert\!\}_k)\}}_{\mathtt{Eve}} \{\!\lvert M \rvert\!\}_k$	$\epsilon \cdot \mathtt{I}(\mathtt{Eve}, \{\!\lvert M \rvert\!\}_k) \cdot \mathtt{I}(\mathtt{Eve}, k) \vdash^{\{\mathtt{I}(\mathtt{Eve},k)\}}_{\mathtt{Eve}} (\mathtt{Eve}, k)$
$\epsilon \cdot \mathtt{I}(\mathtt{Eve}, \{\!\lvert M \rvert\!\}_k) \cdot \mathtt{I}(\mathtt{Eve}, k) \vdash^{\{\mathtt{I}(\mathtt{Eve},k)\}}_{\mathtt{Eve}} k$	$\epsilon \cdot \mathtt{I}(\mathtt{Eve}, \{\!\lvert M \rvert\!\}_k) \cdot \mathtt{I}(\mathtt{Eve}, k) \vdash^{\{\mathtt{I}(\mathtt{Eve},\{\lvert M \rvert\!\}_k),\mathtt{I}(\mathtt{Eve},k)\}}_{\mathtt{Eve}} \{\!\lvert M \rvert\!\}_k$
$\epsilon \cdot \mathtt{I}(\mathtt{Eve}, \{\!\lvert M \rvert\!\}_k) \cdot \mathtt{I}(\mathtt{Eve}, k) \vdash^{\{\mathtt{I}(\mathtt{Eve},\{\lvert M \rvert\!\}_k),\mathtt{I}(\mathtt{Eve},k)\}}_{\mathtt{Eve}} M$	

the *indicative* mode!) This is a property of Eve's cryptographic *knowledge* w.r.t. its *actuality*. In contrast, consider the tautology (i.e., universal assertion)

$$\models (\mathtt{Eve\ k}\ \{\!|M|\!\}_k \wedge \mathtt{Eve\ k}\ k) \rightarrow \mathtt{Eve\ k}\ M$$

which states a property of a cryptographic *operation*, namely encryption. We believe that $\rhd$ and $\supseteq$ are (perhaps *the*) two natural — and incidentally, relevant — notions of implication for cryptographic knowledge.

3.2.3.3 Conflicting obligations

A particularly interesting use of the spatial and the epistemic conditional is the definition of a cryptographically meaningful notion of permission (cf. Table 3.4) and prohibition (cf. Appendix B). Our definition says that it is permitted that ϕ is true if and only if if ϕ *were* true then whenever a state of violation *would be* reached, it *would not be* due to ϕ being true. This (reductionistic) notion of permission is inspired by [MDW94, Page 9] where a notion of prohibition is defined in the framework of dynamic logic. The authors resume their basic idea as "...some action is forbidden if doing the action leads to a state of violation." Observe that [MDW94] construe a notion of *prohibition* based on *actions*, whereas we construe a notion of *permission* based on *propositions*. We recall that the motivation of reductionistic approaches to (standard) deontic logic (SDL) is the existence of weak paradoxes in SDL. That is, SDL actually contains true statements that are counter to the normative intuition it was originally intended to capture.

In SDL permission, prohibition, and obligation are interdefinable, whereas in CPL only permission and prohibition are. In fact, there is no notion of obligation in CPL because (faulty) cryptographic protocols create a context with *conflicting obligations* whose treatment would require machinery from *defeasible* deontic logic [Nut97]. Consider that it must be obligatory that (1) a state of violation be never reached during protocol execution, and (2) agents always comply with protocol prescription. These two obligations are obviously conflicting in a context created by the execution of a faulty protocol, which by definition does reach a state of violation.

3.2.3.4 Logical omniscience

Our semantics for the epistemic modality reconciles the cryptographically intuitive but incomplete semantics from [AT91] with the complete (but less computational), renaming semantics from [CD05a]. We achieve this by casting the cryptographic intuition from [AT91] in a simple (rule-based) and visibly computational formulation of epistemic accessibility. Similarly to [AT91], we parse unintelligible data in an agent's *a individual* knowledge M into abstract messages ■. In addition, and inspired by [CD05b, CD05a], we parse unintelligible data in an agent's *a propositional* knowledge $\mathsf{K}_a(\phi)$. Thanks to this additional parsing, our epistemic modality avoids weak paradoxes in the context of Dolev-Yao cryptography that, like in SDL, exist in standard epistemic logic (SEL). In the context of Dolev-Yao cryptography, these paradoxes are due to epistemic necessitation

$$\frac{\models \phi}{\models \mathsf{K}_a(\phi)}$$

i.e., the fact that an agent a knows all logical truths (logical omniscience) such as $\exists v(\{\!|M|\!\}_k = \{\!|v|\!\}_k)$. To illustrate, consider the following simple example. Let $P \in \mathcal{P}$ and $M \in \mathcal{M}$. Then paradoxically $(\epsilon, P) \models \mathsf{K}_a(\exists v(\{\!|M|\!\}_k = \{\!|v|\!\}_k))$ "in" SEL but truthfully $(\epsilon, P) \not\models \mathsf{K}_a(\exists v(\{\!|M|\!\}_k = \{\!|v|\!\}_k))$ in CPL because $\models \neg\exists v(\blacksquare = \{\!|v|\!\}_k)$ (cf. "otherwise"-clause in the truth denotation of $\mathsf{K}_a(\phi)$ in Table 3.4). In a cryptographic setting, epistemic necessitation should — and in CPL does — take the following form [CD05a]:

$$\frac{\models \phi}{\models a \,\mathsf{k}\, M \rightarrow \mathsf{K}_a(\phi)} \; M \text{ is a tuple of the key values in } \phi$$

In the context of Dolev-Yao cryptography, the presence of logical omniscience in SEL seems, interestingly, to be due to the absence of relevance in the truth condition of the epistemic modality. The condition is in fact a truth-functional (meta-level) implication, which is true whenever its consequent is true, which in turn is always the case for a tautological consequent. Therefore, any solution to the problem of logical omniscience *must* break the truth-functionality of the meta-level implication and make it relevant. This is precisely what we do: the relevant information is represented by the history $\mathfrak{h}$ of protocol state $\mathfrak{p}@i$ from the antecedent, used for cryptographic parsing in the consequent. Note that our truth condition for the epistemic modality is a simplification of the one of [CD05b, CD05a] in the sense that we eliminate one universal quantifier (the one over renamings) thanks to the employment of cryptographic parsing. Further note that our epistemic modality does capture knowledge, i.e., $\models \mathsf{K}_a(\phi) \rightarrow \phi$, due to the reflexivity of its associated accessibility relation. Treating logical omniscience has a price:

Proposition 2 *Logical equivalence ($\Leftrightarrow$) is* not *a congruence.*

Proof. see Appendix A

3.2.3.5 Other connections

What is more, our (basic) location predicate $a@x$ enables us to invent, by macro-definition, *spatial freeze quantifiers* (in analogy to the well-known temporal freeze quantifiers, which we are also able to macro-define, analogously, in the real-time setting, cf. Section 3.4.3): $\Box_{a.x}(\phi) := \Box(a@x \rightarrow \phi)$ and $\Diamond_{a.x}(\phi) := \neg\Box_{a.x}\neg(\phi)$, and further $\Box_a(\phi) := \forall x(\Box_{a.x}(\phi))$ which corresponds to the location modality $@_a[\phi]$ of distributed temporal logic [CVB05]. Spatial freeze quantifiers are, for example, useful for the macro-definition of action predicates restricted to particular sessions, e.g., $a.x \xrightarrow[\texttt{Eve}]{M} b := \Diamond_{a.x}(a \xrightarrow[\texttt{Eve}]{M} b)$.

Finally, the popularity of strand spaces [FHG99] as an execution model for cryptographic protocols justifies that we briefly compare our classical, trace-based execution model to strand spaces. According to [FHG99, Definition 2.2], a strand space over a set of message terms (in our case $\mathcal{M}$) is a set (say $\mathcal{S}$) (of strand names) with a so-called trace mapping $\mathrm{tr} : \mathcal{S} \rightarrow (\pm\mathcal{M})^*$, where $\pm\mathcal{M} := \{ +M \mid M \in \mathcal{M} \} \cup \{ -M \mid M \in \mathcal{M} \}$ designates the set of so-called signed message terms. In our terminology, the intended meaning of a strand (name) is the one of a located session name ($a.x$), and the one of a positive (resp. negative) message term is insecure output (resp. input). With

these intended meanings and $\mathcal{S} := \{\, a.x \mid a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}} \text{ and } x \in \boldsymbol{\mathcal{X}} \,\}$, strands (and its concept) are obviously strictly included in our (concept of) traces of insecure and secure message input/output events. The inclusion is strict because [FHG99, Definition 2.2] does not allow for secure message input/ouput.

3.3 Application: formalisation case studies

We exemplify the expressiveness of CPL on a selection of *tentative* formalisations of fundamental cryptographic states of affairs. To the best of our knowledge, (1) no other existing crypto logic is sufficiently expressive to allow for the *definition* of the totality of these properties, and (2) the totality of these properties has never been expressed before in any other formalism. In fact, entire logics (e.g., [BAN90], [SS99], [HO02]) have been designed to capture a single cryptographic state of affairs (e.g., authenticity, anonymity, resp. secrecy). We invite the reader to validate our formalisations on the criteria of intuitiveness and succinctness, but also to discern that the simplicity of the formalisation *results* is in sharp contradistinction to the difficulty of their formalisation *process*. However, thanks to the empowerment that CPL confers, a formalisation process involving such a large number of conceptual degrees of freedom has become *tractable* at an engineering level. Observe that our formalisations of cryptographic states of affairs, except for the one of key separation and those of trust-related affairs, involve *no actions*, just *pure knowledge*. Note that the formalisations employ macro-defined predicates (cf. Appendix B; the reader is urged to consult it) and that $\alpha(b)$ abbreviates disjunction of name generation, sending, and receiving performed by b.

3.3.1 Trust-related affairs

Maliciousness Agent b is *malicious*, written $\mathsf{malicious}(b)$, :iff b knowingly performs a forbidden action at some time, written $\Diamond\!\!\!\!+(\alpha(b) \wedge \mathsf{F}\alpha(b) \wedge \mathsf{K}_b(\mathsf{F}\alpha(b)))$.

Honesty Agent b is *honest*, written $\mathsf{honest}(b)$, :iff b is not malicious, written $\neg\mathsf{malicious}(b)$.

Faultiness b is *faulty*, written $\mathsf{faulty}(b)$, :iff b performs a forbidden action at some time, written $\Diamond\!\!\!\!+(\alpha(b) \wedge \mathsf{F}\alpha(b))$.

Prudency b is *prudent*, written $\mathsf{prudent}(b)$, :iff b is not faulty, written $\neg\mathsf{faulty}(b)$.

Trustworthiness Agent a *trusts* b, written a trusts b, :iff a knows that b is prudent, written $\mathsf{K}_a(\mathsf{prudent}(b))$.[20]

3.3.2 Confidentiality-related affairs

Shared Secret Datum M is a *shared secret* among agents a and b, written M $\mathsf{sharedSecret}$ (a, b), :iff only a and b know M, written $a \;\mathsf{k}\; M \wedge b \;\mathsf{k}\; M \wedge \forall(c : \mathtt{A}_{\mathtt{Adv}})(c \;\mathsf{k}\; M \rightarrow (c = a \vee c = b))$.

[20] this is about *justified* trust (*a rightly* trusts *b*) as opposed to *blind* trust (*a possibly wrongly* trusts *b*)

Secrecy A protocol has the (reachability-based) secrecy property :iff the adversary Eve never knows any classified information, written $\boxplus\forall m(\mathsf{F}(\texttt{Eve}\ \mathsf{k}\ m) \rightarrow \neg\,\texttt{Eve}\ \mathsf{k}\ m)$.

Our formalisation is an instance of the pattern $\mathsf{F}\phi \rightarrow \neg\phi$, relating illegitimate to actual states of affairs, and expressing that if something must not be then it actually is not. The pattern is equivalent to $\phi \rightarrow \mathsf{P}\phi$.

Anonymity Agent b is anonymous to agent a in state of affairs $\phi(b)$:iff if a knows that some agent is involved in ϕ then a cannot identify that agent with b, written $\mathsf{K}_a(\exists(c : \texttt{A})(\phi(c))) \rightarrow \neg\mathsf{K}_a(\phi(b))$, which is logically equivalent to $\neg\mathsf{K}_a(\phi(b))$.

Data Derivation Agent b knows M' due to agent a knowing M (when $a \neq b$ then necessarily due to communication from a to b), written $M' \sqsupseteq_{(a,b)} M := b\ \mathsf{k}\ M' \wedge (b\ \mathsf{k}\ M' \sqsupseteq a\ \mathsf{k}\ M)$[21] (when $a = b$ we just write $M' \sqsupseteq_a M$).

Non-Interaction There is absence of interaction between agents a and b, written $a \mid b := \neg\exists m\exists m'(m \sqsupseteq_{(a,b)} m' \vee m \sqsupseteq_{(b,a)} m')$.

Perfect Forward Secrecy "[...] compromise of long-term keys [k] does not compromise past session keys [k']." [MvOV96, Page 496], written $\neg\diamondplus\exists(k : \texttt{K}^{\infty})\exists(k' : \texttt{K}^{1})(k' \sqsupseteq_{\texttt{Eve}} k)$

Known-Key Attack "[...] an adversary obtains some keys used previously and then uses this information to determine new keys." [MvOV96, Page 41], written $\exists(k : \texttt{CK})\exists(k' : \texttt{CK})(k' \neq k \wedge k' \sqsupseteq_{\texttt{Eve}} k)$

Agent Corruption The adversary, somehow, comes to know all what an agent (say a) knows in state of affairs ϕ, written $\forall m(a\ \mathsf{k}\ m \rightarrow (\texttt{Eve}\ \mathsf{k}\ m \blacktriangleright \phi))$.

3.3.3 Authentication-related affairs

Key Confirmation "[...] one party [a] is assured that a second (possibly unidentified) party [b] actually has possession of a particular secret[22] key [k]." [MvOV96, Page 492], written $k : \texttt{K} \wedge \mathsf{K}_a(b\ \mathsf{k}\ k)$

Key Authentication

- *implicit*: "[...] one party [a] is assured that no other party [c] aside from a specifically identified second party [b] (and possibly additional identified trusted parties) may gain access to a particular secret key [k]." [MvOV96, Page 492], written $k : \texttt{K} \wedge \mathsf{K}_a(\forall(c : \texttt{A}_{\texttt{Adv}})(c\ \mathsf{k}\ k \rightarrow (c = a \vee c = b)))$
- *explicit*: "[...] both (implicit) key authentication and key confirmation hold." [MvOV96, Page 492], written simply as conjunction of implicit key authentication and key confirmation

Message Integrity Agent b knows that M is an *intact* message from agent a, written $\mathsf{K}_b(M \sqsupseteq_{(a,b)} M)$.

[21] A material conditional would not do here because the antecedent and the consequent are *epistemically* — and thus not truth-functionally — *related* via data derivation.

[22] in our terminology, 'secret' here means 'symmetric'

Message Authorship Agent a *authored* datum M, written a $\mathsf{authored}$ M, :iff once a was the only one to know M, written $\diamondsuit(a\ \mathsf{k}\ M \wedge \forall(b : \mathtt{A_{Adv}})(b\ \mathsf{k}\ M \rightarrow b = a))$.

Message Authentication (or *Authenticity*) Datum M is *authentic w.r.t. its origin* (say agent a) from the viewpoint of agent b :iff b can authentically attribute (i.e., in the sense of authorship) M to a, i.e., b knows that a authored M, written $\mathsf{K}_b(a\ \mathsf{authored}\ M)$.

Key Transport (safety) between agents a and b initiated by a

- unacknowledged $\mathsf{uaKT}(a, b)$:

$$\boxplus\forall(k : \mathtt{K})(\mathsf{K}_b(a\ \mathsf{authored}\ k) \rightarrow \mathsf{K}_b(k\ \mathsf{sharedSecret}\ (a, b)))$$

- acknowledged $\mathsf{aKT}(a, b)$:

$$\boxplus\forall(k : \mathtt{K})(\mathsf{K}_a(\mathsf{K}_b(a\ \mathsf{authored}\ k)) \rightarrow \mathsf{K}_a(\mathsf{K}_b(k\ \mathsf{sharedSecret}\ (a, b))))$$

Key Agreement (safety) between agents a and b initiated by a

- unacknowledged $\mathsf{uaKA}(a, b)$:

$$\boxplus\forall m_a \forall m_b((\mathsf{K}_b(a\ \mathsf{authored}\ m_a) \wedge \mathsf{K}_a(b\ \mathsf{authored}\ m_b)) \rightarrow \mathsf{K}_a((m_a, m_b)\ \mathsf{sharedSecret}\ (a, b)))$$

- acknowledged $\mathsf{aKA}(a, b)$:

$$\boxplus\forall m_a \forall m_b((\mathsf{K}_b(a\ \mathsf{authored}\ m_a) \wedge \mathsf{K}_b(\mathsf{K}_a(b\ \mathsf{authored}\ m_b))) \rightarrow \mathsf{K}_b(\mathsf{K}_a((m_a, m_b)\ \mathsf{sharedSecret}\ (a, b))))$$

Entity Authentication (or *Identification*) (safety) via a shared secret between agents a and b initiated by a

- *unilateral* (or *weak*) entity authentication $\mathsf{uEA}(a, b)$: "[...] the process whereby one party [b] is assured (through acquisition of corroborative evidence [m]) of the identity of a second party [a] involved in a protocol, and that the second has actually participated (i.e., is active at, or immediately prior to, the time the evidence is acquired)." [MvOV96, Page 386], written

$$\boxplus\forall m(\mathsf{K}_b(a\ \mathsf{authored}\ m) \rightarrow \mathsf{K}_b(m\ \mathsf{sharedSecret}\ (a, b)))$$

  Notice that unilateral entity authentication is unacknowledged transport of an *arbitrary* secret, e.g., not necessarily a symmetric key.

- *weakly mutual* (or *strong-weak*) entity authentication $\mathsf{wmEA}(a, b)$: "[...] [one party (say a)] has fresh assurance that [the other party (say b)] has knowledge of [a] as her peer entity." [BM03, Page 39], written

$$\boxplus\forall m_a \forall m_b((\mathsf{K}_b(a\ \mathsf{authored}\ m_a) \wedge \mathsf{K}_b(\mathsf{K}_a(b\ \mathsf{authored}\ m_b))) \rightarrow \mathsf{K}_b(\mathsf{K}_a((m_a, m_b)\ \mathsf{sharedSecret}\ (a, b))))$$

  Notice that weakly mutual entity authentication coincides with acknowledged key agreement.

- *strongly mutual* (or *strong-strong*) entity authentication $\mathsf{smEA}(a, b)$:

$$\boxplus\forall m_a \forall m_b((\mathsf{K}_a(\mathsf{K}_b(a \text{ authored } m_a)) \wedge \mathsf{K}_b(\mathsf{K}_a(b \text{ authored } m_b))) \rightarrow \mathsf{K}_a(\mathsf{K}_b(\mathsf{K}_a((m_a, m_b) \text{ sharedSecret } (a, b)))))$$

Notice that our formalisations of key transport/agreement and entity authentication only address *safety*, but not *liveness*, i.e., that some key actually gets transported/agreed upon and that some entity is authenticated. The reason is that due to the adversary, liveness *cannot* be guaranteed.

Visibly, both key transport/agreement and entity authentication rely on message authentication as well as a shared secret, and authentication-related affairs rely on confidentiality-related affairs.

Slogan 16 *Authenticity is epistemic accessibility between agents and their data. Secrecy is epistemic inaccessibility to agents and their data. The two states of affairs are linked.*

3.3.4 Commitment-related affairs

Proof Datum M is a *cryptographic*[23] proof for the truth of proposition ϕ, written M proofFor ϕ, :iff assuming an arbitrary agent a knows M guarantees that a knows that ϕ is true, written $\forall(a : \mathtt{A_{Adv}})(a \mathsf{\ k\ } M \rhd \mathsf{K}_a(\phi))$.

Evidence Datum M is cryptographic evidence for the truth of proposition ϕ, written M evidenceFor ϕ, :iff assuming an arbitrary agent a knows that ϕ is true guarantees that a knows M, written $\forall(a : \mathtt{A_{Adv}})(\mathsf{K}_a(\phi) \rhd a \mathsf{\ k\ } M)$.

Provability Agent a can prove that proposition ϕ is true, written $\mathsf{P}_a(\phi)$, :iff a knows a (cryptographic) proof for ϕ, written $\exists m(m \text{ proofFor } \phi \wedge a \mathsf{\ k\ } m)$.

Non-Repudiation Agent b cannot repudiate authorship of M to agent a :iff a can prove that b authored M, written $\mathsf{P}_a(b \text{ authored } M)$.

Notice that non-repudiation is authenticity strengthened (from knowledge) to provability.

Contract Signing "[...] two players [say a and b] wish to sign a contract m in such a way that either each player obtains the other's signature [S], or neither player does." (fair exchange of electronic signatures $\mathsf{FEES}(a, b)$), written $\bigoplus((a \mathsf{\ k\ } S_b \wedge b \mathsf{\ k\ } S_a) \vee (\neg\, a \mathsf{\ k\ } S_b \wedge \neg\, b \mathsf{\ k\ } S_a))$

- *Optimism*: "[...] no honest party [neither a nor b] interacts with the trusted third party [say c]." [ASW00], written $a \mid c \ \wedge\ b \mid c$
- *Fairness*: "[...] it is infeasible for the adversary [Eve] to get the honest player's [a] signature [S_a], without the honest player getting the adversary's signature [$S_{\mathtt{Eve}}$]." [ASW00], written $\boxplus(\mathtt{Eve} \mathsf{\ k\ } S_a \rightarrow \bigoplus(a \mathsf{\ k\ } S_{\mathtt{Eve}}))$
- *Completion*: "[...] it is infeasible for the adversary [...] to prevent [a] and [b] from successfully exchanging their signatures." [ASW00], writtten $\bigoplus(a \mathsf{\ k\ } S_b \wedge b \mathsf{\ k\ } S_a)$

[23] as opposed to *propositional* proof (i.e., a sequence of propositions that is compliant with a relation of deduction); cryptographic proofs can be viewed as *cryptographic encodings* (i.e., cryptographic *Gödel-numberings*) of propositional proofs

- *Accountability*: "[...] if the trusted third party misbehaves [i.e., the contract signing property FEES is violated] then this can be *proven*." [ASW00], written $\boxplus(\neg\mathsf{FEES}(a,b) \rightarrow \diamondplus(\mathsf{P}_a(\neg\mathsf{FEES}(a,b)) \wedge \mathsf{P}_b(\neg\mathsf{FEES}(a,b))))$
- *Abuse-freeness*: "[...] [b] does not obtain publicly verifiable information about (honest) [a] signing the contract until [b] is also bound by the contract."[24] [GJM99], written $\neg\mathsf{P}_b(a \text{ authored } S_a)\mathsf{U}b \text{ authored } S_b$

It has been argued that contract signing requires branching time [CKW07]. However, our tentative formalisation of contract signing suggests that branching-time logic is not necessary for this purpose. It has even been argued that linear-time is preferable (implying "sufficient") over branching-time logic in general [Var01]. We shall not settle this argument here, but confine ourselves to alimenting it. In any case, it would be easy to replace CPL's linear-time skeleton with a branching-time skeleton such as CTL*.

Visibly, cryptographic proof and evidence are *dual concepts*, and commitment-related affairs rely on authentication-related affairs.

Then, we have actually been able to macro-define a Gödel-style *provability modality*, and, with it, are able to macro-define the *intuitionistic conditional* in CPL!

Theorem 2 *The operator* P_a *is compliant with the modal system* ***S4*** *adapted to Dolev-Yao cryptography (i.e., with the necessitation rule* $\boldsymbol{N}$ *replaced by* $\boldsymbol{N^{DY}}$*)*[25].

Proof. P_a complies with (cf. Appendix A for an elementary, Fitch-style proof)

K $\quad \models \mathsf{P}_a(\phi \rightarrow \phi') \rightarrow (\mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\phi'))$

T $\quad \models \mathsf{P}_a(\phi) \rightarrow \phi$

4 $\quad \models \mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\mathsf{P}_a(\phi))$

$\mathbf{N^{DY}}$ $\quad \dfrac{\models \phi}{\models a \,\mathsf{k}\, M \rightarrow \mathsf{P}_a(\phi)}$ M is a tuple of the key values in ϕ

Hence, (the classical logic) CPL can capture the provability meaning of intuitionistic implication via the following macro-definition:

$$\phi \mapsto \phi' := \exists(a : \mathtt{A_{dv}})(\mathsf{P}_a(\phi \rightarrow \phi'))$$

The intuitionistic conditional is another example of relevant implication: information (a proof of ϕ) based on which the antecedent is evaluated is relevant to the information (a proof of ϕ') based on which the consequent is evaluated in the sense that any proof of ϕ is also a proof of ϕ' (cf. **K**).

The obvious temptation is to attempt a *Curry-Howard isomorphism* [dG95] between cryptographic protocols and propositions. That is, to look

[24] symmetrically for "(honest) [b]"

[25] As P_a is defined in terms of K_a, P_a is compliant with **S4** *simpliciter* when K_a is compliant with **S5** *simpliciter*. Our K_a can be (re)made compliant with **S5** *simpliciter* by simply removing the treatment of logical omniscience (i.e., the cryptographic parsing) in its definition.

1. at a *proposition* $\phi \in \mathcal{F}$ for which there are $(\mathfrak{h}, P), (\mathfrak{h}', P') \in \mathcal{H} \times \mathcal{P}$, $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, and $M \in \mathcal{M}$ s.t. $(\mathfrak{h}, P) \longrightarrow (\mathfrak{h}', P')$ and $(\mathfrak{h}', P') \models M$ $\mathsf{proofFor}$ $\phi \wedge a \,\mathsf{k}\, M$ as a *type* for process term P, and

2. at process *term* P as an interactive *proof procedure* (to the benefit of agent a) for the cryptographic *proof* M of ϕ.

Our (macro-defined) concepts of cryptographic proof and provability are related to [AN05], where a notion of *justification* for propositional knowledge is introduced as a primitive concept in the (propositional) epistemic logic **S4** resulting in a *hybrid modality* for both knowledge and provability. That notion of justification roughly corresponds in our (first-order, epistemic-**S5**) setting to the notion of cryptographic proof. However, [AN05] is currently not quite suitable for cryptography due to standard epistemic necessitation and an unsuitable form of positive introspection, namely the one that the existence of a proof of a proposition implies the (hybrid) knowledge-provability of that proposition. Hence, given that Gödel's 1933 paper on a modal logic of provability left the[26] open problem of finding "a precise provability semantics for the modal logic **S4**" [BvBW07, Page 932], we can justly claim having solved via macro-definition, i.e., via *syntactic translation*, a cryptographic analogue of that problem. Gödel's problem was solved in its original, non-cryptographic format in [Art95, Art01].

3.3.5 Compositionality-related affairs

Key Separation The protocol space can be separated in an establishment (production) and a use (consumption) part w.r.t. the key k, written

$$\boxplus\forall m((\exists(a, b : \mathtt{A})(a \xrightarrow[\mathtt{Eve}]{m} b) \wedge k \mathrel{\mathfrak{E}\!\!-}^* m) \longrightarrow \neg\, k \mathrel{-\!\!\mathfrak{Z}}^* m) \otimes$$
$$\boxplus\forall m((\exists(a, b : \mathtt{A})(a \xrightarrow[\mathtt{Eve}]{m} b) \wedge k \mathrel{-\!\!\mathfrak{Z}}^* m) \longrightarrow \neg\, k \mathrel{\mathfrak{E}\!\!-}^* m)$$

Compositional Correctness Protocol (plug-in) P with prehistory $\mathfrak{h}$ is

1. *solely correct* w.r.t. an internal correctness criterion, i.e., *endo-condition* ϕ :iff $(\mathfrak{h}, P) \models \phi$
2. *compositionally correct*, i.e., either
   (a) *existentially composable* w.r.t. an external correctness criterion, i.e., *exo-condition* ϕ' :iff $(\mathfrak{h}, P) \models \phi' \blacktriangleright \phi$,[27] or
   (b) *conditionally composable*, i.e., composable w.r.t. exo-condition ϕ', :iff $(\mathfrak{h}, P) \models \phi' \rhd \phi$, or
   (c) *universally composable* :iff $(\mathfrak{h}, P) \models \top \rhd \phi$.[28]

   The concept of an exo-condition (endo-condition) is to interactive programs what a pre-condition (post-condition) is to non-interactive programs.[29] Our slogan, especially applying to cryptographic protocols, is:

[26] actually two open problems (cf. [BvBW07, Page 932])

[27] the case where ϕ' is $\top$ is obviously uninteresting

[28] the *name* of this notion of correctness coincides with the one from [Can01], and should roughly correspond to the notion of *robust satisfaction* [GL91]

[29] $(\mathfrak{h}, P) \models \phi' \rhd \phi$ roughly corresponds to a Hoare triple $\phi'\{P\}\phi$. Observe the absence of a computation history in Hoare triples: *non*-interactive programs are characteristically history-*in*dependent; *interactive* programs are characteristically history-*dependent*!

Slogan 17 *Stating the possibly weakest* exo-*condition for an* interactive *program is at least as necessary as stating the possibly weakest* pre-*condition for a* non-*interactive program.*

Attack Scenario Protocol P with prehistory $\mathfrak{h}$ and internal correctness criterion ϕ is vulnerable in a protocol context — *de facto* constituting a potential *attack scenario* — with property ϕ' :iff $(\mathfrak{h}, P) \models \phi' \blacktriangleright \neg\phi$.

Notice that a statement of an attack scenario is a negated statement of conditional composability.

Remark 4 *The concept of a* chosen-protocol attack *[KSW98], understood as the adversarial choice of a* different *(attacking) protocol than P* is an instance of *the concept of an attack scenario, and understood as the adversarial choice of an* arbitrary *attacking protocol* coincides with *the concept of an attack scenario.*

3.3.5.1 A popular attack scenario

We exemplify our concept of attack scenario with the perhaps most popular attack on a cryptographic protocol, namely the man-in-the-middle attack on the Needham-Schroeder public-key protocol (NSPuK) for (weakly mutual) entity authentication (acknowledged key agreement). Our choice is motivated by the fact that we wish to explain the unfamiliar (our approach) with the familiar (a paradigmatic attack). Notwithstanding the popularity of the attack, we believe that its contextual formalisation in CPL explicates it to a novel extent of explicitness. The attack is also particularly interesting because the protocol requirement that it violates is particularly challenging to formalise — satisfactorily. We contend that common formulations of entity authentication are unsatisfactory. They usually purport to formalise an intuition expressed as "I know who I'm talking to.". However the actual formulations then only involve belief to varying degrees of explicitness [Low97]. Our slogan, and fact, is:

Slogan 18 *Debatable requirements entail debatable attacks.*

Table 3.7 displays the protocol narration (i.e., an intended run) of core NSPuK, i.e., NSPuK where the public keys of the initiator (e.g., Alice) and the responder (i.e., Bob) are assumed to have already been established. The

Table 3.7: Protocol narration for core NSPuK

1.	$\mathtt{Alice} \rightarrow \mathtt{Bob}$	:	$\{\!\lvert(x_{\mathtt{Alice}}, \mathtt{Alice})\rvert\!\}^{+}_{p^{+}_{\mathtt{Bob}}}$
2.	$\mathtt{Bob} \rightarrow \mathtt{Alice}$	:	$\{\!\lvert(x_{\mathtt{Alice}}, x_{\mathtt{Bob}})\rvert\!\}^{+}_{p^{+}_{\mathtt{Alice}}}$
3.	$\mathtt{Alice} \rightarrow \mathtt{Bob}$	:	$\{\!\lvert x_{\mathtt{Bob}}\rvert\!\}^{+}_{p^{+}_{\mathtt{Bob}}}$

narration describes (elliptically) that first, Alice sends to Bob the encryption under Bob's public key $p^{+}_{\mathtt{Bob}}$ of a tuple of a freshly-generated nonce $x_{\mathtt{Alice}}$ and her name `Alice`; (upon reception, Bob decrypts the message with his private key, stores the first component of the tuple, gets the public key $p^{+}_{\mathtt{Alice}}$ corresponding

to the second component from his key store, generates a fresh nonce $x_{\texttt{Bob}}$, and encrypts the tuple of Alice's and his nonce with Alice's public key;) second, Bob sends his reply to Alice; (upon reception, Alice decrypts the message with her private key, checks that the first component of the tuple is her nonce previously sent to Bob, and encrypts the second component $x_{\texttt{Bob}}$ with Bob's public key $p^{+}_{\texttt{Bob}}$;) third, Alice sends her reply to Bob. Protocol narrations are elliptical in the sense that non-interactive protocol actions are visibly not explicit.

The intention of each protocol step is as follows: the intention of the first step is to challenge the responder (e.g., Bob) to authenticate with the initiator (e.g., Alice); the intention of the second step is twofold, i.e., to accomplish authentication of the responder with the initiator, and to challenge the initiator to authenticate with the responder; the intention of the third step is twofold, i.e., to acknowledge authentication of the responder with the initiator to the responder, and to accomplish authentication of the initiator with the responder. The protocol intends to achieve weakly (due to the *uni*lateral acknowledgement) mutual entity authentication (acknowledged key agreement) between an initiator and a responder.

The protocol narration of NSPuK can be transcribed into a (non-elliptic) formal language, *for example* into the one of Section 4.1 by instantiating the protocol template displayed in Table 3.8 via substitution of `Alice` for *init* and `Bob` for *resp*. Features of that language are: a primitive for key lookup, an input primitive with pattern-matching and guard, and primitives for out-of-band communication. The left (right) column of the table defines the ini-

Table 3.8: Protocol template for core NSPuK

$\text{NSPuK}_{\texttt{INIT}}(slf, oth) :=$	$\text{NSPuK}_{\texttt{RESP}}(slf) :=$
$\texttt{New}\ (x_{slf} : \texttt{X}).$ $\texttt{Get}_{oth}\ (k_{oth} : \texttt{K}^{+}, oth)\ \texttt{in}$ $\texttt{Out}_{oth}\ \{\!\|(x_{slf}, slf)\|\!\}^{+}_{k_{oth}}.$ $\texttt{Get}_{slf}\ (k_{slf} : \texttt{K}^{-}, slf)\ \texttt{in}$ $\texttt{In}\ \{\!\|(x_{slf}, x_{oth})\|\!\}^{+}_{\frac{}{k_{slf}}}\ \texttt{when}\ x_{oth} : \texttt{X}.$ $\texttt{Out}_{oth}\ \{\!\|x_{oth}\|\!\}^{+}_{k_{oth}}.1$	$\texttt{Get}_{slf}\ (k_{slf} : \texttt{K}^{-}, slf)\ \texttt{in}$ $\texttt{In}\ \{\!\|(x_{oth}, oth)\|\!\}^{+}_{\frac{}{k_{slf}}}\ \texttt{when}\ x_{oth} : \texttt{X} \wedge oth : \texttt{A}.$ $\texttt{New}\ (x_{slf} : \texttt{X}).$ $\texttt{Get}_{oth}\ (k_{oth} : \texttt{K}^{+}, oth)\ \texttt{in}$ $\texttt{Out}_{oth}\ \{\!\|(x_{oth}, x_{slf})\|\!\}^{+}_{k_{oth}}.$ $\texttt{In}\ \{\!\|x_{slf}\|\!\}^{+}_{\frac{}{k_{slf}}}.1$
$\text{NSPuK}(init, resp, x_{init}, x_{resp}) := init.x_{init}[\,\text{NSPuK}_{\texttt{INIT}}(init, resp)\,] \;\|\|\; resp.x_{resp}[\,\text{NSPuK}_{\texttt{RESP}}(resp)\,]$	

tiator (responder) role. The bottom row defines the protocol template, distributing (via parallel composition) the roles at the corresponding locations $init.x_{init}[\,\cdot\,]$ and $resp.x_{resp}[\,\cdot\,]$, respectively. The protocol template assumes that each agent has generated her own private and public key, and that each agent's public key has been established with the other agent. The actions of the initiator role are the following: $\texttt{New}\ (x_{slf} : \texttt{X})$ generation — and binding in variable x_{slf} — of a new nonce; $\texttt{Get}_{oth}\ (k_{oth} : \texttt{K}^{+}, oth)\ \texttt{in}$ look up — and binding in variable k_{oth} — of the other agent's (cf. subscript *oth*) public key generated by the other agent herself (cf. parameter *oth*); $\texttt{Out}_{oth}\ \{\!\|(x_{slf}, slf)\|\!\}^{+}_{k_{oth}}$ output of the message $\{\!\|(x_{slf}, slf)\|\!\}^{+}_{k_{oth}}$ to the other, hopefully responding, agent;

$\mathtt{Get}_{slf}\ (k_{slf} : \mathtt{K}^-, slf)\ \mathtt{in}$ look up — and binding in variable k_{slf} — of the local agent's (cf. subscript slf) private key generated by that agent herself (cf. parameter slf); $\mathtt{In}\ \{\!|(x_{slf}, x_{oth})|\!\}^+_{k_{slf}}\ \mathtt{when}\ x_{oth} : \mathtt{X}$ guarded (cf. guard $x_{oth} : \mathtt{X}$) input of a message with pattern[30] $\{\!|(x_{slf}, x_{oth})|\!\}^+_{k_{slf}}$ and binding in variable x_{oth} of the other, apparently responding, agent's nonce; $\mathtt{Out}_{oth}\ \{\!|x_{oth}|\!\}^+_{k_{oth}}$ output of the message $\{\!|x_{oth}|\!\}^+_{k_{oth}}$ to the other agent; and, finally, 1 — termination. The actions of the responder role are (almost) symmetrical to the ones of the initiator role.

The previously mentioned assumptions about preliminary generation and (authenticated, of course) establishment of public keys can be modelled by means of corresponding key-generation and out-of-band communication events, chained up to form the protocol prehistory displayed in Table 3.9. We recall that out-of-band (or private) communication is, by definition, authenticated (and secret), and that the adversary (Eve) can, as in the mentioned attack, also be an insider.

Table 3.9: Prehistory for core NSPuK

$$\begin{aligned}\mathfrak{h} := \epsilon \cdot\ & \mathtt{N}(\mathtt{Alice}, x_{a0}, p_{\mathtt{Alice}}, \mathtt{Alice}) \cdot \mathtt{N}(\mathtt{Bob}, x_{b0}, p_{\mathtt{Bob}}, \mathtt{Bob}) \cdot \\ & \mathtt{sO}(\mathtt{Alice}, x_{a0}, p^+_{\mathtt{Alice}}, \mathtt{Bob}) \cdot \mathtt{sI}(\mathtt{Bob}, x_{b0}, p^+_{\mathtt{Alice}}, \mathtt{Alice}) \cdot \\ & \mathtt{sO}(\mathtt{Bob}, x_{b0}, p^+_{\mathtt{Bob}}, \mathtt{Alice}) \cdot \mathtt{sI}(\mathtt{Alice}, x_{a0}, p^+_{\mathtt{Bob}}, \mathtt{Bob}) \cdot \\ & \mathtt{sO}(\mathtt{Alice}, x_{a0}, p^+_{\mathtt{Alice}}, \mathtt{Eve}) \cdot \mathtt{sI}(\mathtt{Eve}, x_{e0}, p^+_{\mathtt{Alice}}, \mathtt{Alice}) \cdot \\ & \mathtt{sO}(\mathtt{Bob}, x_{b0}, p^+_{\mathtt{Bob}}, \mathtt{Eve}) \cdot \mathtt{sI}(\mathtt{Eve}, x_{e0}, p^+_{\mathtt{Bob}}, \mathtt{Bob})\end{aligned}$$

This completes the definition of the initial state

$$(\mathfrak{h}, \mathrm{NSPuK}(\mathtt{Alice}, \mathtt{Bob}, x_{a1}, x_{b1}))$$

of (our attack scenario for) core NSPuK.

Table 3.10 displays the narration of the actual attack. The attack can be

Table 3.10: Attack narration for NSPuK

$$\begin{array}{lrcl}
1. & \mathtt{Alice} & \rightarrow \mathtt{Eve} & : \{\!|(x_{\mathtt{Alice}}, \mathtt{Alice})|\!\}^+_{p^+_{\mathtt{Eve}}} \\
1'. & \mathtt{Eve}_{\mathtt{Alice}} & \rightarrow \mathtt{Bob} & : \{\!|(x_{\mathtt{Alice}}, \mathtt{Alice})|\!\}^+_{p^+_{\mathtt{Bob}}} \\
2'. & \mathtt{Bob} & \rightarrow \mathtt{Eve}_{\mathtt{Alice}} & : \{\!|(x_{\mathtt{Alice}}, x_{\mathtt{Bob}})|\!\}^+_{p^+_{\mathtt{Alice}}} \\
2. & \mathtt{Eve} & \rightarrow \mathtt{Alice} & : \{\!|(x_{\mathtt{Alice}}, x_{\mathtt{Bob}})|\!\}^+_{p^+_{\mathtt{Alice}}} \\
3. & \mathtt{Alice} & \rightarrow \mathtt{Eve} & : \{\!|x_{\mathtt{Bob}}|\!\}^+_{p^+_{\mathtt{Eve}}} \\
3'. & \mathtt{Eve}_{\mathtt{Alice}} & \rightarrow \mathtt{Bob} & : \{\!|x_{\mathtt{Bob}}|\!\}^+_{p^+_{\mathtt{Bob}}}
\end{array}$$

orchestrated by an *active insider adversary* that performs *denial of service* and *impersonation* across two different, interleaved sessions, cf. (un)primed numbering. It consists in:

[30] with pattern-matching effectuating the identity check on the received nonce

1. Eve tricking (wrongly trusting) Alice (believing that Eve is a legitimate agent) to initiate a regular session

$$Q := \mathtt{Alice}.x_{a2}[\,\mathrm{NSPuK}_{\mathtt{INIT}}(\mathtt{Alice},\mathtt{Eve})\,]$$

   with Eve

2. Eve *dis*abling the execution (denial of service) of the regular session initiation

$$\mathtt{Alice}.x_{a1}[\,\mathrm{NSPuK}_{\mathtt{INIT}}(\mathtt{Alice},\mathtt{Bob})\,]$$

3. Eve impersonating Alice in the face of (wrongly trusting) Bob (lead to believe that he is talking only to Alice while in fact talking also to Eve) by *en*abling the execution of the regular session response

$$\mathtt{Bob}.x_{b1}[\,\mathrm{NSPuK}_{\mathtt{RESP}}(\mathtt{Bob})\,]$$

   concurrently with Q.

In result, Alice is lead to believe that she is talking only to Eve (via session x_{a2}) while in fact talking also to Bob (via impersonator Eve and session x_{b1}), and Bob is lead to believe that he is talking only to Alice (via session x_{b1}) while in fact talking also to Eve (via impersonator Eve and session x_{a1}). The protocol obviously fails to achieve its requirement.

The assumption about private- and public-key generation and public-key establishment is, of course, also valid for Eve and regular interactions between Alice and Eve, respectively. That is, the protocol context is assumed to contain the prehistory $\mathfrak{h}' := \epsilon \cdot \mathtt{N}(\mathtt{Eve}, x_{e0}, p_{\mathtt{Eve}}, \mathtt{Eve}) \cdot \mathtt{sO}(\mathtt{Eve}, x_{e0}, p^{+}_{\mathtt{Eve}}, \mathtt{Alice}) \cdot \mathtt{sI}(\mathtt{Alice}, x_{a0}, p^{+}_{\mathtt{Eve}}, \mathtt{Eve}) \cdot \mathtt{sO}(\mathtt{Eve}, x_{e0}, p^{+}_{\mathtt{Eve}}, \mathtt{Bob}) \cdot \mathtt{sI}(\mathtt{Bob}, x_{b0}, p^{+}_{\mathtt{Eve}}, \mathtt{Eve})$. More formally,

- $(\mathfrak{h}', Q) \in \mathcal{H} \times \mathcal{P}$ and
- $(\mathfrak{h}', Q) \models \mathsf{uEA}(\mathtt{Alice}, \mathtt{Eve})$ and
- $(\mathfrak{h} \circ \mathfrak{h}', \mathrm{NSPuK}(\mathtt{Alice}, \mathtt{Bob}, x_{a1}, x_{b1}) \,|||\, Q) \models \neg\mathsf{wmEA}(\mathtt{Alice}, \mathtt{Bob})$

by which we obtain

$$(\mathfrak{h}, \mathrm{NSPuK}(\mathtt{Alice}, \mathtt{Bob}, x_{a1}, x_{b1})) \models \mathsf{uEA}(\mathtt{Alice}, \mathtt{Eve}) \blacktriangleright \neg\mathsf{wmEA}(\mathtt{Alice}, \mathtt{Bob})$$

representing our (property-based or logical) attack scenario for NSPuK. We invite the reader to compare this scenario to the corresponding, model-based (or process-algebraic) attack scenario described in Section 4.1.4.

3.4 tCPL: an extension of CPL with real time

We extend (core) CPL (qualitative time) with real time, i.e., time stamps, timed keys, and potentially drifting local clocks, to tCPL (quantitative time). Our extension is conservative and really simple (a single section is enough to describe it!). It requires only the refinement of two relational symbols (one new defining rule resp. parameter) and of one modality (one new conjunct in its truth condition), and the addition of two relational symbols (but no operators!). Our work

thus provides further evidence for Lamport's claim that adding real time to an untimed formalism is really simple [Lam05]. The special-purpose machinery for timed (including cryptographic) settings need not be built from scratch nor be heavy-weight.

3.4.1 Historical and topical context

The formal specification, modelling, and verification of *general-purpose timed* systems has received considerable attention from the formal methods community since the end of the nineteen-eighties. See [Wan04] for a survey of timed system models (automata, Petri nets), model- and property-based specification languages (process calculi, resp. logics), and verification tools; and [BMN00] for a survey of timed property-based specification languages (logics).

However, the formal methods community has paid comparatively little, and only recent (since the end of the nineteen-nineties), attention to the timed aspects of *cryptographic* systems, e.g., cryptographic protocols, which due to their complexity deserve *special-purpose* models, and formalisms[31] for their specification and verification.

We are aware of the following special-purpose formalisms for timed cryptographic protocols.

- Model-based formalisms (process calculi): [ES00], [GM04], [HJ05] with *discrete* time; [Sch99], [BEL05], and our own contribution (cf. Section 4.2) with *dense* time
- Property-based formalisms (logics): *interval*-based [HS04a]; time-parametrised epistemic modalities [KM99] and a second-order logic [BEL05] both *point*-based, and our hereby presented logic tCPL allowing for *both* temporal points *and* intervals.

Clearly, "[d]ense-time models are better for distributed systems with multiple clocks and timers, which can be tested, set, and reset independently." [Wan04]. Specifically in cryptographic systems, "[c]locks can become unsynchronized due to sabotage on or faults in the clocks or the synchronization mechanism, such as overflows and the dependence on potentially unreliable clocks on remote sites [...]" [Gon92]. Moreover, "[e]rroneous behaviors are generally expected during clock failures [...]" [Gon92].

Timed logics can be classified w.r.t. their *order* and the nature of their *temporal domain*.

Order Propositional logic is simply too weak for specification purposes (but is good for fully-automated, approximative verification); modal logics provide powerful abstractions for specification purposes, but are still not expressive enough (cf. Section 3.1.2); higher-order logics are too expressive at the cost of axiomatic and algorithmic incompleteness (but are good as logical frameworks); finally "[f]irst-order logics seem a good compromise between expressiveness and computability, since they are [axiomatically] complete in general." [Wan04]. We

[31] In our view, a formalism consists of exactly three components: a formal (e.g., programming or logical) language, a mathematical model (or interpretation structure), and a formal semantics (e.g., effect or truth) for the language in terms of the model.

recall that core CPL is a first-order, poly-dimensional modal (norms, knowledge, space, *qualitative* time) logic.

Temporal domain We recall that core CPL can be instantiated with a transitive, irreflexive, linear and bounded in the past, possibly branching (but a priori flattened) and unbounded (depending on the protocol) in the future, discrete (due to event-induced protocol states) temporal accessibility relation (cf. Section 4.1). That is, CPL has a hybrid (state- and event-based) temporal domain: "[...] neither pure state-based nor pure event-based languages quite support the natural expressiveness desirable for the specification of real-world systems [...]" [Wan04]. tCPL can be instantiated with a temporal accessibility relation that *additionally* accounts for *quantitative* time (cf. Section 4.2). That is, time is (1) rational-number valued, yielding a dense temporal grain; (2) referenced explicitly (the truth of a timed formula does not depend on its evaluation time), but implicit-time operators are macro-definable (cf. Section 3.4.3); (3) measured with potentially drifting local clocks (one per agent), where the (standard Dolev-Yao) adversary's local clock has drift rate 1; (4) advanced monotonically by letting the adversary choose the amount by which she desires to increase her local clock (*de facto* the system clock); and (5) determinant for adversarial break of short-term keys, enabled jointly by key expiration and ciphertext-only attacks (the weakest reasonable attack).

Rational versus real numbers Cryptographic messages have finite length, which implies that real numbers, e.g., real-valued time stamps, are not transmittable as such, and real clocks only have finite precision.

Timed adversary model Our model amounts to a natural generalisation of the adversary's scheduling power from the control of the (relative) temporal *order* of protocol events in the network (space) to the control of their (absolute) temporal *issuing* (time).

The following section describes the extension of CPL to tCPL. The extension depends on the core described in the previous sections (the reader is urged to consult them) and parallels the extension from C^3 (cf. Section 4.1) to tC^3 (cf. Section 4.2).

3.4.2 Extension

The notion of execution from Section 4.2, which we adopt as the temporal accessibility relation for tCPL, generates the following two kinds of timed events: $\mathtt{N}(a, x, n, (o, V))$ for the generation of name n with intended owners o and *temporal validity* $V := (t_b, t_e)$ for the declaration of the intended beginning (t_b) and end (t_e) of validity of the generated name (typically a key) by agent a in session x, and $\mathtt{S}(a, x, t)$ for the setting of a's *local clock* to clock value t by a in session x. By convention, these events are unobservable by the adversary, i.e., they are secure. $t \in \mathcal{CV} := \mathbb{Q}$ denotes *clock values* having the associated type `CV`, and $t_b, t_e \in \mathcal{TV} := \mathcal{CV} \cup \{-\infty, \infty\}$ denote *time values* having the associated type `TV`. Time values are transmittable as messages.

The syntactic and semantic novelties are the following:

1. addition of two new, binary relational symbols $\leq$ and @ (overloading the session locality symbol) forming atomic formulae $E \leq E'$ and $E@a$, for

the comparison of temporal expressions (calculation of temporal intervals and bounds) $E ::= t \mid E + E \mid E - E$ and the testing of agent a's local clock with time E, respectively. Their truth denotation is as follows:

$$[\![E \leq E']\!]^i_{\mathfrak{p}} := ([\![E]\!] \text{ is smaller than or equal to } [\![E']\!], \emptyset)$$

where $[\![\cdot]\!]$ designates the obvious evaluation function from temporal expressions to time values (not to be confused with the function of truth denotation $[\![\cdot]\!]^i_{\mathfrak{p}}$); and

$$[\![E@a]\!]^i_{\mathfrak{p}} := ([\![E]\!] = t + \delta_a \cdot \Delta, \{\mathtt{S}(a, x, t), \mathtt{S}(\mathtt{Eve}, \blacksquare, t_i)\})$$

where

- t designates the clock value of a's last clock-set event in $\mathfrak{h}$, i.e., there are $\mathfrak{h}_1, \mathfrak{h}_2$, x s.t. $\mathfrak{h} = \mathfrak{h}_1 \cdot \mathtt{S}(a, x, t) \circ \mathfrak{h}_2$ and there is no x', t' s.t. $\mathtt{S}(a, x', t') \in \dot{\mathfrak{h}}_2$
- $\delta_a \in \mathcal{TV}$ designates the drift rate of a's local clock
- Δ designates the temporal difference between Eve's last clock-set event before $\mathtt{S}(a, x, t)$ and Eve's last clock-set event so far in $\mathfrak{h}$, i.e.,

$$\Delta = \begin{cases} t_2 - t_1 & \text{if for } i \in \{1, 2\} \text{ there are } \mathfrak{h}_{i'}, \mathfrak{h}''_i,\ t_i \text{ s.t.} \\ & \quad \mathfrak{h}_i = \mathfrak{h}'_i \cdot \mathtt{S}(\mathtt{Eve}, \blacksquare, t_i) \circ \mathfrak{h}''_i \text{ and there is no } t'_i \text{ s.t.} \\ & \quad \mathtt{S}(\mathtt{Eve}, \blacksquare, t'_i) \in \dot{\mathfrak{h}}''_i, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

- $\blacksquare$ serves as a dummy session identifier for Eve's clock-set events

2. refinement (i.e., one new parameter) of the relational symbol for new-name generation $\circlearrowleft$ with a *validity tag* $V := (t_b, t_e)$ for the declaration of the intended beginning ($t_b \in \mathcal{TV}$) and end ($t_e \in \mathcal{TV}$) of validity of the generated name (typically a key). Its truth denotation is the following:

$$[\![a \circlearrowleft n.o.V)]\!]^i_{\mathfrak{p}} := (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{\mathcal{X}}} \{\mathtt{N}(a, x, n, (o, V))\} \cap \dot{\mathfrak{h}}$$

3. refinement (i.e., adding of one new defining rule) of the relation $\vdash^{\mathcal{E}}_a \subseteq \mathcal{H} \times \mathcal{M}$ for the derivation of individual knowledge (cf. Table 3.3) with adversarial break of short-term keys (k) enabled jointly by *key expiration* ($\mathsf{expired}(k)$) and the existence of a *ciphertext-only attack* on the key ($\mathfrak{h}' \vdash^{\mathcal{E}}_{\mathtt{Eve}} \{\!|M|\!\}_k$):

$$\frac{\mathfrak{h}' \vdash^{\mathcal{E}}_{\mathtt{Eve}} \{\!|M|\!\}_k}{\mathfrak{h} \vdash^{\mathcal{E}}_{\mathtt{Eve}} k} \quad \begin{array}{rl} & \mathfrak{h}' \text{ is a prefix of } \mathfrak{h}, \text{ and there is } t \in \mathcal{TV} \text{ s.t.} \\ \mathfrak{h}' & \models t@\mathtt{Eve} \text{ and} \\ \mathfrak{h} & \models \mathsf{expired}(k) \wedge \\ & \quad \exists t_v (t_v \ \mathsf{validityOf}\ k \wedge \exists t_n (t_n@\mathtt{Eve} \wedge t_v < t_n - t)) \end{array}$$

where t_v designates the duration of validity of the considered key (i.e., the strength of the key, corresponding to its length in a bit-string representation), and $t_n - t$ the duration of the attack on the considered key (i.e., the time during which the corresponding ciphertext has been known to the adversary, and during which the adversary has potentially been attacking

— i.e., performing computations on — the ciphertext in order to recover the desired key); and

$$\begin{array}{rcl} \mathsf{expired}(k) & := & \exists t_n(t_n@\mathtt{Eve} \wedge \exists t_e(k \ \mathsf{validUntil} \ t_e \wedge t_e < t_n)) \\ k \ \mathsf{validUntil} \ t_e & := & \exists t_b(k \ \mathsf{validBetween} \ (t_b, t_e)) \\ k \ \mathsf{validBetween} \ (t_b, t_e) & := & \exists a \exists o(a \circlearrowright k.o.(t_b, t_e)) \\ t_v \ \mathsf{validityOf} \ k & := & k \ \mathsf{validBetween} \ (t_b, t_e) \wedge t_e - t_b = t_v \end{array}$$

4. refinement (i.e., one new conjunct) of the state of violation Σ with *key expiration* in the truth condition of the permission modality (cf. Table 3.4):

$$\Sigma := \exists(k : \mathtt{CK})(\mathtt{Eve} \ \mathsf{k} \ k \wedge \neg \, k \ \mathsf{ck} \ \mathtt{Eve} \wedge \boxed{\neg \mathsf{expired}(k)})$$

3.4.3 Expressiveness

We demonstrate the expressiveness of tCPL on the macro-definability of important modalities from general-purpose timed logics:

- *point*-parametrised *future*-time (similarly for *past*-time) modalities (so-called *freeze quantifiers*):

$$\boxplus_t(\phi) := \boxplus(t@\mathtt{Eve} \rightarrow \phi) \qquad \diamondplus_t(\phi) := \neg \boxplus_t (\neg\phi)$$

- *interval*-parametrised *future*-time (similarly for *past*-time) modalities with an:
  - *absolute*-time understanding of *closed* (similarly for *open*) intervals $[t_1, t_2]$:

$$\begin{array}{rcl} \boxplus_{[t_1,t_2]}(\phi) & := & \forall t(t_1 \leq t \leq t_2 \rightarrow \boxplus_t(\phi)) \\ \diamondplus_{[t_1,t_2]}(\phi) & := & \neg \boxplus_{[t_1,t_2]} (\neg\phi) \end{array}$$

  - understanding of intervals that is *relative* to the current time $t@\mathtt{Eve}$:

$$\begin{array}{rcl} \boxplus_{[\Delta]}(\phi) & := & \forall t(t@\mathtt{Eve} \rightarrow \boxplus_{[t,t+\Delta]}(\phi)) \\ \diamondplus_{[\Delta]}(\phi) & := & \forall t(t@\mathtt{Eve} \rightarrow \diamondplus_{[t,t+\Delta]}(\phi)) \end{array}$$

- the *chop* connective:

$$\phi \frown_{[t,t']} \phi' := \exists t''(\boxplus_{[t,t'']}(\phi) \wedge \boxplus_{[t'',t']}(\phi'))$$

- *durations* [ZHR91], [ZH04] (cf. Table 3.11)

The cryptographic states of affairs involving qualitative temporal modalities from Section 3.3 can easily be quantitatively adapted by replacing the qualitative temporal modalities by the above quantitative ones with actual time values (points and/or intervals) as desired.

Table 3.11: Definability of durations

$$
\begin{aligned}
\Delta\ \mathsf{duration}_{(t,t')}\ \phi \quad &:= \quad \Diamond\!\!\!\!-_t(\Delta\ \mathsf{duration}_{t'}\ \phi) \vee \Diamond\!\!\!\!+_t(\Delta\ \mathsf{duration}_{t'}\ \phi) \\
\Delta\ \mathsf{duration}_{t'}\ \phi \quad &:= \quad (\phi \rightarrow \forall t_d(t_d@\mathtt{Eve} \rightarrow \\
&\qquad \bigoplus((\phi \rightarrow \forall t_m((t_m@\mathtt{Eve} \wedge t_m \leq t') \rightarrow \\
&\qquad\qquad \Delta - (t_m - t_d)\ \mathsf{duration}_{t'}\ \phi)) \wedge \\
&\qquad (\neg\phi \rightarrow \bigoplus(\Delta\ \mathsf{duration}_{t'}\ \phi))))) \wedge \\
&\quad (\neg\phi \rightarrow \bigoplus(\Delta\ \mathsf{duration}_{t'}\ \phi))
\end{aligned}
$$

3.4.4 Application: a timed attack scenario

We exemplify our concept of attack scenario in the timed setting with another popular attack on a cryptographic protocol, namely the man-in-the-middle attack on the Wide-Mouthed-Frog protocol (WMF) (cf. Table 3.12). WMF is a server-based, (session) key-transport protocol employing symmetric cryptography intended to guarantee timely, unacknowledged transport of a session key between an initiator and a responder mediating a trusted third party (the server). Timeliness of key transport means that the responder only accepts session keys within a fixed interval of time. The protocol presumes that the long-term sym-

Table 3.12: Protocol narration for WMF

$$
\begin{aligned}
&1a. \quad \mathtt{Alice} \rightarrow \mathtt{Trent} \ : \ \mathtt{Alice} \\
&1b. \quad \mathtt{Alice} \rightarrow \mathtt{Trent} \ : \ \{\!|((t_{\mathtt{Alice}}, \mathtt{Bob}), k_{\mathtt{AliceBob}})|\!\}_{k_{\mathtt{AliceTrent}}} \\
&2. \quad \mathtt{Trent} \rightarrow \mathtt{Bob} \ : \ \{\!|((t_{\mathtt{Trent}}, \mathtt{Alice}), k_{\mathtt{AliceBob}})|\!\}_{k_{\mathtt{BobTrent}}}
\end{aligned}
$$

metric keys (e.g., $k_{\mathtt{AliceTrent}}$ and $k_{\mathtt{BobTrent}}$) between the initiator (`Alice`) and the server (`Trent`) and between the responder (`Bob`) and the server have already been generated by the server and established with all other corresponding clients.

The intention of each protocol step is as follows: the intention of the first step is to announce the initiator to the server; the intention of the second step is twofold, i.e., to transport the session key (e.g., $k_{\mathtt{AliceBob}}$) from the initiator to the server and to solicit the server to transport the session key to the responder; the intention of the third step is twofold, i.e., to transport the session key from the server to the responder and to transmit from the server to the responder the intention of the initiator to communicate securely with the responder by means of the transported session key. The time stamps are from the initiator's and the server's local clock, respectively. Their purpose is to ensure freshness of the session key.

The protocol narration can be transcribed into a formal language, *for example* into the one of Section 4.2, a timed extension of the one of Section 4.1, by instantiating the protocol template displayed in Table 3.13 via substitution of `Alice` for *init*, `Trent` for *serv*, and `Bob` for *resp*; and choice of a positive time value for Δ_v, i.e., half the desired duration of validity of the transported key. Features of that language are: a double-purpose primitive for lookup of stored

Table 3.13: Protocol template for WMF

$\mathrm{WMF}_{\mathtt{INIT}}(slf, srv, oth, \Delta_v) :=$	$\mathrm{WMF}_{\mathtt{SERV}}(slf, \Delta_v) :=$	$\mathrm{WMF}_{\mathtt{RESP}}(slf, srv, \Delta_v) :=$
$\mathtt{Get}_{slf}\ (t_s : \mathtt{CV}, \blacksquare)\ \mathtt{in}$ $\mathtt{New}\ (k_{so} : \mathtt{K}, ((slf, oth), (t_s, t_s + \Delta_v + \Delta_v))).$ $\mathtt{Get}_{srv}\ (k_{ss} : \mathtt{K}, (slf, srv))\ \mathtt{in}$ $\mathtt{Out}_{srv}\ slf.$ $\mathtt{Out}_{srv}\ \{\!\lvert((t_s, oth), k_{so})\rvert\!\}_{k_{ss}}.1$	$\mathtt{In}\ fst\ \mathtt{when}\ fst : \mathtt{A}.$ $\mathtt{Get}_{slf}\ (k_{sf} : \mathtt{K}, (slf, fst))\ \mathtt{in}$ $\mathtt{In}\ \{\!\lvert((t, snd), key)\rvert\!\}_{\overline{k_{sf}}}\ \mathtt{when}\ t : \mathtt{TV} \wedge$ $\exists t_s(t_s : \mathtt{TV} \wedge t_s @ slf \wedge t + \Delta_v \leq t_s) \wedge$ $snd : \mathtt{A} \wedge$ $key : \mathtt{K}.$ $\mathtt{Get}_{slf}\ (k_{ss} : \mathtt{K}, (slf, snd))\ \mathtt{in}$ $\mathtt{Out}_{snd}\ \{\!\lvert((t_s, fst), key)\rvert\!\}_{k_{ss}}.1$	$\mathtt{Get}_{srv}\ (k_{ss} : \mathtt{K}, (slf, srv))\ \mathtt{in}$ $\mathtt{In}\ \{\!\lvert((t, oth), key)\rvert\!\}_{\overline{k_{ss}}}\ \mathtt{when}\ t : \mathtt{TV} \wedge$ $\exists t_s(t_s : \mathtt{TV} \wedge t_s @ slf \wedge t + \Delta_v \leq t_s) \wedge$ $oth : \mathtt{A} \wedge$ $key : \mathtt{K}.1$
$\mathrm{WMF}(init, srv, resp, x_{init}, x_{serv}, x_{resp}, \Delta_v) := init.x_{init}[\,\mathrm{WMF}_{\mathtt{INIT}}(init, srv, resp, \Delta_v)\,] \mid\mid\mid$ $srv.x_{srv}[\,\mathrm{WMF}_{\mathtt{SERV}}(srv, \Delta_v)\,] \mid\mid\mid$ $resp.x_{resp}[\,\mathrm{WMF}_{\mathtt{RESP}}(resp, srv, \Delta_v)\,]$		

keys and (local) time, an input primitive with pattern-matching and guard, and primitives for out-of-band communication. The left (right) column of the table defines the initiator (responder) role, and the middle column the server role. The bottom row defines the protocol template, distributing (via parallel composition) the roles at the corresponding locations $init.x_{init}[\cdot]$, $srv.x_{srv}[\cdot]$, and $resp.x_{resp}[\cdot]$, respectively. Observe that lookup of local time is done in two different ways, namely *imperatively* by means of the get-instruction (with ■ serving as a dummy owner), and *declaratively* by means of the @-predicate.

The previously mentioned assumption about preliminary symmetric key generation and establishment can be modelled by means of corresponding key-generation and out-of-band communication events, chained up to form the protocol prehistory displayed in Table 3.14. Observe that the prehistory includes set events for the resetting of all local clocks (with ■ serving as a dummy session identifier for Eve's set event).

Table 3.14: Prehistory for WMF

$$\begin{aligned}\mathfrak{h} := \epsilon \cdot\ & \mathtt{N}(\mathtt{Trent}, x_{t0}, k_{\mathtt{AliceTrent}}, ((\mathtt{Trent}, \mathtt{Alice}), (-\infty, \infty))) \cdot \\ & \mathtt{N}(\mathtt{Trent}, x_{t0}, k_{\mathtt{BobTrent}}, ((\mathtt{Trent}, \mathtt{Bob}), (-\infty, \infty))) \cdot \\ & \mathtt{sO}(\mathtt{Trent}, x_{t0}, k_{\mathtt{AliceTrent}}, \mathtt{Alice}) \cdot \mathtt{sI}(\mathtt{Alice}, x_{a0}, k_{\mathtt{AliceTrent}}, \mathtt{Trent}) \cdot \\ & \mathtt{sO}(\mathtt{Trent}, x_{t0}, k_{\mathtt{BobTrent}}, \mathtt{Bob}) \cdot \mathtt{sI}(\mathtt{Bob}, x_{b0}, k_{\mathtt{BobTrent}}, \mathtt{Trent}) \cdot \\ & \mathtt{S}(\mathtt{Eve}, ■, 0) \cdot \mathtt{S}(\mathtt{Alice}, x_{a0}, 0) \cdot \mathtt{S}(\mathtt{Bob}, x_{b0}, 0) \cdot \mathtt{S}(\mathtt{Trent}, x_{t0}, 0)\end{aligned}$$

This completes the definition of the initial state

$$(\mathfrak{h}, \mathrm{WMF}(\mathtt{Alice}, \mathtt{Trent}, \mathtt{Bob}, x_{a1}, x_{t1}, x_{b1}, \Delta_v))$$

of (our attack scenario for) WMF.

Table 3.15 displays the narration of the actual attack. The attack can be orchestrated by an *active outsider adversary* that performs *interception*, *impersonation*, and *reflection* (i.e., replay to the *same* agent) across three different, interleaved sessions. However, the attack does not exploit drifting of local clocks (i.e., all drift rates are 1). It consists in:

1. Eve impersonating Bob in the face of Trent by reflecting back to Trent a previously intercepted service reply $\{|((t_{\mathtt{Trent}}, \mathtt{Alice}), k_{\mathtt{AliceBob}})|\}_{k_{\mathtt{BobTrent}}}$ —

Table 3.15: Attack narration for WMF

$$\begin{array}{lrcll}
1a'. & \mathtt{Eve}_{\mathtt{Bob}} & \rightarrow & \mathtt{Trent} & : \mathtt{Bob} \\
1b'. & \mathtt{Eve}_{\mathtt{Bob}} & \rightarrow & \mathtt{Trent} & : \{|((t_{\mathtt{Trent}}, \mathtt{Alice}), k_{\mathtt{AliceBob}})|\}_{k_{\mathtt{BobTrent}}} \\
2'. & \mathtt{Trent} & \rightarrow & \mathtt{Eve}_{\mathtt{Alice}} & : \{|((t'_{\mathtt{Trent}}, \mathtt{Bob}), k_{\mathtt{AliceBob}})|\}_{k_{\mathtt{AliceTrent}}} \\
1a''. & \mathtt{Eve}_{\mathtt{Alice}} & \rightarrow & \mathtt{Trent} & : \mathtt{Alice} \\
1b''. & \mathtt{Eve}_{\mathtt{Alice}} & \rightarrow & \mathtt{Trent} & : \{|((t'_{\mathtt{Trent}}, \mathtt{Bob}), k_{\mathtt{AliceBob}})|\}_{k_{\mathtt{AliceTrent}}} \\
2''. & \mathtt{Trent} & \rightarrow & \mathtt{Bob} & : \{|((t''_{\mathtt{Trent}}, \mathtt{Alice}), k_{\mathtt{AliceBob}})|\}_{k_{\mathtt{BobTrent}}}
\end{array}$$

perceived as a service *request* from Bob by (forgetful) Trent — from Trent to Bob to a service request from Alice

2. Eve intercepting Trent's service reply $\{\!|((t'_{\mathtt{Trent}}, \mathtt{Bob}), k_{\mathtt{AliceBob}})|\!\}_{k_{\mathtt{AliceTrent}}}$ destined to Alice

3. Eve impersonating Alice in the face of Trent by reflecting back to Trent Trent's service reply destined to Alice — perceived as a service *request* from Alice by (forgetful) Trent — and Trent marshalling the corresponding service reply $\{\!|((t''_{\mathtt{Trent}}, \mathtt{Alice}), k_{\mathtt{AliceBob}})|\!\}_{k_{\mathtt{BobTrent}}}$ to Bob.

In result, Bob accepts a session key that possibly is stale due to Eve achieving first *delay of key delivery* through repeated reflection of service replies from Trent back to Trent, and second *prolongation of key validity* through Trent, who, on each reflection, trustingly restamps the key with a new time stamp (cf. $t'_{\mathtt{Trent}}$ and $t''_{\mathtt{Trent}}$) of his, each time more advanced, local clock. The session key necessarily is stale when Eve delays each reflection by Δ_v time units. In sum, the protocol fails to achieve its requirement of timely, unacknowledged key transport between initiating Alice, mediating Trent, and responding Bob.

More formally, let

$$\begin{aligned} \mathsf{tuaKT}_{\Delta}(a,b) &:= \boxplus\forall(k:\mathtt{K})((\mathsf{K}_b(a \text{ authored } k) \rightarrow \mathsf{K}_b(k \text{ sharedSecret } (a,b))) \rightarrow \\ &\qquad \diamondminus_{[\Delta]}(a \text{ authored } k)) \\ Q &:= \mathtt{Trent}.x_{t2}[\,\mathrm{WMF}_{\mathtt{SERV}}(\mathtt{Trent}, \Delta_v)\,] \;|||\; \\ &\quad \mathtt{Trent}.x_{t3}[\,\mathrm{WMF}_{\mathtt{SERV}}(\mathtt{Trent}, \Delta_v)\,] \end{aligned}$$

Then:

- $(\mathfrak{h}, Q) \in \mathcal{H} \times \mathcal{P}$ and
- $(\mathfrak{h}, Q) \models \mathsf{tuaKT}_{\Delta_v}(\mathtt{Eve}, \mathtt{Trent}) \otimes \mathsf{tuaKT}_{\Delta_v}(\mathtt{Eve}, \mathtt{Trent})$ and
- $(\mathfrak{h} \circ \mathfrak{h}, \mathrm{WMF}(\mathtt{Alice}, \mathtt{Trent}, \mathtt{Bob}, x_{a1}, x_{t1}, x_{b1}, \Delta_v) \;|||\; Q) \models$ $\neg\mathsf{tuaKT}_{\Delta_v+\Delta_v}(\mathtt{Alice}, \mathtt{Bob})$

by which we obtain

$$\begin{aligned} (\mathfrak{h}, \mathrm{WMF}(\mathtt{Alice}, \mathtt{Trent}, \mathtt{Bob}, x_{a1}, x_{t1}, x_{b1}, \Delta_v)) \models \\ (\mathsf{tuaKT}_{\Delta_v}(\mathtt{Eve}, \mathtt{Trent}) \otimes \mathsf{tuaKT}_{\Delta_v}(\mathtt{Eve}, \mathtt{Trent})) \blacktriangleright \\ \neg\mathsf{tuaKT}_{\Delta_v+\Delta_v}(\mathtt{Alice}, \mathtt{Bob}) \end{aligned}$$

representing our (property-based or logical) attack scenario for WMF. We invite the reader to compare this scenario to the corresponding, model-based (or process-algebraic) attack scenario described in Section 4.2.2.

Chapter 4

Calculus of Cryptographic Communication

We exemplify the temporal accessibility relation $\longrightarrow \subseteq (\mathcal{H} \times \mathcal{P})^2$ of CPL with a reduction relation, defined by a *reduction calculus*[1] C^3 (or *operational semantics* in the jargon of formal program semantics), on protocol states $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$.

Based on the operational semantics, we define an *observational equivalence* $\approx_a^* \subseteq (\mathcal{H} \times \mathcal{P})^2$ on protocol states w.r.t. some observing agent $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$. The purpose of that observational equivalence is *algebraic* analysis of cryptographic protocols. We illustrate this purpose on the application of the observational equivalence to the algebraic analysis of the attack scenario analysed logically in Section 3.3.5.1. Further, we extend C^3 with real time to tC^3, and illustrate the use of this *timed calculus* on the application of the (timed) observational equivalence to the algebraic analysis of the timed attack scenario analysed logically in Section 3.4.4. Furthermore, we define a *denotational semantics* of cryptographic protocols. The purpose of denotational semantics in general is to define *meaning* — in our case, the meaning of a cryptographic protocol. From the meaning of a cryptographic protocol, we obtain natural definitions of the concepts of (1) a protocol *invariant*, (2) protocol *safety*, (3) protocol *refinement*, and (4) protocol *information content*.

Co-design The programming language $\mathcal{P}$ for cryptographic protocols that we define is complementary to the logical language $\mathcal{F}$ of CPL. The complementarity of the two languages is the result of five natural, but notwithstanding novel (except for Item 4 and 5) integrating design decisions: (1) define the meaning of a cryptographic protocol (its denotational semantics) in terms of the meaning of the cryptographic messages it produces during its execution, i.e., define the *what* (denotation) in terms of the *how* (operation); (2) define the meaning of a cryptographic message in terms of the *propositional* knowledge a protocol agent *would* acquire from the (*individual*) knowledge of that message; (3) define *dynamic* observational equivalence (indistinguishability of execution *paths*) in terms of *static* observational equivalence (indistinguishability of execution

[1] In general, a *calculus* is a set of axioms and rules inductively defining a set. Typically, the defined set is a binary relation, e.g., a relation of *de*duction (a proof system for some logical language) or *re*duction (an operational semantics for some programming language).

states); (4) identify *static observational equivalence* with *epistemic accessibility* $\approx_a \subseteq (\mathcal{H} \times \mathcal{P})^2$ (the semantics of propositional knowledge); and (5) identify the *operational semantics* (protocol execution) with *temporal accessibility* (the semantics of temporal propositions).

From the (denotational) meaning of a cryptographic message, we obtain (1) an equational definition of its *context-sensitivity*, and (2) a *formalisation* of the first of Abadi and Needham's principles for prudent engineering practice for cryptographic protocols. Last but not least, we show that *protocol agents* can be conceived as evolving *Scott information systems*.

4.1 Core calculus

4.1.1 Syntax

Our programming language $\mathcal{P}$ for cryptographic protocols provides high-level linguistic abstractions P for the computation and agent-based communication of cryptographic messages $M \in \mathcal{M}$. We refer to its programs $P \in \mathcal{P}$ as C^3-processes. Our communication model is based on agents rather than channels for the sake of separating specification (high-level) from implementation (low-level) concerns.

C^3-processes are parallel compositions of *located threads*. A non-idle thread T located at the agent c and session x, written $c.x[\,T\,]$, has either an action prefix or a lookup prefix. The action prefixes $\texttt{Out}_a\ F$ and $\texttt{sOut}_a\ F$ stipulate insecure (intercepted by the adversary) resp. secure (unobservable by the adversary) output of F[2] to a; $\texttt{In}\ \Pi\ \texttt{when}\ \varphi$ and $\texttt{sIn}_a\ \Pi\ \texttt{when}\ \varphi$ stipulate insecure resp. secure input (from a) of a message matching the pattern Π and having the property φ; and $\texttt{New}\ (v : \varsigma, O)$ stipulates the generation and binding to the variable v of a fresh name of type ς *tagged* with O, where O (a tuple of agent names) stipulates intended ownership. The lookup prefix $\texttt{Get}_a\ (v : \varsigma, O)\ \texttt{in}$ stipulates the lookup and binding to v of a name of type ς generated by a with ownership tag O.

Definition 10 (C^3-processes) *C^3-processes $P \in \mathcal{P}$ are defined in Table 4.1. There, $a, b \in \boldsymbol{\mathcal{A}} \cup \mathcal{V}$. A process P is* epistemically local *:iff for all located threads $c.x[\,T\,]$ in P and for all $a \circlearrowleft n.O$ and $a\ \texttt{k}\ F$ in T, $a = c$. So-called* implementation processes *must be epistemically local, whereas so-called* specification processes *need not be. In addition, in implementation processes $\forall v$ may not bind v in any F, whereas in specification processes this need not be.*

Action prefixes π, as opposed to the lookup prefix, generate events when executed. Action prefixes for secure I/O generate unobservable events; they model out-of-band communication such as trusted couriers, personal contact between communicating parties, and dedicated communication links. The same prefixes can also be used for (1) encoding extensions to the core calculus, such as *vertical composition*, i.e., sub-protocol calls (cf. [BKN06, Section 3.2]), and *conditionals* (i.e., execution guards); (2) defining *specification processes* relied upon by equivalence-based specification of secrecy *à la* Spi-Calculus; and (3) defining *initialisation traces* (cf. Table 3.9 and 3.14). The purpose of including

[2] recall that F denotes *message forms*, i.e., messages with variables, which are used in processes where they may instantiate to (transmittable) *messages*

Table 4.1: C^3-processes

$$
\begin{array}{lcl}
P & ::= & c.x[\,T\,] \mid P \,|||\, P \\
T & ::= & 1 \mid \pi.T \mid \mathtt{Get}_a\ (v : \varsigma, O)\ \mathtt{in}\ T \\
\pi & ::= & \mathtt{Out}_a\ F \mid \mathtt{sOut}_a\ F \mid \mathtt{In}\ \Pi\ \mathtt{when}\ \varphi \mid \mathtt{sIn}_a\ \Pi\ \mathtt{when}\ \varphi \mid \mathtt{New}\ (v : \varsigma, O) \\
\Pi & ::= & F \mid (\Pi, \Pi) \mid \{\!|\Pi|\!\}_{\overline{F}} \mid \{\!|\Pi|\!\}^{+}_{F} \mid \{\!|\Pi|\!\}^{-}_{\overline{F}} \\
\varphi & ::= & a \circlearrowright n.O \mid n : \sigma \mid a\ \mathtt{k}\ F \mid F \preccurlyeq F \mid \neg\varphi \mid \varphi \wedge \varphi \mid \forall v(\varphi)
\end{array}
$$

session identifiers x in locations $c.x[\,T\,]$ is to enable the factorisation of the history of a protocol execution into individual sessions (cf. Section 3.2.3.5).

Remark 5 *Names in C^3 are pure logical constants. In contrast, the concept of names in classical process calculi such as the Pi- and the Spi-Calculus is* hybrid *in the sense that their names have both bound-variable character via α-convertibility and logical-constant character via substitutability in free (input) variables. Philosophically speaking, a bound variable has* no *individuality, whereas a logical constant has* only *individuality. The advantage of the classical concept of names is that it does not demand a new (Pitts-style) new-name quantifier in the logical language.*

4.1.2 Semantics

C^3 provides *pattern-matching* in the style of [HJ04] as a high-level linguistic abstraction for *cryptographic computation*. The result of a pattern-message match is a substitution relating variables to matched subterms. Substitutions are partial functions from variables to messages, and are tacitly lifted to terms in the standard way. Matching is computed by the partial function match defined in Table 4.2. There, $\uplus$ denotes composition, if the domains of the operands are disjoint, and is otherwise undefined. For example, the pattern $\{\!|\Pi|\!\}^{-}_{\overline{v}}$ matches any message signed with the private key corresponding to the public key v. The line over the letter v is part of the pattern, recalling that the pattern matches a message encrypted with the key dual to v without needing this key (e.g., the private key of another agent) occur in the pattern term.

Table 4.2: Pattern matching

$$
\begin{array}{rcl}
\mathrm{match}(v, M) & := & \left\{{}^{M}/_{v}\right\} \\
\mathrm{match}(M, M) & := & \emptyset \\
\mathrm{match}((\Pi, \Pi'), (M, M')) & := & \mathrm{match}(\Pi, M) \uplus \mathrm{match}(\Pi', M') \\
\mathrm{match}(\{\!|\Pi|\!\}_{\overline{M'}}, \{\!|M|\!\}_{M'}) & := & \mathrm{match}(\Pi, M) \\
\mathrm{match}(\{\!|\Pi|\!\}^{+}_{\overline{p}}, \{\!|M|\!\}^{+}_{p^+}) & := & \mathrm{match}(\Pi, M) \\
\mathrm{match}(\{\!|\Pi|\!\}^{-}_{\overline{p^+}}, \{\!|M|\!\}^{-}_{p}) & := & \mathrm{match}(\Pi, M)
\end{array}
$$

A novel, integrating feature of C^3 is that all its *reduction constraints* (e.g., freshness of new names, input guards, key lookup constraints) are *CPL-definable* and decidable, and thus *checkable* via CPL-satisfaction.

The *key store* of an agent is induced (on the syntactic object level) by the protocol history. (The popular alternative of a lookup table on the semantic meta-level is, for syntactic purists, an obnoxious alternative because it would harm the syntactic uniformity of their framework.) Keys are looked up in the protocol history w.r.t. the local view of the retrieving agent and by referring to their creator and the tag they were given at creation. A lookup succeeds only if the desired tag is a subterm of the one used at the creation of the key. Our lookup policy is *CPL-definable*: $\mathsf{looksUp}(c, x, n, \varsigma, d, O) \in \mathcal{F}$, pronounced "$c$ in session x looks up n of type ς generated by d with a tag containing O", is defined in Table 4.3 (cf. Appendix B for the macro-definitions of the employed auxiliary predicates). The policy enforces that the retrieved key (1) has the desired type ($n : \varsigma$); (2) was known by c ($c \mathsf{k} n$); (3) was generated by d ($d \circlearrowleft v.o$); (4) has a compatible, intended ownership ($O \preccurlyeq o$); and (5) when a session key, was received in the current session ($\exists m(c.x \leftarrow m \wedge n \preccurlyeq m)$).

Table 4.3: Lookup predicate

$$\begin{aligned}\mathsf{looksUp}(c, x, n, \varsigma, d, O) := n : \varsigma \wedge c \mathsf{k} n \wedge \exists v \exists o(d \circlearrowleft v.o \wedge (n = v \vee n = v^+) \wedge \\ O \preccurlyeq o \wedge (n : \mathtt{K}^1 \rightarrow \exists m(c.x \leftarrow m \wedge n \preccurlyeq m)))\end{aligned}$$

Definition 11 (Calculus of Cryptographic Communication) *Let* $\longrightarrow \subseteq (\mathcal{H} \times \mathcal{P})^2$*, defined in Table 4.4, designate reduction of protocol states* $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$*. Then,*

$$C^3 := \langle \mathcal{H} \times \mathcal{P}, \longrightarrow \rangle.$$

The generation of a new name (Rule New and New-Eve) is possible only if that name has not been generated yet, i.e., generated names are always *fresh* w.r.t. the current state. The adversary `Eve` may generate a new name at any time. Insecure input (Rule In) is incepted by the adversary and may consist in any message from her knowledge that matches the input pattern Π and that satisfies the input constraint φ. Successful input results in the substitution of the matching message parts for the matched variables in the receiving thread. Secure communication (Rule sOut, sIn and sCom-L, with sCom-R being tacit) is synchronous. To achieve this, we introduce two auxiliary transition relations $\xrightarrow{\mathtt{sI}}$ and $\xrightarrow{\mathtt{sO}}$ not visible on the top level. Insecure communication between two legitimate agents is asynchronous because it goes through the adversary, and secure communication is synchronous because it does not. Reduction of parallel processes happens via *interleaving concurrency* (Rule Par-L, Par-R being tacit). Finally, observe how *non-determinism* abstracts away three determining choices in the execution of a protocol, i.e., the choice of (1) the message sent by the adversary in an insecure input, (2) the new name selected at name generation time, and (3) the scheduling of located threads.

Slogan 19 *Actions are potentiality of effect — events, actuality.*

Table 4.4: Process and thread execution

Below, $\xrightarrow{\alpha} \in \{\longrightarrow, \xrightarrow{\texttt{sO}}, \xrightarrow{\texttt{sI}}\}$.

$$\textsc{New}\ \frac{\mathfrak{h} \models n : \varsigma \wedge \neg\exists a \exists o (a \circlearrowleft n.o)}{\begin{pmatrix} c.x[\,\texttt{New}\ (v : \varsigma, O).T\,] \\ \mathfrak{h} \end{pmatrix} \longrightarrow \begin{pmatrix} c.x[\,\{^{n}/_{v}\}T\,] \\ \mathfrak{h} \cdot \texttt{N}(c, x, n, O) \end{pmatrix}}$$

$$\textsc{New-Eve}\ \frac{\mathfrak{h} \models \texttt{Eve}\ \mathsf{k}\ O \wedge \neg\exists a \exists o (a \circlearrowleft n.o)}{\begin{pmatrix} P \\ \mathfrak{h} \end{pmatrix} \longrightarrow \begin{pmatrix} P \\ \mathfrak{h} \cdot \texttt{N}(\texttt{Eve}, \blacksquare, n, O) \end{pmatrix}}$$

$$\textsc{Out}\ \frac{}{\begin{pmatrix} c.x[\,\texttt{Out}_d\ M.T\,] \\ \mathfrak{h} \end{pmatrix} \longrightarrow \begin{pmatrix} c.x[\,T\,] \\ \mathfrak{h} \cdot \texttt{O}(c, x, M, d) \cdot \texttt{I}(\texttt{Eve}, \blacksquare, M) \end{pmatrix}}$$

$$\textsc{In}\ \frac{\mathfrak{h} \models \texttt{Eve}\ \mathsf{k}\ M \wedge \mathrm{match}(\Pi, M)\varphi}{\begin{pmatrix} c.x[\,\texttt{In}\ \Pi\ \texttt{when}\ \varphi.T\,] \\ \mathfrak{h} \end{pmatrix} \longrightarrow \begin{pmatrix} c.x[\,\mathrm{match}(\Pi, M)T\,] \\ \mathfrak{h} \cdot \texttt{O}(\texttt{Eve}, \blacksquare, M, c) \cdot \texttt{I}(c, x, M) \end{pmatrix}}$$

$$\textsc{sOut}\ \frac{}{\begin{pmatrix} c.x[\,\texttt{sOut}_d\ M.T\,] \\ \mathfrak{h} \end{pmatrix} \xrightarrow{\texttt{sO}} \begin{pmatrix} c.x[\,T\,] \\ \mathfrak{h} \cdot \texttt{sO}(c, x, M, d) \end{pmatrix}}$$

$$\textsc{sIn}\ \frac{\mathfrak{h} \models \mathrm{match}(\Pi, M)\varphi}{\begin{pmatrix} d.x[\,\texttt{sIn}_c\ \Pi\ \texttt{when}\ \varphi.T\,] \\ \mathfrak{h} \cdot \texttt{sO}(c, x', M, d) \end{pmatrix} \xrightarrow{\texttt{sI}} \begin{pmatrix} d.x[\,\mathrm{match}(\Pi, M)T\,] \\ \mathfrak{h} \cdot \texttt{sO}(c, x', M, d) \cdot \texttt{sI}(d, x, M, c) \end{pmatrix}}$$

$$\textsc{sCom-L}\ \frac{\begin{pmatrix} P \\ \mathfrak{h} \end{pmatrix} \xrightarrow{\texttt{sO}} \begin{pmatrix} P' \\ \mathfrak{h}' \end{pmatrix} \qquad \begin{pmatrix} Q \\ \mathfrak{h}' \end{pmatrix} \xrightarrow{\texttt{sI}} \begin{pmatrix} Q' \\ \mathfrak{h}'' \end{pmatrix}}{\begin{pmatrix} P \mid\mid\mid Q \\ \mathfrak{h} \end{pmatrix} \longrightarrow \begin{pmatrix} P' \mid\mid\mid Q' \\ \mathfrak{h}'' \end{pmatrix}}$$

$$\textsc{LookUp}\ \frac{\begin{pmatrix} c.x[\,\{^{n}/_{v}\}T\,] \\ \mathfrak{h} \end{pmatrix} \xrightarrow{\alpha} \begin{pmatrix} c.x[\,T'\,] \\ \mathfrak{h}' \end{pmatrix} \qquad \mathfrak{h} \models \mathsf{looksUp}(c, x, n, \varsigma, d, O)}{\begin{pmatrix} c.x[\,\texttt{Get}_d\ (v : \varsigma, O)\ \texttt{in}\ T\,] \\ \mathfrak{h} \end{pmatrix} \xrightarrow{\alpha} \begin{pmatrix} c.x[\,T'\,] \\ \mathfrak{h}' \end{pmatrix}}$$

$$\textsc{Par-L}\ \frac{\begin{pmatrix} P \\ \mathfrak{h} \end{pmatrix} \xrightarrow{\alpha} \begin{pmatrix} P' \\ \mathfrak{h}' \end{pmatrix}}{\begin{pmatrix} P \mid\mid\mid Q \\ \mathfrak{h} \end{pmatrix} \xrightarrow{\alpha} \begin{pmatrix} P' \mid\mid\mid Q \\ \mathfrak{h}' \end{pmatrix}}$$

4.1.3 Observational equivalence

Our notion of observational equivalence is a compromise between trace equivalence and standard bisimilarity. More precisely, it is a bisimilarity on protocol states w.r.t. the execution *paths* (as opposed to single execution *steps*) producible from those states, and w.r.t. a given observing protocol agent.

Definition 12 (Dynamic observational equivalence) *Let* $\mathfrak{s}_1, \mathfrak{s}_2 \in \mathcal{H} \times \mathcal{P}$ *and* $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$*. Then,*

- $\mathfrak{s}_1$ observationally refines $\mathfrak{s}_2$ *from the viewpoint of* a*, written* $\mathfrak{s}_1 \lessapprox_a^* \mathfrak{s}_2$*, :iff for all* $\mathfrak{s}_1' \in \mathcal{H} \times \mathcal{P}$*, if* $\mathfrak{s}_1 \longrightarrow^* \mathfrak{s}_1'$ *then there is* $\mathfrak{s}_2' \in \mathcal{H} \times \mathcal{P}$ *s.t.* $\mathfrak{s}_2 \longrightarrow^* \mathfrak{s}_2'$ *and* $\mathfrak{s}_1' \approx_a \mathfrak{s}_2'$*; and*
- $\mathfrak{s}_1$ *and* $\mathfrak{s}_2$ *are* observationally equivalent *from the viewpoint of* a*, written* $\mathfrak{s}_1 \approx_a^* \mathfrak{s}_2$*, :iff* $\mathfrak{s}_1 \lessapprox_a^* \mathfrak{s}_2$ *and* $\mathfrak{s}_2 \lessapprox_a^* \mathfrak{s}_1$*.*

Observe how *dynamic* observational equivalence (on protocol states) is defined in terms of *static* observational equivalence on protocol states, i.e., *epistemic accessibility* ($\approx_a$, cf. Definition 7) between protocol states. As previously stated, the idea of identifying an observational equivalence with epistemic accessibility seems to have been published first in [HS04b]. However, the authors adopt a very different approach based on so-called function views.

4.1.4 Application: an algebraic attack scenario

We recall that algebraic correctness statements of cryptographic protocols enunciate a purported observational equivalence between two models (processes) of the protocol under scrutiny. The choice of the actual processes depends on the cryptographic requirement that the correctness statement is intended to encode. For example, authenticity can be encoded as an observational equivalence between, on the one side, an obviously (via different kinds of "magic") correct specification and, on the other side, an implementation expressing the protocol as it would be coded in a realistic, and thus possibly less-obviously correct implementation.

In our case, we use our observational equivalence w.r.t. Eve's point of observation $\approx_{\mathtt{Eve}}^*$ and create the specification and implementation via a minor adaptation of the responder process. More precisely, we introduce an additional, formal parameter oth' in the template $\mathrm{NSPuK}_{\mathtt{RESP}}(slf)$ (cf. Table 3.8) and an additional conjunct φ in the guard of its first (insecure) input prefix

$$\mathtt{In}\ \{\!|(x_{oth}, oth)|\!\}^{+}_{k_{slf}}\ \mathtt{when}\ x_{oth} : \mathtt{X} \wedge oth : \mathtt{A}.$$

The modified responder process thus is

$$\mathrm{NSPuK}_{\mathtt{RESP}}(slf, oth') := \ldots\ \mathtt{In}\ \{\!|(x_{oth}, oth)|\!\}^{+}_{k_{slf}}\ \mathtt{when}\ x_{oth} : \mathtt{X} \wedge oth : \mathtt{A} \wedge \varphi \ldots$$

where '...' designates the parts left unchanged by the modification. The specification process NSPuK_{spec} is then obtained from $\mathrm{NSPuK}_{\mathtt{RESP}}$ by stipulating that $\varphi := (oth' = oth)$ (the authentication guarantee), and the implementation process NSPuK_{impl} by stipulating that $\varphi := \top$. Observe that the kind of magic

in our specification is represented by the passing of the name of the actual correspondent oth' to the responder *at the meta-level* (as opposed to the object level of message communication provided by the calculus), i.e., as a formal parameter.

An implementation set-up $\mathfrak{s}_{impl} \in \mathcal{H} \times \mathcal{P}$ that potentially yields, via process reduction, a trace corresponding to the attack narration displayed in Table 3.10 is

$$\mathfrak{s}_{impl} := (\mathfrak{h} \circ \mathfrak{h}',\ \mathtt{Alice}.x_{a1}[\,\mathrm{NSPuK}_{\mathtt{INIT}}(\mathtt{Alice},\mathtt{Eve})\,] \ |||\ \mathtt{Bob}.x_{b1}[\,\mathrm{NSPuK}_{impl}(\mathtt{Bob},\mathtt{Eve})\,])$$

where $\mathfrak{h} \circ \mathfrak{h}'$ is the same protocol prehistory as the one in the logical analysis of the same attack scenario (cf. Section 3.3.5.1). The trace being considered is defined in Table 4.5.

Table 4.5: Attack trace for NSPuK

$$\begin{aligned}
\mathfrak{h}_{impl} := \mathfrak{h} \circ \mathfrak{h}' \cdot\ & \\
& \mathtt{N}(\mathtt{Alice}, x_{a1}, x_{\mathtt{Alice}}, \blacksquare) \cdot \\
& \mathtt{O}(\mathtt{Alice}, x_{a1}, \{\!|(x_{\mathtt{Alice}}, \mathtt{Alice})|\!\}^{+}_{p^{+}_{\mathtt{Eve}}}, \mathtt{Eve}) \cdot \mathtt{I}(\mathtt{Eve}, \blacksquare, \{\!|(x_{\mathtt{Alice}}, \mathtt{Alice})|\!\}^{+}_{p^{+}_{\mathtt{Eve}}}) \cdot \\
& \mathtt{O}(\mathtt{Eve}, \blacksquare, \{\!|(x_{\mathtt{Alice}}, \mathtt{Alice})|\!\}^{+}_{p^{+}_{\mathtt{Bob}}}, \mathtt{Bob}) \cdot \mathtt{I}(\mathtt{Bob}, x_{b1}, \{\!|(x_{\mathtt{Alice}}, \mathtt{Alice})|\!\}^{+}_{p^{+}_{\mathtt{Bob}}}) \cdot \\
& \mathtt{N}(\mathtt{Bob}, x_{b1}, x_{\mathtt{Bob}}, \blacksquare) \cdot \\
& \mathtt{O}(\mathtt{Bob}, x_{b1}, \{\!|(x_{\mathtt{Alice}}, x_{\mathtt{Bob}})|\!\}^{+}_{p^{+}_{\mathtt{Alice}}}, \mathtt{Alice}) \cdot \mathtt{I}(\mathtt{Eve}, \blacksquare, \{\!|(x_{\mathtt{Alice}}, x_{\mathtt{Bob}})|\!\}^{+}_{p^{+}_{\mathtt{Alice}}}) \cdot \\
& \mathtt{O}(\mathtt{Eve}, \blacksquare, \{\!|(x_{\mathtt{Alice}}, x_{\mathtt{Bob}})|\!\}^{+}_{p^{+}_{\mathtt{Alice}}}, \mathtt{Alice}) \cdot \mathtt{I}(\mathtt{Alice}, x_{a1}, \{\!|(x_{\mathtt{Alice}}, x_{\mathtt{Bob}})|\!\}^{+}_{p^{+}_{\mathtt{Alice}}}) \cdot \\
& \mathtt{O}(\mathtt{Alice}, x_{a1}, \{\!|x_{\mathtt{Bob}}|\!\}^{+}_{p^{+}_{\mathtt{Eve}}}, \mathtt{Eve}) \cdot \mathtt{I}(\mathtt{Eve}, \blacksquare, \{\!|x_{\mathtt{Bob}}|\!\}^{+}_{p^{+}_{\mathtt{Eve}}}) \cdot \\
& \mathtt{O}(\mathtt{Eve}, \blacksquare, \{\!|x_{\mathtt{Bob}}|\!\}^{+}_{p^{+}_{\mathtt{Bob}}}, \mathtt{Bob}) \cdot \mathtt{I}(\mathtt{Bob}, x_{b1}, \{\!|x_{\mathtt{Bob}}|\!\}^{+}_{p^{+}_{\mathtt{Bob}}})
\end{aligned}$$

In contrast, our specification set-up

$$\mathfrak{s}_{spec} := (\mathfrak{h} \circ \mathfrak{h}',\ \mathtt{Alice}.x_{a1}[\,\mathrm{NSPuK}_{\mathtt{INIT}}(\mathtt{Alice},\mathtt{Eve})\,] \ |||\ \mathtt{Bob}.x_{b1}[\,\mathrm{NSPuK}_{spec}(\mathtt{Bob},\mathtt{Eve})\,])$$

cannot, due to the failure of the authentication guarantee ($\mathtt{Eve} \neq \mathtt{Alice}$), yield a trace $\mathfrak{h}_{spec}$ that matches the implementation trace $\mathfrak{h}_{impl}$ beyond the event $\mathtt{I}(\mathtt{Eve}, \blacksquare, \{\!|(x_{\mathtt{Alice}}, \mathtt{Alice})|\!\}^{+}_{p^{+}_{\mathtt{Eve}}})$. Hence, $\mathfrak{s}_{impl} \not\approx^{*}_{\mathtt{Eve}} \mathfrak{s}_{spec}$.

4.2 tC³: an extension of C³ with real time

For practical usability, special-purpose models of timed cryptographic protocols are preferable over their general-purpose counterparts because (untimed) general-purpose models tend to create considerable (en)coding overhead: "[...] the coding up required would make the complex behaviour difficult to understand, and it is preferable to use a language designed to express such real-time behaviour." [ES00].

Related work In tock-CSP [ES00], tCryptoSPA [GM04], and the timed Spi-Calculus [HJ05] time is *natural*-number valued, yielding a *discrete* time domain. tock-CSP and tCryptoSPA provide local processes that globally synchronise through so-called tock events resp. tick actions, which represent the passage of one unit of time. And the timed Spi-Calculus provides a process constructor for querying a global clock. Thus, tock-CSP, tCryptoSPA, and the timed Spi-Calculus lack local clocks that potentially advance at different rates across different processes/locations. As pointed out in Section 3.4.1, this lack becomes a deficiency in distributed systems. Hence, a faithful model of timed cryptographic protocols must allow for potentially desynchronised, local clocks.

In [BEL05] (which we will refer to as tBEL), time — in particular, a time stamp — is *real*-number valued, yielding a *dense* time domain. We contend that real-valued time-stamps are too fine-grained because cryptographic messages have finite length, which implies that real numbers are not transmittable as such. Moreover, real clocks only have finite precision. tBEL does provide local clocks, yet they "advance at the same rate as time." [BEL05, Page 2]. Further, adversarial break of short-term keys is modelled only indirectly with a parallel process rather than directly as part of the adversary model. Furthermore, tBEL lacks a process equivalence. On the other hand, tBEL comes with a (second-order) special-purpose logic for reasoning about tBEL models, and a decision procedure for a class of reachability properties of bounded protocols based on syntactic control points. In our opinion, tBEL and its associated logic are unnecessarily domain-specific. They seem to have been built from scratch rather than as Occham's-razor extensions of untimed formalisms. Adding real-time to a model or logic without explicit time can be simple [Lam05]. Moreover, the logic's — according to the authors, main — modality is actually *not* a modality (i.e., a non-truth-functional operator on *formulae*), but rather a relational symbol between *individuals* (and finite sets thereof).

Our approach In contrast to the models discussed, tC^3 (1) inherits the observational equivalence of its core C^3, and (2) extends C^3 with (2.1) *rational*-number valued time (which still is dense), (2.2) local clocks that may progress at different rates across different locations, and (2.3) adversarial break of short-term keys based on ciphertext-only attacks enabled by key expiration. C^3 neatly extends to tC^3 by maintaining backwards compatibility. Essentially, only two additional axioms (and no modified axioms/rules!) are needed in its operational semantics.

We extend C^3 to tC^3 according to the following two *design principles*: first, for every legitimate participant and the adversary, we introduce a rational-number valued, local clock with an associated drift rate, where the adversary's clock always displays the actual time; and second, we model the advancement of (real) time by means of the adversary, who, at any (logical) time of protocol execution, may advance the time by an arbitrary but always finite amount by advancing her local clock. In this way, the adversary is *de facto* in full control of time though subject to monotonicity. Adding communication and/or computation delays imposes non-zero lower bounds on the advancement of time between certain pairs of actions, and could be handled by only considering traces where the adversary respects these bounds.

4.2.1 Extension

Syntax

1. addition of an action prefix $\boxed{\texttt{Set}\ E}$ for the (re)setting of local clocks to a value E (cf. Section 3.4.2 for temporal expressions E)

2. extension of the ownership tag O of the action prefix $\texttt{New}\ (v : \varsigma, (O, \boxed{V}))$ for new-name generation with a tag V for the validity of the generated name (cf. Section 3.4.2 for validity tags V)

A process P is *epistemically local* :iff for all located threads $c.x[\,T\,]$ in P and for all $a \circlearrowleft n.(O, \boxed{V})$, $a\ \texttt{k}\ F$, and $\boxed{t@a}$ in T, $a = c$ (cf. Section 3.4.2 for the time predicate $t@a$ and the associated concept of drift rate).

Semantics

1. addition of the axiom SET to the operational semantics:

$$\textsc{Set}\ \frac{}{\begin{pmatrix} a.x[\,\texttt{Set}\ t.T\,] \\ \mathfrak{h} \end{pmatrix} \longrightarrow \begin{pmatrix} a.x[\,T\,] \\ \mathfrak{h} \cdot \texttt{S}(a, x, t) \end{pmatrix}}$$

2. addition of the axiom TIME to the operational semantics:

$$\textsc{Time}\ \frac{}{\begin{pmatrix} P \\ \mathfrak{h} \end{pmatrix} \longrightarrow \begin{pmatrix} P \\ \mathfrak{h} \cdot \texttt{S}(\texttt{Eve}, \blacksquare, t) \end{pmatrix}}\ \mathfrak{h} \models \exists t'(t'@\texttt{Eve} \wedge t' \leq t < \infty)$$

   where $\blacksquare$ acts as a dummy session identifier.

   Notice that the choice of the time value by which the adversary advances time is abstracted away by non-determinism.

3. adaptation of the lookup predicate (cf. Table 4.6) so that the retrieved name n either is the locally-measured time ($n@c$), or is a key that (1) has the desired type ($n : \varsigma$); (2) was known by c ($c\ \texttt{k}\ n$); (3) was generated by d ($d \circlearrowleft v.(o, (t_b, t_e))$); (4) has a compatible, intended ownership ($O \preccurlyeq o$); and (5) is perceived as timely by c ($\exists t_c(t_c@c \wedge t_b \leq t_c \leq t_e)$).

Table 4.6: Timed lookup predicate

$$\begin{aligned} \mathsf{looksUp}(c, x, n, \varsigma, d, O) := {} & n : \varsigma \wedge c\ \texttt{k}\ n \wedge (n@c \vee \\ & \exists v \exists o \exists t_b \exists t_e (d \circlearrowleft v.(o, (t_b, t_e)) \wedge \\ & (n = v \vee n = v^+) \wedge O \preccurlyeq o \wedge \exists t_c(t_c@c \wedge t_b \leq t_c \leq t_e))) \end{aligned}$$

Observe that thanks to sorts (cf. Section 3.4.2 for timed sorts), tags, and the abstract message the new-name and the lookup-prefix can also elegantly handle timed information: new-timed-key generation as $\texttt{New}\ (v : \texttt{K}, (O, V))$, timed-key lookup as $\texttt{Get}_a\ (v : \texttt{K}, O)\ \texttt{in}$, and time lookup as $\texttt{Get}_a\ (v : \texttt{CV}, \blacksquare)\ \texttt{in}$ with $\blacksquare$ acting as a dummy ownership tag.

Observational equivalence No redefinition is required: clock-set events are, by convention (cf. Section 3.4.2), unobservable, and unobservable events are generically covered by Clause 5 of Definition 6.

4.2.2 Application: a timed, algebraic attack scenario

We are interested in the timeliness requirement for WMF, namely that the responder only accepts the session key within a fixed interval of time (cf. Section 3.4.4). It turns out that the requirement can be checked in a similar set-up as for authenticity, i.e., as an equivalence from Eve's viewpoint ($\approx^*_{\mathtt{Eve}}$) between a specification and an implementation.

We create the specification and implementation again via a minor adaptation of the responder process. More precisely, we introduce (again) an additional guard-conjunct φ and (additionally) an additional action-prefix $\mathtt{Out}_{\mathtt{Eve}}\ t$ in the (insecure) input prefix

$$\begin{array}{l}\mathtt{In}\ \{\!|((t, oth), key)|\!\}_{\overline{k_{ss}}}\ \mathtt{when}\ t : \mathtt{TV} \wedge \\ \exists t_s(t_s : \mathtt{TV} \wedge t_s@slf \wedge t + \Delta_v \leq t_s) \wedge \\ oth : \mathtt{A} \wedge \\ key : \mathtt{K}.1\end{array}$$

of the template $\mathrm{WMF}_{\mathtt{RESP}}(slf, srv, \Delta_v)$ (cf. Table 3.8).

The modified responder process thus is

$$\begin{array}{rl}\mathrm{WMF}_{\mathtt{RESP}}(slf, srv, \Delta_v) := \ldots & \mathtt{In}\ \{\!|((t, oth), key)|\!\}_{\overline{k_{ss}}}\ \mathtt{when}\ t : \mathtt{TV} \wedge \\ & \exists t_s(t_s : \mathtt{TV} \wedge t_s@slf \wedge t + \Delta_v \leq t_s) \wedge \\ & oth : \mathtt{A} \wedge \\ & key : \mathtt{K} \wedge \\ & \varphi. \\ & \mathtt{Out}_{\mathtt{Eve}}\ t\ .1\end{array}$$

where '...' designates the part left unchanged by the modification. The specification process WMF_{spec} is then obtained from $\mathrm{WMF}_{\mathtt{RESP}}$ by stipulating that $\varphi := \exists t \exists t'(oth \circlearrowleft key.((slf, oth), (t, t')) \wedge \exists t_s(t_s@slf \wedge t \leq t_s \leq t')$, and the implementation process WMF_{impl} by stipulating (again) that $\varphi := \top$.

The "magic" in our specification process is the non-trivial and *epistemically non-local* guard of the input of the responder process. In the implementation, we only check the types of the atoms in the message and that the time stamp is recent. In the specification, we additionally check that the key has been created by the initiator for communication with the responder and that the locally-measured time is within the validity interval of the key, as expressed by the additional guard-conjunct. It is possible to have the simple time stamp check succeed but the "more obviously correct" validity check fail, namely with the set-up

$$\begin{array}{l}(\mathfrak{h}, \mathrm{WMF}(\mathtt{Alice}, \mathtt{Trent}, \mathtt{Bob}, x_{a1}, x_{t1}, x_{b1}, \Delta_v)\ |||\ \\ \quad \mathtt{Trent}.x_{t2}[\,\mathrm{WMF}_{\mathtt{SERV}}(\mathtt{Trent}, \Delta_v)\,]\ |||\ \\ \quad \mathtt{Trent}.x_{t3}[\,\mathrm{WMF}_{\mathtt{SERV}}(\mathtt{Trent}, \Delta_v)\,])\end{array}$$

from Section 3.4.4.

This set-up can, via process reduction, produce the family of traces that is generated by instantiating the parameters t_1, t_2, and t_3 of the history template

Table 4.7: History template for WMF

$$
\begin{aligned}
\mathfrak{h} := \mathfrak{h}_{init} \cdot \mathtt{N}(\mathtt{Alice}, x_{a1}, k_{(\mathtt{Alice},\mathtt{Bob})}, ((\mathtt{Alice}, \mathtt{Bob}), (-\infty, 2\Delta_v))) \cdot \\
\mathtt{O}(\mathtt{Alice}, x_{a1}, \mathtt{Alice}, \mathtt{Trent}) \cdot \\
\mathtt{O}(\mathtt{Alice}, x_{a1}, \{\!|((0, \mathtt{Bob}), k_{(\mathtt{Alice},\mathtt{Bob})})|\!\}_{k_{(\mathtt{Alice},\mathtt{Trent})}}, \mathtt{Trent}) \cdot \mathtt{S}(\mathtt{Eve}, \blacksquare, \Delta_v) \cdot \\
\mathtt{I}(\mathtt{Trent}, x_{t1}, \mathtt{Alice}) \cdot \mathtt{I}(\mathtt{Trent}, x_{t1}, \{\!|((0, \mathtt{Bob}), k_{(\mathtt{Alice},\mathtt{Bob})})|\!\}_{k_{(\mathtt{Alice},\mathtt{Trent})}}) \cdot \\
\mathtt{O}(\mathtt{Trent}, x_{t1}, \{\!|((t_1, \mathtt{Alice}), k_{(\mathtt{Alice},\mathtt{Bob})})|\!\}_{k_{(\mathtt{Bob},\mathtt{Trent})}}, \mathtt{Bob}) \cdot \mathtt{S}(\mathtt{Eve}, \blacksquare, 2\Delta_v) \cdot \\
\mathtt{I}(\mathtt{Trent}, x_{t2}, \mathtt{Bob}) \cdot \mathtt{I}(\mathtt{Trent}, x_{t2}, \{\!|((t_1, \mathtt{Alice}), k_{(\mathtt{Alice},\mathtt{Bob})})|\!\}_{k_{(\mathtt{Bob},\mathtt{Trent})}}) \cdot \\
\mathtt{O}(\mathtt{Trent}, x_{t2}, \{\!|((t_2, \mathtt{Bob}), k_{(\mathtt{Alice},\mathtt{Bob})})|\!\}_{k_{(\mathtt{Alice},\mathtt{Trent})}}, \mathtt{Alice}) \cdot \mathtt{S}(\mathtt{Eve}, \blacksquare, 3\Delta_v) \cdot \\
\mathtt{I}(\mathtt{Trent}, x_{t3}, \mathtt{Alice}) \cdot \mathtt{I}(\mathtt{Trent}, x_{t3}, \{\!|((t_2, \mathtt{Bob}), k_{(\mathtt{Alice},\mathtt{Bob})})|\!\}_{k_{(\mathtt{Alice},\mathtt{Trent})}}) \cdot \\
\mathtt{O}(\mathtt{Trent}, x_{t3}, \{\!|((t_3, \mathtt{Alice}), k_{(\mathtt{Alice},\mathtt{Bob})})|\!\}_{k_{(\mathtt{Bob},\mathtt{Trent})}}, \mathtt{Bob}) \cdot \\
\mathtt{I}(\mathtt{Bob}, x_{b1}, \{\!|((t_3, \mathtt{Alice}), k_{(\mathtt{Alice},\mathtt{Bob})})|\!\}_{k_{(\mathtt{Bob},\mathtt{Trent})}}) \cdot \mathtt{O}(\mathtt{Bob}, x_{b1}, t_3, \mathtt{Eve})
\end{aligned}
$$

shown in Table 4.7 — for clarity, without interception events (cf. Rule Out and In in Table 4.4). The only possible time values for the parameters t_1, t_2, and t_3 in this history template are those mentioned in the set-events of the adversary. For the implementation, we may have $t_1 = \Delta_v$, $t_2 = 2\Delta_v$, and $t_3 = 3\Delta_v$, where t_3 is observed by the adversary in clear in the last event. However, the specification cannot accept a stale key, i.e., a key older than $2\Delta_v$. Hence, the specification cannot generate a history that conforms to the above template, in particular a history such that the respondent outputs $3\Delta_v$ as her last action. Thus the implementation and the specification are not equivalent from the point of view of the adversary, so the specification is not met.

4.3 Denotational semantics

Our definition of the meaning of a cryptographic protocol is motivated by our definition of the meaning of a cryptographic message, which in turn is motivated by Abadi and Needham's Principle 1 for designing cryptographic protocols [AN96a]. The principle says:

> Every message should say what it means: the interpretation of the message should depend only on its contents. [...]

With a *clin d'œil* and thus for the nonce, one-eyed, we observe that the principle is both self-denying and not self-denying and thus paradoxical, and that this fact can be proven by applying the principle to itself. Here is an informal proof, by contradiction:

> Assume that the principle is not self-denying. Apply it to itself by particularising it with itself. (The principle is itself a message[3] and employs universal quantification over messages.) The message of the principle does not say

[3] Observe that the principle speaks about messages *tout court* rather than *cryptographic* messages. If the principle is to speak about cryptographic messages, then it must depend on a context (which [AN96a] of course constitutes) that properly frames it (which [AN96a] of course does). Yet, that very dependence the principle denies.

> what it means: the interpretation of the message does not depend only on its contents. (The principle does not say what message meaning means.) Hence, the principle is self-denying. Contradiction.
>
> Deduce that the principle is self-denying. Yet, by this very fact the principle is *not* self-denying. (Every message, as the example of the principle demonstrates, should really say what it means.) Conclude that the principle is paradoxical.

This recreational proof is paradoxical in the (double) sense that it demonstrates the paradoxical nature of Principle 1 and paradoxically the importance of the subject matter of the principle. That is, *explicitness* and *context-sensitivity* of meaning. Our task shall be to define, in a unique sense, the meaning[4] of a cryptographic message relative to an execution state and communicating agent.

Related work The most relevant work (though not addressing the problem of logical omniscience) related to ours is [PR03], where, according to the authors, the meaning of a message is given in terms of how it affects the knowledge of the agents involved in the communication. This idea is spiritually close to ours (and is given a very nice philosophical treatment by the authors), but has incarnated in a very different body (of knowledge, so to say) as will become clear in the sequel. The authors define the denotation of a message at a state and w.r.t. an agent as a so-called *view transformer*. And a view transformer is a set of ordered pairs (α, β) of propositions α and β such that "if knowledge of α is part of the view that [agent] i has of the global system state before the communication event, then knowledge of β is part of its view after the communication."

Another relevant work related to ours is [Gro92]. There, the so-called (1) *objective semantics* of a message is defined to be the set of all points, states in our terminology, where the message was sent; and (2) *KS-semantics* of a message is defined to be the set of pairs of a point and an agent such that that agent sends the message at that point. The problem with this approach is, according to the authors, that it is not yet suitable for cryptography.

The relevant commonality between [PR03], [Gro92], and our approach is *that* all approaches employ epistemic logic as a means to defining message meaning. The difference is *how* each approach does so, as we shall see now with the presentation of our approach.

4.3.1 Message meaning

Definition 13 (The meaning of a cryptographic message) *The (communicable) meaning of a cryptographic message $M \in \mathcal{M}$ w.r.t. an agent $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$ and a protocol state $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$ shall be the set of equivalence classes $[\phi]$ of those propositions $\phi \in \mathcal{F}$ whose truth a* would *know (resp., be able to prove) if a* knew *M in $\mathfrak{s}$. (Notice the conditional mode!) Formally,*

$$\begin{aligned} [\![M]\!]^{\mathfrak{s}}_{a} &:= \{\ [\phi] \mid \phi \in \mathcal{F} \text{ and } \mathfrak{s} \models a \,\mathsf{k}\, M \rhd \mathsf{K}_a(\phi)\ \}, \text{ and} \\ [\![M]\!]^{\mathfrak{s}}_{\mathsf{P}_a} &:= \{\ [\phi] \mid \phi \in \mathcal{F} \text{ and } \mathfrak{s} \models a \,\mathsf{k}\, M \rhd \mathsf{P}_a(\phi)\ \} \text{ respectively.} \end{aligned}$$

[4] in the sense of Frege's *sense* (a message makes to an agent) as opposed to *reference* (to a bit-string)

Message meaning is defined as a set of *equivalence classes* of propositions rather than as a set of propositions because we are concerned with just *what* a message means rather than with the manifold *how* it may mean it.

Message meaning is *communicable* when defined in terms of provability because provability requires the capability to produce an actual proof (i.e., a message of a certain cryptographic form), which, by definition, *is* communicable.

Theorem 3 *Message meaning is a distributive proper filter (and thus a proper sub-lattice and a topped $\bigcap$-structure) w.r.t. the Boolean lattice $\langle \mathcal{F}/_{\equiv}, \leq \rangle$, i.e., it is (1) a non-empty proper sub-set of $\mathcal{F}/_{\equiv}$, (2) closed under meet and partial ordering (and thus is directed), and (3) distributive. Formally, let*[5]

$$\overline{[\phi]} := [\neg\phi] \quad \textit{(complement)}$$

$$[\phi] \wedge [\phi'] := [\phi \wedge \phi'] \quad \textit{(meet)} \qquad [\phi] \vee [\phi'] := [\phi \vee \phi'] \quad \textit{(join)}$$

$$[\phi] \leq [\phi'] \quad \textit{:iff} \quad \phi \Rightarrow \phi' \textit{ and } \mathrm{keys}(\phi') \subseteq \mathrm{keys}(\phi) \quad \textit{(partial ordering)}$$

$$\phi \equiv \phi' \quad \textit{:iff} \quad [\phi] \leq [\phi'] \textit{ and } [\phi'] \leq [\phi] \quad \textit{(congruence w.r.t. meet and join)}$$

where $\mathrm{keys}(\phi)$ *designates the set of all key constants occurring in* ϕ*. Then,*

1. $\emptyset \neq [\![M]\!]^{\mathfrak{s}}_a \subset \mathcal{F}/_{\equiv}$
2. (a) *if* $[\phi], [\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$ *then* $[\phi] \wedge [\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$
   (b) *if* $[\phi] \in [\![M]\!]^{\mathfrak{s}}_a$ *and* $[\phi'] \in \mathcal{F}/_{\equiv}$ *and* $\phi \leq \phi'$ *then* $[\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$
3. *if* $[\phi] \vee [\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$ *and* $[\phi] \vee [\phi''] \in [\![M]\!]^{\mathfrak{s}}_a$ *then* $[\phi] \vee ([\phi'] \wedge [\phi'']) \in [\![M]\!]^{\mathfrak{s}}_a$.

Proof. see Appendix A

Filters represent deductively closed (cf. Condition 2.(b)), consistent (i.e., $[\bot] \notin [\![M]\!]^{\mathfrak{s}}_a$, otherwise $[\![M]\!]^{\mathfrak{s}}_a = \mathcal{F}/_{\equiv}$ by Condition 2.(b), which would violate Condition 1.), but possibly incomplete (unless they are maximal) theories.

Theorem 4 *Communicable message meaning is a distributive proper filter w.r.t. the Boolean lattice* $\langle \mathcal{F}/_{\equiv}, \leq \rangle$.

Proof. see Appendix A

Proposition 3 *Communicable message meaning is order-embedded strictly within message meaning. Formally,* $[\![M]\!]^{\mathfrak{s}}_{\mathsf{P}_a} \subset [\![M]\!]^{\mathfrak{s}}_a$ *and* $[\![M]\!]^{\mathfrak{s}}_{\mathsf{P}_a} \hookrightarrow [\![M]\!]^{\mathfrak{s}}_a$.

Proof. see Appendix A

Definition 14 (Context-sensitivity of message meaning) *A cryptographic message* $M \in \mathcal{M}$ *is* context-sensitive *:iff there are* $a, b \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$ *and* $\mathfrak{s}, \mathfrak{s}' \in \mathcal{H} \times \mathcal{P}$ *s.t.* $[\![M]\!]^{\mathfrak{s}}_a \neq [\![M]\!]^{\mathfrak{s}'}_b$. *A cryptographic message* $M \in \mathcal{M}$ *that is not context-sensitive is* context-free.

[5] this is a refinement of the classical Lindenbaum-Tarski-algebra construction: the partial ordering is *not* mere logical consequence ($\Rightarrow$); thus the resulting equivalence ($\equiv$) is finer than logical equivalence ($\Leftrightarrow$)

Note that our notion of context-sensitivity is *semantic* as opposed to the classical notion of formal language theory, which is syntactic.

Formalisation 1 (Abadi and Needham's Principle 1)

"Every message should say what it means: the interpretation of the message should depend only on its contents. [...]" [AN96a]

Every cryptographic message should be context-free *(cf. Definition 14).* ⌟

Examples of context-sensitive cryptographic messages abound. Perhaps the starkest example is the one of sending a bare nonce as the opening of a cryptographic protocol. See [BM03] for several *different*, and reportedly flawed protocols with such *identical* openings. Another way of creating context-sensitive cryptographic messages is the use of *indexicals* (i.e., phrases in natural language that refer to past or future messages, or to extra-protocol out-of-band communication such as personal contact and trusted couriers) in plaintexts. An advantage of our definition of the context-sensitivity of a cryptographic message is that the definition is, being equational, *indifferent to* the many ways of *how* context-sensitivity is created. It[6] only cares about *that* context-sensitivity is created.

Definition 15 (The information content of a cryptographic message) *The information content (in the sense of Kolmogorov-complexity [LV97]) of a cryptographic message $M \in \mathcal{M}$ to agent $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, written $\mathrm{K}_a(M)$, is defined to be the smallest[7] state $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$ s.t. $\mathfrak{s} \models a \,\mathtt{k}\, M$.*

Note that [PR03] also define the information content of a message, but their definition is, as opposed to ours, in the sense of Shannon.

4.3.2 Protocol meaning

Recall Slogan 8. In other words, cryptographic protocols aim at inducing propositional knowledge, i.e., knowledge of cryptographic states of affairs expressed as propositions, by means of individual knowledge, i.e., knowledge of messages (values). Values are only the means — not the ends — of cryptographic[8] computation.

Slogan 20 *In cryptography, individual knowledge is the key to propositional knowledge.*

Definition 16 (The meaning of a cryptographic protocol) *The meaning (or* denotational semantics*) $[\![\mathfrak{s}]\!]^*_a$ of a cryptographic protocol $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$ w.r.t. to an agent $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$ shall be the (directed) union of the meanings w.r.t. a of all those messages that a comes to know during protocol execution. Formally, let $T(\cdot) := \cdot \cup \{\bigwedge \mathrm{Min}(\cdot)\}$ designate a template for the meet-completion of $\cdot$.*

[6] "It", the footnote marker in this line, and "this" are three examples of indexicals.

[7] bear in mind that our protocol states are just finite strings of symbols, each string containing a process term (the program) as a substring

[8] and possibly of interactive computation [GSW06] in general

Then,

$$[\![\mathfrak{s}]\!]_a := T\left(\bigcup_{\substack{M \in \mathcal{M} \\ \mathfrak{s} \models a \,\mathsf{k}\, M}} [\![M]\!]^{\mathfrak{s}}_a\right) \qquad [\![\mathfrak{s}]\!]^n_a := T\left(\bigcup_{\substack{\mathfrak{s}' \in \mathcal{H} \times \mathcal{P} \\ \mathfrak{s} \longrightarrow^n \mathfrak{s}'}} [\![\mathfrak{s}']\!]_a\right) \qquad [\![\mathfrak{s}]\!]^*_a := T\left(\bigcup_{\substack{\mathfrak{s}' \in \mathcal{H} \times \mathcal{P} \\ \mathfrak{s} \longrightarrow^* \mathfrak{s}'}} [\![\mathfrak{s}']\!]_a\right)$$

$$[\![\mathfrak{s}]\!] := \bigwedge_{a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}} [\![\mathfrak{s}]\!]_a \qquad [\![\mathfrak{s}]\!]^n := \bigwedge_{a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}} [\![\mathfrak{s}]\!]^n_a \qquad [\![\mathfrak{s}]\!]^* := \bigwedge_{a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}} [\![\mathfrak{s}]\!]^*_a.$$

The meet-completion ensures that each collective meaning has again a least element, by taking the meet of the set of minimal elements in the union of individual meanings. (The *least* element in each individual meaning becomes a *minimal* element in the union of individual meanings.)

Note that our definition of the meaning (or denotational semantics) of a cryptographic protocol

1. is defined in terms of
   (a) the meaning of cryptographic messages
   (b) the *operational* semantics $\longrightarrow$ of that protocol, which is a very natural, but nevertheless, a novel idea. It is natural to define the *what* (the denotation) in terms of the *how* (the operations), rather than the other way round or defining them independently of each other.
2. has the advantage of being *syntax-independent*. The denotational semantics does not require inductive definition on the structure of cryptographic protocol terms $P \in \mathcal{P}$.

Theorem 5 $[\![\mathfrak{s}]\!]_a = [\![\mathfrak{s}']\!]_a$ *iff [for all* $\phi \in \mathcal{F}$, $\mathfrak{s} \models \mathsf{K}_a(\phi)$ *iff* $\mathfrak{s}' \models \mathsf{K}_a(\phi)$ *]*

Proof. see Appendix A

In other words, two protocol states (worlds) that make equal sense to a protocol agent are *epistemically indistinguishable* (w.r.t. to language $\mathcal{F}$) to that agent. In Wittgenstein's words: "The limits of my language mean the limits of my world." [Wit75, Paragraph 5.6].

Theorem 6

1. $[\![\mathfrak{s}]\!]^n_a \hookrightarrow [\![\mathfrak{s}]\!]^{n+1}_a$ *is order-continuous*[9]
2. $[\![\mathfrak{s}]\!]_a$, $[\![\mathfrak{s}]\!]^n_a$, *and* $[\![\mathfrak{s}]\!]^*_a$ *are topped algebraic* $\bigcap$*-structures (thus algebraic lattices, thus complete lattices, and thus complete partial orders [DP02])*
3. $[\![\mathfrak{s}]\!]$, $[\![\mathfrak{s}]\!]^n$, *and* $[\![\mathfrak{s}]\!]^*$ *are pre-CPOs*

Proof. see Appendix A

An algebraic $\bigcap$-structure $\mathfrak{S}$ can be presented as a *Scott information system* $\mathbf{IS}(\mathfrak{S})$ with "the idea of identifying an object with a set of propositions true of

[9] order-continuity guarantees the existence of fixpoints

it and adequate to define it. These propositions are to be thought of as tokens, each bearing a finite amount of information." [DP02]:

$$\mathbf{IS}(\mathfrak{S}) := \langle \bigcup \mathfrak{S}, \{\ \Gamma \mid \text{there is } S \in \mathfrak{S} \text{ s.t. } \Gamma \Subset S\ \}, \vdash \rangle$$

where $\Gamma \vdash \phi$:iff $\phi \in \bigcap \{\ S \mid S \in \mathfrak{S} \text{ and } \Gamma \Subset S\ \}$ and $\Subset$ designates finitary set-inclusion. In other words, protocol meaning induces a Scott information system for each protocol agent at each protocol state. And protocol execution induces the continuous (cf. Theorem 6.1) evolution of those information systems in time.

The definition of a denotational semantics for cryptographic protocols is not only a useful exercise of conceptual clarification, but is also useful for the actual engineering (verification, refinement) of safe cryptographic protocols:

Protocol invariant A formula $\phi \in \mathcal{F}$ is a *subjective* protocol invariant w.r.t. agent $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$ and initial protocol state $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$:iff for all $n \in \mathbb{N}$, $\phi \in [\![\mathfrak{s}]\!]^n_a$. A formula $\phi \in \mathcal{F}$ is a *universal* protocol invariant w.r.t. initial protocol state $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$:iff for all $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, ϕ is a subjective protocol invariant w.r.t. a and $\mathfrak{s}$.

Protocol safety A protocol state $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$ is *subjectively safe* to agent $a \in \boldsymbol{\mathcal{A}}$:iff the negation of every cryptographic state of affairs undesirable to a is a subjective protocol invariant w.r.t. a. A protocol state $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$ is *universally safe* :iff the negation of every cryptographic state of affairs undesirable to some $a \in \boldsymbol{\mathcal{A}}$ is a subjective protocol invariant w.r.t. a.

Protocol refinement A protocol state $\mathfrak{s}' \in \mathcal{H} \times \mathcal{P}$ *refines* protocol state $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$, written $\mathfrak{s} \leq \mathfrak{s}'$, :iff the meaning of $\mathfrak{s}$ is included in the meaning of $\mathfrak{s}'$. Formally,

$$\mathfrak{s} \leq \mathfrak{s}' \quad \text{:iff} \quad [\![\mathfrak{s}]\!]^* \subseteq [\![\mathfrak{s}']\!]^*.$$

It is well-known that refinement orderings on a set of specification processes on the one side and on a set of implementation processes on the other side induce a *Galois-connection* between the two sides of sets (cf. [DP02]).

Definition 17 (The information content of a cryptographic protocol) *The information content (in the sense of Kolmogorov-complexity [LV97]) of a protocol state $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$, containing the protocol(s), to agent $a \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, written $\mathrm{K}_a(\mathfrak{s})$, is defined to be the smallest message $M \in \mathcal{M}$ s.t. $[\![M]\!]^{\mathfrak{s}}_a = [\![\mathfrak{s}]\!]^*_a$.*

Chapter 5

Towards Probabilistic Polynomial-time Cryptography

5.1 Introduction

We sketch an Ockham's razor extension of core CPL (cf. Chapter 3) with a notion of probabilistic polynomial-time (PP) computation. We hope to intrigue the reader that adding a notion of PP-computation to a (property-based) formalism for cryptographic protocols can perhaps be simple and conceived through a refinement of the Dolev-Yao conception of cryptographic operators. The special-purpose machinery for PP (as for *real* time, cf. Section 3.4 and [Lam05]), need not be built from scratch nor be heavy-weight.

Related work We are aware of the following existing formalisms[1] for the specification and verification of cryptographic constructions.

- Property-based: [DMP03, DDMP05, DDM+05] in the tradition of Hoare logic, with satisfaction and deduction relations, and originally conceived for cryptographic *protocols*; and [IK06] a *higher*-order logic, *deduction*-based, and originally conceived for cryptographic *operators*
- Model-based: [MRST06] a process algebra for equivalence-based specification and verification.

Our approach In contrast, ppCPL is in the tradition of first-order[2], *temporal* — more precisely, poly-dimensional (i.e., norms, knowledge, space, qualitative and possibly quantitative time, cf. Section 3.4) mono-modal — logic, (for the moment still) satisfaction-based, and originally conceived for cryptographic *protocols* but here extended to encompass cryptographic *operators* to some extent.

[1] In our view, a formalism consists of exactly three components: a formal (e.g., programming or logical) language, a mathematical model (or interpretation structure), and a formal semantics (e.g., effect or truth) for the language in terms of the model.

[2] higher-order logics are too expressive at the cost of axiomatic incompleteness

Our general idea is to identify agents with feasible algorithms and, consequently, to *resource-bound* only the truth establishment of individual and propositional knowledge. That is, the machinery for probabilistic polynomial-time computation does not affect the whole logic (as opposed to [DDM+05], [IK06], where it does), but remains nicely confined to — and is observable only through the looking glass of — epistemic operators.

5.1.1 Symbolic logic

We qualify a logic as *symbolic*[3] when its language allows quantification over individuals that are represented as *syntactic terms* formed with term constructors (i.e., functional symbols). In this sense, CPL is symbolic; its language allows quantification over individuals, i.e., cryptographic messages, represented as message terms.

Core CPL (cf. Chapter 3) can further be qualified as *abstract* (in the sense of Dolev-Yao) because message terms are *not* interpreted as bit-strings. In contrast, ppCPL is *concrete* (in the sense of PP-computation) because message terms *are* interpreted as bit-strings. More precisely, in ppCPL message terms are denoted to probability distributions of bit-strings by interpreting logical constants as bit-strings and functional symbols as possibly probabilistic polynomial-time, i.e., feasible, algorithms on bit-strings (cf. Table 5.1[4]).

Table 5.1: Syntactic representation of individual concepts

Concept	Representation
formal variable	possibly primed letters 'v'
ad-hoc variable	lowercase roman letter except 'v's (e.g., 'm' for messages)
meta-variable	roman letter except 'v's (e.g., 'M' for messages)
abstract value	atomic or compound message term
concrete value	bit-string

Furthermore, core CPL can be qualified as *positive about truth and knowledge* because false positives, i.e., false statements wrongly established as true, and false belief respectively, are impossible. In contrast, ppCPL is *probabilistic about truth and knowledge* because false positives are, w.r.t. truth, possible (though only) with negligible probability, and w.r.t. knowledge, (im)probable with variable degrees of support.

Finally, we highlight that in the language of ppCPL probability is *implicit* except for belief where it parametrises the (new) doxastic modality. In particular, there is no likelihood operator (cf. [Hal03]) in ppCPL. The reason is that in modern cryptography truth must be established with overwhelming probability, whereas belief of a human being may be established with possibly non-overwhelming degrees of support. Philosophically speaking (cf. Carnap[5]), prob-

[3] traditional usage of the term is philosophical and not standardised

[4] "[T]o be is to be the value of a variable. More precisely, what one takes there to be are what one admits as values of one's bound variables." [Gib04, Page 111]

[5] Incidentally, Carnap "wrote a [first] thesis setting out an axiomatic theory of space and time. The physics department said it was too philosophical, and [...] the philosophy department said it was pure physics." (cf. http://en.wikipedia.org/wiki/Rudolf_Carnap)...

ability can be an (epistemological) measure of our (subjective) belief of states of affairs as well as an (ontological) measure of their (objective) possibility. The refinement of the doxastic modality with probability allows the expression of *degrees of certitude* that an agent may experience w.r.t. her apprehension of cryptographic states of affairs.

5.1.2 Probability theory

A fundamental concept of probability theory is the one of a *probabilistic experiment* characterised by the indeterminacy of its outcome, i.e., its *entropy*. The fact that such experiments are probabilistic implies that they are *stateful*, i.e., they represent states (with an inherent potential future) of the considered model, and their execution means probabilistic state transition. *A priori*, an experimenter, i.e., a human being about to experience the considered experiment, typically has an *uncertainty* about the *actual* outcome of the experiment, and makes a *hypothesis* about its *expected* outcome. A *posteriori*, the experimenter typically makes an *epistemic error* about the actual outcome of the experiment w.r.t. the hypothesis made a priori.

In ppCPL, exactly two kinds of probabilistic experiments are relevant: *process reduction*, i.e., protocol execution, (cf. Table 5.2) and *message denotation*, i.e., message evaluation, (cf. Table 5.3).

Table 5.2: Probabilistic process reduction

Probability theory	CPL
sample space	$\mathcal{H} \times \mathcal{P}$
variable (experiment) X	protocol state $(\mathfrak{h}, P)$
atomic event	transition $(\mathfrak{h}, P) \longrightarrow (\mathfrak{h}', P')$
possible value (outcome) of X	$(\mathfrak{h}', P')$ s.t. $(\mathfrak{h}, P) \longrightarrow (\mathfrak{h}', P')$
probability distribution $\mathbf{P}(X)$	$\{ ((\mathfrak{h}', P'), p) \mid (\mathfrak{h}, P) \longrightarrow (\mathfrak{h}', P')$ and $p \in]0, 1] \}$ such that $\Sigma_{p \in \mathbf{P}((\mathfrak{h}, P))} p = 1$
hypothesis h about outcome	proposition ϕ
hypothesis H about outcome	$\{ (\mathfrak{h}', P') \mid (\mathfrak{h}, P) \longrightarrow (\mathfrak{h}', P') \models \phi \}$

Note that interleaving concurrency implies that atomic events are mutually exclusive and independent within each branching. Applying the principle of indifference, we fix $\mathbf{P}(\mathfrak{s})$ to the uniform distribution for all $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$.

In Table 5.3, $[\![\cdot]\!]$ designates the function of message denotation.

5.1.3 Probabilistic polynomial-time cryptography

The distinguishing features of probabilistic polynomial-time cryptography are that (1) key and signature generation, and encryption are probabilistic (or randomised); (2) the execution time of the operations under Item 1 and decryption are polynomially bounded in a *security parameter* (the length of the key) used for key generation; (3) adversaries are PP Turing machines with oracle access; and (4) oracles are PP Turing machines.

Table 5.3: Probabilistic message denotation

Probability theory	CPL
sample space	$\{0,1\}^*$
variable (experiment) X	message term $M \in \mathcal{M}$
atomic event	$[\![M]\!] = s$ where $[\![\cdot]\!] \subseteq \mathcal{M} \times \{0,1\}^*$
possible value (outcome) of X	$s \in \{0,1\}^*$ s.t. $s = [\![M]\!]$
probability distribution $\mathbf{P}(X)$	$\{\ (s,p) \mid s = [\![M]\!]$ and $p \in]0,1]\ \}$ such that $\Sigma_{p \in \mathbf{P}(M)} p = 1$
hypothesis h about outcome	$[\![M]\!] = s$
hypothesis H about outcome	$\{\ s \mid s = [\![M]\!]\ \}$

5.2 ppCPL: an extension of CPL with probabilistic polynomial-time

This section describes the extension of CPL to ppCPL. The extension depends on the core described in Chapter 3 (the reader is urged to consult it).

5.2.1 Syntax

The syntactic novelties are the following:

1. *logical constants* (atomic message terms): refinement of the abstract message $\blacksquare_l$ with a length indication $l \in \mathbb{N}$; addition of bit-strings $s ::= \mathtt{0} \mid \mathtt{1} \mid s \bullet s$ and of probability values $q \in \mathcal{PV} := [0,1] \cap \mathbb{Q}$ with the associated sort `PV`

2. *functional symbols*: refinement of hashes $\lceil M \rceil^{HA}$, symmetric $[M]^{SEA}_{M'}$ and asymmetric $[M]^{AEA}_{p^+}$ encryptions, and signatures $]M[^{SA}_{p}$ with a parameter $HA \in \{\mathtt{SHA1}, \mathtt{MD5}, \ldots\}$, $SEA \in \{\mathtt{DES}(MOO), \mathtt{AES}(MOO), \ldots\}$, $AEA \in \{\mathtt{RSA}, \mathtt{Elgamal}, \ldots\}$, resp. $SA \in \{\mathtt{RSA}, \mathtt{Elgamal}, \ldots\}$ for the name of the employed algorithm. $MOO \in \{\mathtt{ECB}, \mathtt{CBC}, \mathtt{CFB}, \mathtt{OFB}, \ldots\}$ is a parameter for the name of the employed mode of operation of a block cipher.

3. *relational symbols*:

   (a) refinement of the predicate $a \overset{\iota}{\underset{NGA}{\circlearrowleft}} n.o$ for new-name generation with a security parameter $\iota \in \mathbb{N}$, and a parameter $NGA ::= SEA \mid AEA \mid SA$ for the name of the employed generation algorithm

   (b) addition of a binary relational symbol $\leq$ for the comparison of probability values (actually the same as for the comparison of time values, cf. Section 3.4)

4. *logical operators*: addition of a modality B^q_a for *belief with error control* q, where q is the probability for agent a not to err in her apprehension of the truth of the considered proposition (say ϕ), written $\mathsf{B}^q_a(\phi)$

5.2.2 Semantics

The principal semantic novelties are the following:

1. (backwards-compatible) refinement of temporal accessibility *simpliciter* to *PP* temporal accessibility: addition of *denotation events* $\mathtt{D}(a, M, s, ALGO)$ stating that agent a denoted the message term M to the string $s \in \{\mathtt{0},\mathtt{1}\}^*$ by application of the algorithm $ALGO ::= NGA \mid \blacksquare$, where $ALGO \in NGA$ if M is a name and $ALGO = \blacksquare$ otherwise

2. refinement of individual knowledge *simpliciter* to *PP* individual knowledge (cf. Table 5.4) relying on (stateful) PP message denotation:

$$[\![M]\!]^{\mathfrak{h}}_a := \begin{cases} \text{choose } s \text{ s.t. } \mathtt{D}(a, n, s, NGA) \in \dot{\mathfrak{h}} \text{ or else } n & \text{if } M = n, \text{ and} \\ \text{choose } s \text{ s.t. there is } p \text{ s.t. } (s,p) \in \mathbf{P}(M') & \text{otherwise.} \end{cases}$$

   where $M' := \bigcup_{n \in \mathrm{names}(M)} \left\{ {}^{[\![n]\!]^{\mathfrak{h}}_a}/_{n} \right\} M$ designates the message that results from the substitution of all names n in M with the corresponding denotations $[\![n]\!]^{\mathfrak{h}}_a$. (The act of choosing s could be made strictly formal with Hilbert's choice operator [Sla06].)

3. let

$$\begin{aligned} \mathcal{K}_a(\mathfrak{p}, i) &:= \{\ \mathfrak{s} \mid \mathfrak{p}@0 \longrightarrow^* \mathfrak{s} \text{ and } \mathfrak{s} \approx_a \mathfrak{p}@i\ \} \\ \mathcal{B}_a(\mathfrak{p}, i) &:= \{\ \mathfrak{s} \mid \mathfrak{p}@0 \longrightarrow^* \mathfrak{s} \text{ and } \mathfrak{s} \approx_a \mathfrak{p}@i \text{ and} \\ &\qquad \text{there is a polynomial } p : \mathbb{N} \to \mathbb{N} \text{ s.t. } |\mathfrak{s}| \leq p(|\mathfrak{p}@i|)\ \} \\ |(\mathfrak{h}, P)| &:= |\dot{\mathfrak{h}}| \end{aligned}$$

   (a) refinement of propositional knowledge *simpliciter*

$$\begin{aligned} [\![\mathsf{K}_a(\phi)]\!]^i_{\mathfrak{p}} := (&\text{for all } \mathfrak{s}, \text{ if } \mathfrak{s} \in \mathcal{K}_a(\mathfrak{p}, i) \text{ then } \mathfrak{s}' \models_{\mathcal{E}'} \phi',\ \mathcal{E}'_{(\mathfrak{s},\phi)}) \\ &\text{where } (\mathfrak{s}', \phi') := \begin{cases} (\mathfrak{s}, \phi) & \text{if } \mathfrak{s} = \mathfrak{p}@i, \text{ and} \\ (\langle\!|\, \mathfrak{s} \,|\!\rangle^{\mathfrak{p}@i}_a, \langle\!|\, \phi \,|\!\rangle^{\mathfrak{p}@i}_a) & \text{otherwise.} \end{cases} \end{aligned}$$

   to *PP* propositional knowledge

$$\begin{aligned} [\![\mathsf{K}_a(\phi)]\!]^i_{\mathfrak{p}} := (&\text{for all } \mathfrak{s}, \text{ if } \ \mathfrak{s} \in \mathcal{K}_a(\mathfrak{p}, i) \\ &\qquad \text{then } \ \mathfrak{s}' \models_{\mathcal{E}'} \phi' \text{ and there is a polynomial} \\ &\qquad\quad p : \mathbb{N} \to \mathbb{N} \text{ s.t. } |\mathfrak{s}| \leq p(|\mathfrak{p}@i|),\ \mathcal{E}'_{(\mathfrak{s},\phi)}) \\ &\text{where } (\mathfrak{s}', \phi') := \begin{cases} (\mathfrak{s}, \phi) & \text{if } \mathfrak{s} = \mathfrak{p}@i, \text{ and} \\ (\langle\!|\, \mathfrak{s} \,|\!\rangle^{\mathfrak{p}@i}_a, \langle\!|\, \phi \,|\!\rangle^{\mathfrak{p}@i}_a) & \text{otherwise.} \end{cases} \end{aligned}$$

   (b) addition of *believe with error control* $q \in \mathcal{PV}$

$$\begin{aligned} [\![\mathsf{B}^q_a(\phi)]\!]^i_{\mathfrak{p}} := (&q = \tfrac{|\mathcal{B}_a(\mathfrak{p},i)|}{|\mathcal{K}_a(\mathfrak{p},i)|} \text{ and} \\ &\text{for all } \mathfrak{s}, \text{ if } \mathfrak{s} \in \mathcal{B}_a(\mathfrak{p}, i) \text{ then } \mathfrak{s}' \models_{\mathcal{E}'} \phi',\ \mathcal{E}'_{(\mathfrak{s},\phi)}) \\ &\text{where } (\mathfrak{s}', \phi') := \begin{cases} (\mathfrak{s}, \phi) & \text{if } \mathfrak{s} = \mathfrak{p}@i, \text{ and} \\ (\langle\!|\, \mathfrak{s} \,|\!\rangle^{\mathfrak{p}@i}_a, \langle\!|\, \phi \,|\!\rangle^{\mathfrak{p}@i}_a) & \text{otherwise.} \end{cases} \end{aligned}$$

Table 5.4: PP derivation of individual knowledge

Random coin tossing

$$\frac{}{\mathfrak{h} \vdash_a^{(\emptyset,1)} \mathtt{0}} \qquad \frac{}{\mathfrak{h} \vdash_a^{(\emptyset,1)} \mathtt{1}}$$

Input data extraction

$$\frac{}{\mathfrak{h} \cdot \varepsilon(a,\overline{M}) \vdash_a^{(\{\varepsilon(a,\overline{M})\},0)} (a,\overline{M})} \qquad \frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M}{\mathfrak{h} \cdot \varepsilon \vdash_a^{(\mathcal{E},r)} M}$$

Data synthesis ***Data analysis***

$$\frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M \quad \mathfrak{h} \vdash_a^{(\mathcal{E}',r')} M'}{\mathfrak{h} \vdash_a^{(\mathcal{E}\cup\mathcal{E}',r+r')} (M,M')} \qquad \frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} (M,M')}{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M} \qquad \frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} (M,M')}{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M'}$$

$$\frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} p}{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} p^+} \qquad \frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M}{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} \lceil M \rceil}$$

$$\frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M \quad \mathfrak{h} \vdash_a^{(\mathcal{E}',r')} M'}{\mathfrak{h} \vdash_a^{(\mathcal{E}\cup\mathcal{E}',r+r')} [M]_{M'}} \qquad \frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} [M]_{M'} \quad \mathfrak{h} \vdash_a^{(\mathcal{E}',r')} M'}{\mathfrak{h} \vdash_a^{(\mathcal{E}\cup\mathcal{E}',r+r')} M}$$

$$\frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M \quad \mathfrak{h} \vdash_a^{(\mathcal{E}',r')} p^+}{\mathfrak{h} \vdash_a^{(\mathcal{E}\cup\mathcal{E}',r+r')} [M]_{p^+}} \qquad \frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} [M]_{p^+} \quad \mathfrak{h} \vdash_a^{(\mathcal{E}',r')} p}{\mathfrak{h} \vdash_a^{(\mathcal{E}\cup\mathcal{E}',r+r')} M}$$

$$\frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M \quad \mathfrak{h} \vdash_a^{(\mathcal{E}',r')} p}{\mathfrak{h} \vdash_a^{(\mathcal{E}\cup\mathcal{E}',r+r')}]M[_p} \qquad \frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)}]M[_p \quad \mathfrak{h} \vdash_a^{(\mathcal{E}',r')} p^+}{\mathfrak{h} \vdash_a^{(\mathcal{E}\cup\mathcal{E}',r+r')} M}$$

Data concretisation ***Data abstraction***

$$\frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M}{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} s} \; s = [\![M]\!]_a^{\mathfrak{h}} \qquad \frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} s}{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M} \; [\![M]\!]_a^{\mathfrak{h}} = s$$

PP-abstraction

$$\frac{\mathfrak{h} \vdash_a^{(\mathcal{E},r)} M}{\mathfrak{h} \vdash_a^{\mathcal{E}} M} \text{ there is a polynomial } p : \mathbb{N} \to \mathbb{N} \text{ s.t. } r \leq p(\Sigma_{\varepsilon(a,\overline{M})\in\mathcal{E}} |[\![(a,\overline{M})]\!]_a^{\mathfrak{h}}|)$$

4. redefinition of the state of violation with the desired kind(s) of breaks (i.e., successful attacks) of cryptographic schemes as formalised in Section 5.3.2.

The polynomial bound on the computational complexity of individual resp. propositional knowledge (cf. Item 2 resp. 3) induces a corresponding bound on the complexity of cryptographic proofs resp. provability (cf. Section 3.3.4).

Remark 6 *In analogy to non-standard analysis, it could be worthwhile to investigate the introduction of* infinitesimals *for negligible probabilities.*

5.3 Application: formalisation case studies

> *Definitions can be worthwhile even in the absence of theorems and proofs.*
>
> **Phillip Rogaway**
> (cf. [Rog04])

We illustrate the expressiveness of ppCPL on tentative formalisation case studies of fundamental and applied concepts. *Fundamental concepts*: (1) one-way function, (2) hard-core predicate, (3) computational indistinguishability, (4) (n-party) interactive proof, and (5) (n-prover) zero-knowledge. *Applied concepts*: (1) security of encryption schemes, (2) unforgeability of signature schemes, (3) attacks on encryption schemes, (4) attacks on signature schemes, and (5) breaks of signature schemes.

Note that in core CPL we focused on cryptographic *protocols* and their *requirements*, but here (in ppCPL) we focus on cryptographic *operators* and their *attacks* and *breaks*. Naturally, properties of good operators take the form of tautologies, i.e., propositions that hold in any (protocol) model (cf. Table 5.5).

Table 5.5: Expressing properties of protocols, operators, and messages

Subject	Expression of a property as a	Linking concept	Style of expression
Protocol	proposition ϕ in an assertion $(\mathfrak{h}, P) \models \phi$ (true statement)	satisfiability	endogenous (i.e., point-free/intensional w.r.t. *protocols*)
Operator	proposition ϕ in an assertion $\models \phi$ (tautology)	validity	point-wise/extensional w.r.t. *messages* (quantification!)
Message	unary predicate $\phi(\cdot)$	substitution ($\phi(M)$)	

Slogan 21 *Predicates speak of individuals (e.g., messages). Propositions talk about models (e.g., protocols). Both express purported facts.*

Our formalisations illustrate the dramatic expressive power that results from the combined use of epistemic and spatial operators.

5.3.1 Fundamental concepts

This section is in the spirit of [Gol01].

Definition 18 (Random propositional guessing)

$$\mathsf{RG}_a(\phi) := \neg\exists(q : \mathtt{PV})(\frac{1}{2} < q \wedge \mathsf{B}^q_a(\phi))$$

Definition 19 (Hard proposition) *A proposition ϕ is* hard *:iff in any model, any agent a can only guess the truth of ϕ. That is, $\mathsf{RG}_a(\phi)$ is a tautology.*

$$\models \mathsf{RG}_a(\phi)$$

We generalise the concept of a hard proposition (a closed formula) to the concept of a hard predicate (an open formula).

Definition 20 (Hard predicate) *An n-ary predicate $\phi(M_1, \ldots, M_n)$ is* hard *on $M_1, \ldots, M_n$ satisfying the $(n+1$-ary$)$ predicate $\varphi(M_1, \ldots, M_n, a)$:iff in any model, if $\varphi(M_1, \ldots, M_n, a)$ then any a can only randomly guess the truth of $\phi(M_1, \ldots, M_n)$.*

$$\models \varphi(M_1, \ldots, M_n, a) \rightarrow \mathsf{RG}_a(\phi(M_1, \ldots, M_n))$$

Formalisation 2 (One-way function)

"[...] a function that is easy to compute but hard to invert." [Gol01, Page 32]

Ease of computation $\models a \,\mathsf{k}\, M \rightarrow a \,\mathsf{k}\, f(M)$

Hardness of invertibility $f^{-1}(M') = M$ *is hard on* M *and* M' *satisfying* $f(M) = M' \wedge \neg\, a \,\mathsf{k}\, M$.

⌟

Notice that the standard definition of one-way functions only requires the operator f to be computable in *deterministic* polynomial-time, whereas the satisfiability of $a \,\mathsf{k}\, f(M)$ may be computable only in *probabilistic* polynomial-time (cf. Table 5.4). We could easily provide a deterministic variant of k by simply disallowing random coin tossing. Further, observe that our definition implies that cryptographic operators are common knowledge among agents. In order to express (individual) knowledge of operators, we would need quantification over functional symbols, which would make our logic higher-order.

Formalisation 3 (Hard-core predicate)

"[...] a polynomial-time *predicate b is called a hard-core of a function f if every efficient algorithm, given $f(x)$, can guess $b(x)$ with success probability that is only negligibly better than one-half." [Gol01, Page 64]*

Let the satisfiability of $\phi(M)$ be computable in polynomial-time. Then $\phi(M)$ is a hard-core *of a function f :iff $\phi(M)$ is hard on M satisfying $a \,\mathsf{k}\, f(M) \wedge \neg\, a \,\mathsf{k}\, M$.*
⌟

Notice that we identify agents with feasible algorithms!

Formalisation 4 (Computational indistinguishability)

"Objects are considered to be computationally equivalent if they cannot be differentiated by any efficient procedure." [Gol01, Page 103]

M and M' are computationally indistinguishable *:iff $\neg(M = M')$ is hard on M and M' satisfying $\neg(M = M')$.* ⌟

Definition 21 (Cryptographic evidence and proof)

$$
\begin{array}{rcl}
M \text{ necessaryEvidenceFor } \phi & := & \forall a(\mathsf{K}_a(\phi) \rhd (\mathsf{K}_a(\phi) \supseteq a \mathsf{\ k\ } M)) \\
M \text{ sufficientEvidenceFor } \phi & := & \forall a(\mathsf{K}_a(\phi) \rhd (a \mathsf{\ k\ } M \supseteq \mathsf{K}_a(\phi))) \\
M \text{ strictEvidenceFor } \phi & := & M \text{ necessaryEvidenceFor } \phi \wedge \\
 & & M \text{ sufficientEvidenceFor } \phi \\
M \text{ evidenceFor } \phi & := & M \text{ necessaryEvidenceFor } \phi \vee \\
 & & M \text{ sufficientEvidenceFor } \phi \\
M \text{ necessaryProofFor } \phi & := & \forall a(a \mathsf{\ k\ } M \rhd (\mathsf{K}_a(\phi) \supseteq a \mathsf{\ k\ } M)) \\
M \text{ sufficientProofFor } \phi & := & \forall a(a \mathsf{\ k\ } M \rhd (a \mathsf{\ k\ } M \supseteq \mathsf{K}_a(\phi))) \\
M \text{ strictProofFor } \phi & := & M \text{ necessaryProofFor } \phi \wedge \\
 & & M \text{ sufficientProofFor } \phi \\
M \text{ proofFor } \phi & := & M \text{ necessaryProofFor } \phi \vee \\
 & & M \text{ sufficientProofFor } \phi
\end{array}
$$

Observe that these concepts of cryptographic evidence and proof are refinements of the respective (basic) concepts defined in Section 3.3.4.

Conjecture 1

1. $\models M$ necessaryProofFor $\phi \rightarrow M$ necessaryEvidenceFor ϕ
2. $\models M$ sufficientProofFor $\phi \rightarrow M$ sufficientEvidenceFor ϕ
3. $\models M$ strictProofFor $\phi \leftrightarrow M$ strictEvidenceFor ϕ

Formalisation 5 (2-party interactive proof)

"A 2-party interactive proof *(or* 2-party computation *or* 2-party protocol*) M between a verifier a and prover b (initiated by a) for a proposition ϕ (protocol goal) is a (possibly minimal) finite chain $M = (M_0, \ldots, M_n)$ of messages s.t. (1) M_n is a proof of ϕ for a, and (2) for all consecutive pairs (M_i, M_j) in M, M_j derives from M_i due to communication between a and b." [author's formulation]*

$$
\begin{array}{rcl}
\overline{M} & ::= & (M, \blacksquare) \mid (M, \overline{M}) \\
I & ::= & \blacksquare \mid \overline{M}
\end{array}
$$

$$
\begin{array}{rcl}
M \text{ iProofFor}_{(a,b)}\ \phi & := & M \text{ iProofFor}^{a}_{(a,b)}\ \phi \\
(M, \blacksquare) \text{ iProofFor}^{c}_{(a,b)}\ \phi & := & c \mathsf{\ k\ } M \wedge M \text{ proofFor } \phi \\
(M, (M', I)) \text{ iProofFor}^{c}_{(a,b)}\ \phi & := & M' \sqsupseteq_{(a,b)} M \wedge (M', I) \text{ iProofFor}^{c}_{(b,a)}\ \phi
\end{array}
$$

⌟

Definition 22 (Interactive provability)

$$\mathsf{IP}_{(a,b)}(\phi) := \exists m(m \text{ iProofFor}_{(a,b)}\ \phi)$$

Our (macro-defined) operator for interactive provability is a tentative generalisation to the interactive setting of our macro-definition of Gödel's provability modality (cf. Section 3.3.5).

Proposition 4 $\models \mathsf{IP}_{(a,b)}(\phi) \rightarrow \mathsf{P}_a(\phi)$

Conjecture 2 (Characteristics of interactive proofs) *For "certain" ϕ,*

Soundness $\models \neg\phi \rightarrow \neg\mathsf{IP}_{(a,b)}(\phi)$

Completeness $\models \phi \rightarrow \mathsf{IP}_{(a,b)}(\phi)$

Formalisation 6 (Proof of knowledge)

"[...] [interactive] proofs in which the prover [b] asserts "knowledge" of some object [...] and not merely its existence [...]" [Gol01, Page 262]

$$\mathsf{IP}_{(a,b)}(b \;\mathsf{k}\; M)$$

⌟

Formalisation 7 (Zero-Knowledge)

"Zero-knowledge proofs are defined as those [interactive] proofs that convey no additional knowledge other than the correctness of the proposition [ϕ] in question." [GMR89]

$$\begin{aligned}\mathsf{ZK}_{(a,b)}(\phi) := \mathsf{IP}_{(a,b)}(&\mathsf{K}_a(\exists m'(\mathsf{K}_b(m' \text{ proofFor } \phi))) \wedge \\ &\neg\exists m''(\mathsf{K}_a(\mathsf{K}_b(m'' \text{ evidenceFor } \phi))))\end{aligned}$$

⌟

Spelled out, a (the verifier) knows through interaction with b (the prover) that b knows a proof (m') for the proposition ϕ, however a does not know that proof nor any evidence (m'') that could corroborate the truth of ϕ. (Observe the importance of the scope of the existential quantifiers.) Philosophically speaking, a has *pure propositional* knowledge of ϕ, i.e., a has *zero individual* (and thus *zero intuitionistic*—no witness!) knowledge *relevant* to the truth of ϕ. In Goldreich's words, it is "as if [the verifier] was told by a trusted party that the assertion holds" [Gol05, Page 39].

Standard zero-knowledge, i.e., zero-knowledge w.r.t. a malicious verifier is an instance of the above scheme where $a = \texttt{Eve}$. Zero-knowledge w.r.t. an honest verifier is definable as $\mathsf{ZK}_{(a,b)}(\phi) \wedge \mathsf{honest}(a)$.

Conjecture 3 *"[A]nything that is feasibly computable from a zero-knowledge proof is also feasibly computable from the (valid) assertion itself." [Gol05, Page 39]*

$$\models \phi \rightarrow ((\mathsf{K}_a(\varphi) \trianglerighteq \mathsf{ZK}_{(a,b)}(\phi)) \rightarrow (\mathsf{K}_a(\varphi) \trianglerighteq \phi))$$

This is a logical formulation of an instance of the *simulation paradigm* [GM84].

Formalisation 8 (n-party interactive proof)

"An n-party interactive proof *(or n-party* computation *or n-party* protocol*) M between agents $\{a_0, a_1, \ldots, a_{n-1}\}$ (initiated by a_0) for a proposition ϕ (protocol goal) is a (possibly minimal) finite chain $M = ((a_0, M_0), \ldots, (a_l, M_m))$ s.t. (1) M_m is a proof of ϕ for a_0, and (2) for all consecutive pairs $((a_i, M_i), (a_j, M_j))$ in M, M_j derives from M_i due to communication between a_i and a_j." [author's formulation]*

$$A ::= (a, \blacksquare) \mid (b, A)$$

$$\begin{aligned}
M\ \mathsf{iProofFor}_{(a,A)}\ \phi &:= M\ \mathsf{iProofFor}^{a}_{(a,A)}\ \phi \\
(M, \blacksquare)\ \mathsf{iProofFor}^{c}_{(a,\blacksquare)}\ \phi &:= c\ \mathsf{k}\ M \wedge M\ \mathsf{proofFor}\ \phi \\
(M, (M', I))\ \mathsf{iProofFor}^{c}_{(a,(b,A))}\ \phi &:= M' \supseteq_{(a,b)} M \wedge (M', I)\ \mathsf{iProofFor}^{c}_{(b,A)}\ \phi
\end{aligned}$$

⌟

Definition 23 (Quotient proof)

$$\begin{aligned}
M \mid_a M' &:= \neg(a\ \mathsf{k}\ M \supseteq a\ \mathsf{k}\ M') \wedge \\
&\quad\ \neg(a\ \mathsf{k}\ M' \supseteq a\ \mathsf{k}\ M) \\
(M, M')\ \mathsf{disjointEvidenceFor}\ \phi &:= \forall a(\mathsf{K}_a(\phi) \rhd (\mathsf{K}_a(\phi) \supseteq a\ \mathsf{k}\ (M, M') \wedge \\
&\qquad\qquad M \mid_a M')) \\
(M, \blacksquare)\ \mathsf{mutuallyDisjointEvidenceFor}\ \phi &:= M\ \mathsf{evidenceFor}\ \phi \\
(M, \overline{M})\ \mathsf{mutuallyDisjointEvidenceFor}\ \phi &:= (M, \overline{M})\ \mathsf{disjointEvidenceFor}\ \phi \wedge \\
&\quad\ \overline{M}\ \mathsf{mutuallyDisjointEvidenceFor}\ \phi \\
M\ \mathsf{quotientProofFor}\ \phi &:= M\ \mathsf{proofFor}\ \phi \wedge \\
&\quad\ M\ \mathsf{mutuallyDisjointEvidenceFor}\ \phi
\end{aligned}$$

We pronounce $M \mid_a M'$ as "M is (epistemically) independent from M' w.r.t. to a's knowledge". Quotient proofs could also be called *compositional* proofs.

Definition 24 (Multi-prover Zero-Knowledge)

$$\begin{aligned}
\mathsf{ZK}_{(a,A)}(\phi) := \mathsf{IP}_{(a,A)}(&\mathsf{K}_a(\exists m'(m'\ \mathsf{quotientProofFor}\ \phi \wedge A\ \mathsf{k}\ m')) \wedge \\
&\neg\exists m'(\mathsf{K}_a(m'\ \mathsf{evidenceFor}\ \phi \wedge A\ \mathsf{k}\ m')))
\end{aligned}$$

where

$$\begin{aligned}
(a, \blacksquare)\ \mathsf{k}\ (M, \blacksquare) &:= a\ \mathsf{k}\ M \\
(b, A)\ \mathsf{k}\ (M, \overline{M}) &:= b\ \mathsf{k}\ M \wedge A\ \mathsf{k}\ \overline{M}
\end{aligned}$$

Observe again the importance of the scope of the existential quantifiers.

Formalisation 9 (Oblivious Transfer)

"[...] we can view this protocol as one in which Alice sends a [confidential] letter to Bob, which arrives exactly half the time." [Kil88] (cf. [Rab81] for the original reference of the idea)

$$\boxplus\forall m(a\ \mathsf{k}\ m \wedge \forall(c : \mathtt{A})(c\ \mathsf{k}\ m \rightarrow (c = a \vee c = b)) \wedge \mathsf{RG}_b(b\ \mathsf{k}\ m))$$

⌟

5.3.2 Applied concepts

This section is in the spirit of [Gol04] and [MvOV96]. Note that for clarity, names of cryptographic algorithms are omitted in message terms in the sequel.

Definition 25 (Security of encryption schemes)

1. *"Standard* security*: the infeasibility of* obtaining information regarding the plaintext*" [Gol04, Page 470]*

   Semantic security *"[...] given any a priori information about the plaintext, it is infeasible to obtain any (new) information about the plaintext from the ciphertext (beyond what is feasible to obtain from the a priori information on the plaintext)." [Gol04, Page 378]*

   $$\models \underbrace{(a \mathsf{\ k\ } C \wedge \mathsf{K}_a(\phi(M)))}_{\textit{a priori information}} \rightarrow (\underbrace{(\mathsf{K}_a(\varphi(M)) \supseteq a \mathsf{\ k\ } C)}_{\textit{obtaining information}} \rightarrow \underbrace{(\phi(M) \supseteq \varphi(M))}_{\textit{no news}})$$

   where $C ::= [M]_k \mid [M]_{p^+}$
   This is again a logical formulation of an instance of the simulation paradigm *[GM84]. Observe the similarity with the previous instance.*

   Indistinguishability of encryptions

   - $[M]_k$ *(or* $[M]_{p^+}$*) and* $[M']_k$ *(or* $[M']_{p^+}$*) are computationally indistinguishable (in the sense of our formalisation)*
   - *there is* $l \in \mathbb{N}$ *s.t.* $[M]_k$ *(or* $[M]_{p^+}$*) and* $\blacksquare_l$ *are computationally indistinguishable (in the sense of our formalisation)*

2. **Non-malleability** *"[...] it [is] infeasible for an adversary, given a ciphertext, to produce a valid ciphertext (under the same encryption-key) for a related plaintext." [Gol04, Page 470]*

   $$\models (\mathtt{Eve\ k\ } [M]_k \wedge \underbrace{\phi(M) \wedge \phi(M')}_{M' \textit{ is related to } M}) \rightarrow (\mathtt{Eve\ k\ } [M']_k \rightarrow \mathtt{Eve\ k\ } k)$$

   $$\models (\mathtt{Eve\ k\ } [M]_{p^+} \wedge \phi(M) \wedge \phi(M')) \rightarrow (\mathtt{Eve\ k\ } [M']_{p^+} \rightarrow \mathtt{Eve\ k\ } M')$$

Formalisation 10 (Unforgeability of signature schemes)
"it is infeasible to produce signatures of other users *to documents they did not sign." [Gol04, Page 498]*

$$\models a \mathsf{\ authored\ }]M[_p \rightarrow a \mathsf{\ k\ } p$$

⌟

Attacks on encryption schemes We state formalisations of attacks on encryption schemes as vulnerabilities and in increasing strength.

Formalisation 11 (Ciphertext-only attack)
"[...] the adversary (or cryptanalyst) tries [⊇] to deduce the decryption key [k (symmetric) resp. p (private)] or plaintext [M] by only [≡] observing ciphertext $[m_1, \ldots, m_n]$*." [MvOV96, Page 41]*

$$\begin{aligned} \mathtt{Eve\ k\ } M &\equiv \exists(k : \mathtt{K})(\mathtt{Eve\ k\ } [M]_k) \\ \mathtt{Eve\ k\ } M &\equiv \exists(p : \mathtt{K}^-)(\mathtt{Eve\ k\ } [M]_{p^+}) \end{aligned}$$

$$\begin{aligned} \mathtt{Eve\ k\ } k &\equiv (\exists(m_1 : \mathtt{SC}_k[\mathtt{M}])(\mathtt{Eve\ k\ } m_1) \wedge \cdots \wedge \exists(m_n : \mathtt{SC}_k[\mathtt{M}])(\mathtt{Eve\ k\ } m_n)) \\ \mathtt{Eve\ k\ } p &\equiv (\exists(m_1 : \mathtt{AC}_{p^+}[\mathtt{M}])(\mathtt{Eve\ k\ } m_1) \wedge \cdots \wedge \exists(m_n : \mathtt{AC}_{p^+}[\mathtt{M}])(\mathtt{Eve\ k\ } m_n)) \end{aligned}$$

⌟

Formalisation 12 (Known-plaintext attack)

"[...] the adversary has a quantity of plaintext $[m_1, \ldots, m_n]$ and corresponding ciphertext." [MvOV96, Page 41]

$$\begin{aligned}\mathtt{Eve\ k}\ k &\equiv (\exists m_1(\mathtt{Eve\ k}\ m_1 \wedge \mathtt{Eve\ k}\ [m_1]_k) \wedge \cdots \wedge \\ &\quad\ \exists m_n(\mathtt{Eve\ k}\ m_n \wedge \mathtt{Eve\ k}\ [m_n]_k)) \\ \mathtt{Eve\ k}\ p &\equiv (\exists m_1(\mathtt{Eve\ k}\ m_1 \wedge \mathtt{Eve\ k}\ [m_1]_{p^+}) \wedge \cdots \wedge \\ &\quad\ \exists m_n(\mathtt{Eve\ k}\ m_n \wedge \mathtt{Eve\ k}\ [m_n]_{p^+}))\end{aligned}$$

Our interpretation of 'a quantity of plaintext' can be refined from 'a number of plaintexts' to 'a number of parts of *plaintexts' by replacing* $\mathtt{Eve\ k}\ m_1$ *with* $\exists m_{11}(m_{11} \preccurlyeq m_1 \wedge \mathtt{Eve\ k}\ m_{11}) \wedge \cdots \wedge \exists m_{1i}(m_{1i} \preccurlyeq m_1 \wedge \mathtt{Eve\ k}\ m_{1i})$*, and* $\mathtt{Eve\ k}\ m_n$ *with* $\exists m_{n1}(m_{n1} \preccurlyeq m_1 \wedge \mathtt{Eve\ k}\ m_{n1}) \wedge \cdots \wedge \exists m_{nj}(m_{nj} \preccurlyeq m_1 \wedge \mathtt{Eve\ k}\ m_{nj})$. ⌟

Formalisation 13 (Chosen-plaintext attack)

"[...] the adversary chooses [▷] plaintext [m] and is then given [▶] corresponding ciphertext. Subsequently [⨁], the adversary uses any information deduced [⊒] in order to recover plaintext [M] corresponding to previously unseen ciphertext." [MvOV96, Page 41]

$$\begin{aligned}&\exists(k : \mathtt{K})(\neg\,\mathtt{Eve\ k}\ [M]_k \wedge \\ &\qquad \exists m(\mathtt{Eve\ k}\ m \ \triangleright \\ &\qquad\quad (\mathtt{Eve\ k}\ [m]_k \blacktriangleright \bigoplus(M \sqsupseteq_{\mathtt{Eve}} (m, [m]_k)))))\\ &\exists(p : \mathtt{K}^-)(\neg\,\mathtt{Eve\ k}\ [M]_{p^+} \wedge \\ &\qquad \exists m(\mathtt{Eve\ k}\ m \ \triangleright \\ &\qquad\quad (\mathtt{Eve\ k}\ [m]_{p^+} \blacktriangleright \bigoplus(M \sqsupseteq_{\mathtt{Eve}} (m, [m]_{p^+})))))\end{aligned}$$

*Our interpretation of 'plaintext' can be refined from '*the *plaintext' to '*some of *the plaintext' by replacing* $\mathtt{Eve\ k}\ M$ *with* $\exists m_1(m_1 \preccurlyeq M \wedge \mathtt{Eve\ k}\ m_1) \wedge \cdots \wedge \exists m_n(m_n \preccurlyeq M \wedge \mathtt{Eve\ k}\ m_n)$. ⌟

Formalisation 14 (Adaptive chosen-plaintext attack)

"[...] a chosen-plaintext attack wherein the choice of plaintext may depend on the ciphertext received from previous requests." [MvOV96, Page 41]

Let A denote chosen-plaintext chains in the public key p^+ *of the form*

$$A ::= \blacksquare \mid ((M, [M]_{p^+}), A)$$

and $\mathsf{aCPC}_a^{p^+}$ *an inductively-defined macro expressing the realisation of such a chain for agent a*

$$\begin{aligned}\blacksquare\ \mathsf{aCPC}_a^{p^+}\ \phi &:= \phi \\ ((M, [M]_{p^+}), A)\ \mathsf{aCPC}_a^{p^+}\ \phi &:= a\ \mathtt{k}\ M \triangleright (a\ \mathtt{k}\ [M]_{p^+} \blacktriangleright A\ \mathsf{aCPC}_a^{p^+}\ \phi)\end{aligned}$$

Then

$$\begin{aligned}&\exists(p : \mathtt{K}^-)(\neg\,\mathtt{Eve\ k}\ [M]_{p^+} \wedge \\ &\qquad \exists m(\exists m'(m'\ \mathsf{aCPC}_{\mathtt{Eve}}^{p^+}\ \mathtt{Eve\ k}\ m) \ \triangleright \\ &\qquad\quad (\mathtt{Eve\ k}\ [m]_{p^+} \blacktriangleright \bigoplus(M \sqsupseteq_{\mathtt{Eve}} (m, [m]_{p^+})))))\end{aligned}$$

formalises an adaptive chosen-plaintext attack on a private key. (The formalisation of a corresponding attack on a symmetric key is similar.) ⌟

Formalisation 15 (Chosen-ciphertext attack)

"[...] the adversary selects the ciphertext and is then given the corresponding plaintext." [MvOV96, Page 41]

$$\begin{array}{l}\exists(k : \mathtt{K})(\neg\, \mathtt{Eve\ k}\ [M]_k \wedge \\ \quad \exists m(\mathtt{Eve\ k}\ [m]_k \ \rhd \\ \quad\quad (\mathtt{Eve\ k}\ m \blacktriangleright \oplus\!\!\!\!\!\!+(M \supseteq_{\mathtt{Eve}} (m, [m]_k)))))\end{array}$$

⌟

Formalisation 16 (Adaptive chosen-ciphertext attack)

"[...] a chosen-ciphertext attack where the choice of ciphertext may depend on the plaintext received from previous requests." [MvOV96, Page 42]

Let S denote chosen-ciphertext chains in the symmetric key k of the form

$$S ::= \blacksquare \mid (([M]_k, M), S)$$

and sCCC_a^k an inductively-defined macro expressing the realisation of such a chain for agent a

$$\begin{array}{rcl}\blacksquare\ \mathsf{sCCC}_a^k\ \phi & := & \phi \\ (([M]_k, M), S)\ \mathsf{sCCC}_a^k\ \phi & := & a\ \mathtt{k}\ [M]_k \rhd (a\ \mathtt{k}\ M \blacktriangleright S\ \mathsf{sCCC}_a^k\ \phi)\end{array}$$

Then

$$\begin{array}{l}\exists(k : \mathtt{K})(\neg\, \mathtt{Eve\ k}\ [M]_k \wedge \\ \quad \exists m(\exists m'(m'\ \mathsf{sCCC}^k_{\mathtt{Eve}}\ \mathtt{Eve\ k}\ m) \ \rhd \\ \quad\quad (\mathtt{Eve\ k}\ [m]_k \blacktriangleright \oplus\!\!\!\!\!\!+(M \supseteq_{\mathtt{Eve}} (m, [m]_k)))))\end{array}$$

formalises an adaptive chosen-plaintext attack on a symmetric key. (The formalisation of a corresponding attack on an asymmetric key is similar.) ⌟

Attacks on signature schemes We state formalisations of attacks on signature schemes in increasing strength.

Formalisation 17 (Key-only attack)

"[...] an adversary knows only the signer's public key." [MvOV96, Page 432]

$$\exists v \exists(a : \mathtt{A})(v^+\ \mathsf{puk}\ a \wedge \mathtt{Eve\ k}\ v^+ \wedge \forall m(\mathtt{Eve\ k}\ ]m[_v \rightarrow \neg\, \mathtt{Eve\ k}\ m))$$

⌟

Formalisation 18 (Known-message attack)

"An adversary has signatures for a set of messages which are known to the adversary but not chosen by him." [MvOV96, Page 432]

$$\begin{array}{l}\exists v \exists(a : \mathtt{A})(v^+\ \mathsf{puk}\ a \wedge \mathtt{Eve\ k}\ v^+ \ \wedge \\ \quad \exists m_1 \cdots \exists m_n(\mathtt{Eve\ k}\ ]m_1[_v \wedge \cdots \wedge \mathtt{Eve\ k}\ ]m_n[_v \ \wedge \\ \quad\quad \mathtt{Eve\ k}\ m_1 \wedge \cdots \wedge \mathtt{Eve\ k}\ m_n))\end{array}$$

⌟

Formalisation 19 (Chosen-message attack)

"An adversary obtains valid signatures from a chosen list of messages before attempting to break the signature scheme." [MvOV96, Page 433]

$$\exists v \exists (a : \mathtt{A})(v^+ \ \mathsf{puk}\ a \wedge \mathtt{Eve}\ \mathsf{k}\ v^+ \wedge \exists m_1 \cdots \exists m_n((\mathtt{Eve}\ \mathsf{k}\ m_1 \wedge \cdots \wedge \mathtt{Eve}\ \mathsf{k}\ m_n) \rhd (\mathtt{Eve}\ \mathsf{k}\ ]m_1[_v \wedge \cdots \wedge \mathtt{Eve}\ \mathsf{k}\ ]m_n[_v)))$$

⌟

Formalisation 20 (Adaptive chosen-message attack)

"An adversary is allowed to use the signer as an oracle; the adversary may request signatures of messages which depend on the signer's public key and he may request signatures of messages which depend on previously obtained signatures or messages." [MvOV96, Page 433]

Let C denote chosen-message chains in the private key p of the form

$$C ::= \blacksquare \mid ((M,]M[_p), C)$$

and CMC_a^p an inductively-defined macro expressing the realisation of such a chain for agent a

$$\begin{aligned} \blacksquare\ \mathsf{CMC}_a^p\ \phi &:= \phi \\ ((M,]M[_p), C)\ \mathsf{CMC}_a^p\ \phi &:= a\ \mathsf{k}\ M \rhd (a\ \mathsf{k}\ ]M[_p \blacktriangleright C\ \mathsf{CMC}_a^p\ \phi) \end{aligned}$$

Then

$$\exists v \exists (a : \mathtt{A})(v^+ \ \mathsf{puk}\ a \wedge \mathtt{Eve}\ \mathsf{k}\ v^+ \wedge \exists m(m\ \mathsf{CMC}_a^v\ \mathtt{Eve}\ \mathsf{k}\ m))$$

formalises an adaptive chosen-message attack on a signing key. ⌟

Breaks of signature schemes We state formalisations of breaks of signature schemes in increasing strength.

Formalisation 21 (Existential forgery)

"An adversary is able to forge a signature for at least one message. The adversary has little or no control over the message whose signature is obtained, and the legitimate signer may be involved in the deception . . . " [MvOV96, Page 432]

$$\exists v \exists (a : \mathtt{A})(v^+ \ \mathsf{puk}\ a \wedge \mathtt{Eve}\ \mathsf{k}\ v^+ \wedge \exists m(]m[_v \sqsupseteq_{\mathtt{Eve}} m))$$

⌟

Formalisation 22 (Selective forgery)

"An adversary is able to create a valid signature for a particular [∃] message or class of messages chosen [▷] a priori. Creating the signature does not directly involve the legitimate signer." [MvOV96, Page 432]

$$\exists v \exists (a : \mathtt{A})(v^+ \ \mathsf{puk}\ a \wedge \mathtt{Eve}\ \mathsf{k}\ v^+ \wedge \exists m(\mathtt{Eve}\ \mathsf{k}\ m \rhd (]m[_v \sqsupseteq_{\mathtt{Eve}} m)))$$

⌟

Formalisation 23 (Universal forgery)

"An adversary is able to create a valid signature for an arbitrary [∀] message chosen a priori."

$$\exists v \exists (a : \mathtt{A})(v^{+}\ \mathsf{puk}\ a \wedge \mathtt{Eve}\ \mathsf{k}\ v^{+} \wedge \forall m(\mathtt{Eve}\ \mathsf{k}\ m \rhd (]m[_{v} \supseteq_{\mathtt{Eve}} m)))$$

⌟

Formalisation 24 (Total break)

"An adversary is either able to compute the private key information of the signer, or finds an efficient signing algorithm functionally equivalent to the valid signing algorithm." [MvOV96, Page 432]

$$\exists v \exists (a : \mathtt{A})(v^{+}\ \mathsf{puk}\ a \wedge \mathtt{Eve}\ \mathsf{k}\ v^{+} \wedge \mathtt{Eve}\ \mathsf{k}\ v)$$

⌟

Part III

Epilogue

Chapter 6

Conclusion

6.1 Review of achievements

We believe having achieved with CPL an original construction and powerful tool for the logical conceptualisation of the security of communication. In particular, we have:

1. defined a cryptographically meaningful (cf. Section 3.2.3.4) and indeed omnipresent *epistemic modality* (for propositional knowledge) that commutes in both senses with quantifiers being relativised (this is a novel idea) to individual knowledge (cf. Corollary 1)

2. invented a cryptographically interesting (cf. Section 3.2.3.2) *epistemic conditional* thanks to the auxiliary invention of *complex truth values*

3. pioneered the *application of spatial logic* (cf. Section 3.2.3.2) to the formalisation of cryptographic states of affairs and auxiliary concepts such as choice, compositionality, and corruption (cf. Section 3.3.5, 5.3.1, 5.3.2)

4. invented, by macro-definition, *spatial freeze quantifiers*, and shown that with them distributed temporal logic is definable within the spatio-temporal fragment of CPL (cf. Section 3.2.3.5)

5. (a) demonstrated the macro-definability of a *Gödel-style provability modality* within the spatio-epistemic fragment of CPL (cf. Theorem 2). With this modality, CPL can capture the provability meaning of intuitionistic implication, and provability is shown to be the key to the formalisation of *commitment* and related cryptographic states of affairs (cf. Section 3.3.4).

   (b) sketched a promising generalisation of the (non-interactive) provability modality to an *interactive provability* modality (cf. Definition 22) and applied it to the formalisation of zero-knowledge

6. demonstrated the definability of cryptographically meaningful (cf. Section 3.2.3.3) *deontic modalities* within the spatio-epistemico-temporal fragment of CPL, and by that, shown that cryptographic permission (and prohibition) is parametrically reducible to the desired notion of (undesired) *state of violation* of the employed crypto system

7. demonstrated that the *addition of dense real-time* to an untimed, *property*-based formalism for cryptographic protocols (core CPL) can be simple and backwards-compatible, when properly conceived, but is still powerful enough for the macro-definition of *durations* (cf. Section 3.4)

8. conceived and exemplified an original and promising *co-design* of poly-dimensional modal logic (CPL) and formal program semantics (C^3):

   Co-design In particular, we have demonstrated that

   (a) *temporal accessibility* can be identified with *protocol execution* (cf. Chapter 4)

   (b) the *execution constraints* of the operational semantics (defining protocol execution) are *CPL-definable and -checkable* (cf. Table 4.4)

   (c) *epistemic accessibility* can be identified with *static observational equivalence* on program execution states (cf. Section 4.1.3)

   (d) the meaning of a cryptographic protocol, its *denotational semantics* (cf. Section 4.3.2), can be defined in terms of the (possibly context-sensitive) meaning of the cryptographic messages it produces during its execution, and the (communicable) meaning of a cryptographic message (cf. Section 4.3.1) can in turn be defined via hypothetical knowledge (and provability)

   (e) the *information content* (in the sense of Kolmogorov) of a cryptographic protocol can be defined in terms of the information content of a cryptographic message, which in turn can be defined in terms of the minimal process inducing the (individual) knowledge of that message (cf. Definition 17 resp. 15)

   (f) *protocol agents* can be conceived as evolving *Scott information systems* (cf. Section 4.3.2)

   C^3 In particular, we have (in joint work):

   (a) invented a dynamic *key lookup mechanism* based on the explicit declaration of key ownership via tagging (cf. Table 4.3)

   (b) defined a comprehensive *message reception mechanism* integrating pattern-matching (cf. Table 4.2) and conditional input (cf. Rules IN and SIN in Table 4.4) as linguistic abstractions for cryptographic computation

   (c) demonstrated that the *addition of dense real-time* to an untimed, *model*-based formalism for cryptographic protocols can be simple and backwards-compatible, when properly conceived (cf. Section 4.2)

   (d) invented a powerful *specification technique* based on *epistemically non-local processes* complementing the traditional techniques of meta-communication (via formal parameters) and out-of-band communication (via secure I/O primitives) (cf. Section 4.2.2)

9. conceived a promising, backwards-compatible extension of core CPL with a notion of probabilistic polynomial-time computation by resource-bounding individual and propositional knowledge in a novel way and by inventing a notion of *belief with error control* (cf. Chapter 5)

10. conceived a novel *modal encoding* of weak second-order logic via *individual-generating individuals* (message-generating protocols) (cf. Section 3.2.3.1).

The key to our backwards-compatible extensions is the paradigm of *event-based modelling*. That is, the conservative extension of protocol histories with new protocol events, e.g., clock-set events for the extension with real time and denotation events for the extension with probabilistic polynomial-time. In consequence, old languages can be interpreted over new models because new modelling events are irrelevant to that (oblivious) interpretation.

Thanks to the powerful linguistic abstractions that CPL provides, we have also achieved the logical formalisation of an unprecedented variety of cryptographic states of affairs and cryptographic concepts. Concretely, these formalised states of affairs are:

Trust-related affairs maliciousness, honesty, faultiness, prudency, and trustworthiness of protocol agents (cf. Section 3.3.1).

Confidentiality-related affairs shared secret, secrecy, anonymity, data derivation, non-interaction, perfect forward secrecy, known-key attack, and agent corruption (cf. Section 3.3.2).

Authentication-related affairs key confirmation, key authentication (implicit and explicit), message integrity, message authorship, message authentication (authenticity), key transport (unacknowledged and acknowledged), key agreement (unacknowledged and acknowledged), entity authentication (identification) (unilateral, weakly mutual, and strongly mutual) (cf. Section 3.3.3).

Commitment-related affairs cryptographic proof, cryptographic evidence, provability, non-repudiation, contract signing, (optimism, completion, accountability, and abuse-freeness) (cf. Section 3.3.4).

Compositionality-related affairs key separation, compositional correctness (existential composability, conditional composability, and universal composability), and attack scenario (cf. Section 3.3.5).

Concretely, these (tentatively) formalised concepts are:

Fundamental concepts one-way function, hard-core predicate, computational indistinguishability, (n-party) interactive proof, interactive provability, proof of knowledge, and (n-prover) zero-knowledge (cf. Section 5.3.1).

Applied concepts security of encryption schemes (standard, semantic, via indistinguishability, and via non-malleability), unforgeability of signature schemes, attacks on encryption schemes (ciphertext-only attack, known-plaintext attack, chosen-plaintext attack, adaptive chosen-plaintext attack, chosen-ciphertext attack, and adaptive chosen-ciphertext attack), attacks on signature schemes (key-only attack, known-message attack, chosen-message attack, and adaptive chosen-message attack), and breaks of signature schemes (existential forgery, selective forgery, universal forgery, and total break) (cf. Section 5.3.2).

We hope that our tentative formalisations have convinced the reader that CPL is an interesting candidate as a *lingua franca* for requirements-engineering cryptographic protocols.

6.2 Future work

CPL

Short-term Our immediate concerns are the consolidation of ppCPL, and the validation of our formalisations of fundamental and applied concepts w.r.t. their traditional Turing-machine-based definitions.

Mid-term Our next concerns are the construction of proof systems for core CPL, tCPL, and ppCPL; and the study of decidable fragments of core CPL and its extensions.

Long-term Our long-term concerns are the construction of a Curry-Howard isomorphism between cryptographic protocols and propositions; and the extension of CPL with quantum cryptography.

C^3

Short-term Our short-term concerns are a symbolic version of C^3, and the axiomatisation of C^3's observational equivalence.

Mid-term Our mid-term concern is the extension of C^3 with a notion of probabilistic-polynomial-time computation.

Long-term Our long-term concerns are the extension of C^3 with quantum computation and communication; the investigation of different degrees of communication (of: messages, message types, logical formulae, processes, process types); and the extension of C^3 with further key management capabilities (e.g., public-key certificate issuing and revocation) and object-orientation (e.g., method-based sub-protocol calls and dynamic session creation).

Finally, we are concerned with the conception of an integrated *engineering methodology* for our formalisms, and the development of *tool support*, and application to more *case studies*.

Appendix A

Proofs

Proof of Theorem 1

- Barcan: Suppose that $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$ and $\mathfrak{s} \models \forall m(\mathsf{K}_a(\phi))$. Then, for all $M \in \mathcal{M}$, $\mathfrak{s} \models \mathsf{K}_a(\phi)$. Hence, for all $M \in \mathcal{M}$, $\mathfrak{s} \models \phi$ (by $\models \mathsf{K}_a(\phi) \rightarrow \phi$), and thus $\mathfrak{s} \models \forall m(\phi)$. Conclude that $\mathfrak{s} \models \mathsf{K}_a(\forall m(\phi))$ by the hypothesis that $\mathfrak{s} \models \mathsf{K}_a(\phi)$ and the fact that keys$(\forall m(\phi)) \subset$ keys(ϕ) (fulfilling the truth condition for propositional knowledge).
- Relativised co-Barcan: Suppose that $\mathfrak{s} \in \mathcal{H} \times \mathcal{P}$ and $\mathfrak{s} \models \mathsf{K}_a(\forall m(a \mathsf{k} m \rightarrow \phi))$. Further, suppose that $M \in \mathcal{M}$ and $\mathfrak{s} \models a \mathsf{k} M$. Hence, $\mathfrak{s} \models \mathsf{K}_a(a \mathsf{k} M)$ and $\mathfrak{s} \models \mathsf{K}_a(a \mathsf{k} M \rightarrow \phi)$. Consequently, $\mathfrak{s} \models \mathsf{K}_a(\phi)$ (by $\models \mathsf{K}_a(\varphi \rightarrow \varphi') \rightarrow (\mathsf{K}_a(\varphi) \rightarrow \mathsf{K}_a(\varphi'))$). Conclude that $\mathfrak{s} \models a \mathsf{k} M \rightarrow \mathsf{K}_a(\phi)$ by the hypothesis $\mathfrak{s} \models a \mathsf{k} M$, and finally that $\mathfrak{s} \models \forall m(a \mathsf{k} m \rightarrow \mathsf{K}_a(\phi))$.

Proof of Proposition 2 by counterexample (logical equivalence is incompatible with K_a):

- $\mathtt{Eve} : \mathtt{Adv} \Leftrightarrow k : \mathtt{K}$ because $\models \mathtt{Eve} : \mathtt{Adv}$ and $\models k : \mathtt{K}$, but
- $\mathsf{K}_a(\mathtt{Eve} : \mathtt{Adv}) \not\Leftrightarrow \mathsf{K}_a(k : \mathtt{K})$ because the establishment of the truth of $\mathsf{K}_a(\mathtt{Eve} : \mathtt{Adv})$ does not dependent on a's knowledge of the key values in $\mathtt{Eve} : \mathtt{Adv}$ (there aren't any), whereas the establishment of the truth of $\mathsf{K}_a(k : \mathtt{K})$ does dependent on a's knowledge of the key values in $k : \mathtt{K}$ (there is one).

Hence, $\models \mathsf{K}_a(\mathtt{Eve} : \mathtt{Adv})$, i.e., $\mathsf{K}_a(\mathtt{Eve} : \mathtt{Adv})$ is a logical truth (tautology), but $\not\models \mathsf{K}_a(k : \mathtt{K})$, i.e., $\mathsf{K}_a(k : \mathtt{K})$ is only a contingent truth. Yet, a logical truth is, by definition, *not* logically equivalent to a contingent truth.

Lemma 1 $\phi \Rightarrow \phi'$ *iff* $\models \phi \rightarrow \phi'$

Proof.

$\models \phi \rightarrow \phi'$ iff
for all $\mathfrak{s}$, $\mathfrak{s} \models \phi \rightarrow \phi'$ iff
for all $\mathfrak{s}$, $\mathfrak{s} \models \neg\phi \vee \phi'$ iff
for all $\mathfrak{s}$, $\mathfrak{s} \models \neg(\neg\neg\phi \wedge \neg\phi')$ iff

for all $\mathfrak{s}$, not $\mathfrak{s} \models \neg\neg\phi \wedge \neg\phi'$ iff
for all $\mathfrak{s}$, not ($\mathfrak{s} \models \neg\neg\phi$ and $\mathfrak{s} \models \neg\phi'$) iff
for all $\mathfrak{s}$, not (not not $\mathfrak{s} \models \phi$ and not $\mathfrak{s} \models \phi'$) iff
for all $\mathfrak{s}$, not ($\mathfrak{s} \models \phi$ and not $\mathfrak{s} \models \phi'$) iff
for all $\mathfrak{s}$, not $\mathfrak{s} \models \phi$ or not not $\mathfrak{s} \models \phi'$ iff
for all $\mathfrak{s}$, not $\mathfrak{s} \models \phi$ or $\mathfrak{s} \models \phi'$ iff
for all $\mathfrak{s}$, if $\mathfrak{s} \models \phi$ then $\mathfrak{s} \models \phi'$ iff
$\phi \Rightarrow \phi'$

Proposition 5 $\models \mathsf{P}_a(\phi \to \phi') \to (\mathsf{P}_a(\phi) \to \mathsf{P}_a(\phi'))$

Proof.

1.	$\mathfrak{s} \models \mathsf{P}_a(\phi \to \phi')$	hyp.
2.	$\mathfrak{s} \models \mathsf{P}_a(\phi)$	hyp.
3.	$\mathfrak{s} \models \exists m(m \ \mathsf{proofFor}\ (\phi \to \phi') \wedge a \ \mathsf{k}\ m)$	1
4.	there is $M \in \mathcal{M}$ s.t. $\mathfrak{s} \models M \ \mathsf{proofFor}\ (\phi \to \phi') \wedge a \ \mathsf{k}\ M$	3
5.	$M \in \mathcal{M}$ and $\mathfrak{s} \models M \ \mathsf{proofFor}\ (\phi \to \phi') \wedge a \ \mathsf{k}\ M$	hyp.
6.	$\mathfrak{s} \models \exists m(m \ \mathsf{proofFor}\ (\phi) \wedge a \ \mathsf{k}\ m)$	2
7.	there is $M \in \mathcal{M}$ s.t. $\mathfrak{s} \models M \ \mathsf{proofFor}\ (\phi) \wedge a \ \mathsf{k}\ M$	6
8.	$M' \in \mathcal{M}$ and $\mathfrak{s} \models M' \ \mathsf{proofFor}\ (\phi) \wedge a \ \mathsf{k}\ M'$	hyp.
9.	$\mathfrak{s} \models M \ \mathsf{proofFor}\ (\phi \to \phi')$ and $\mathfrak{s} \models a \ \mathsf{k}\ M$	5
10.	$\mathfrak{s} \models \forall(v : \mathtt{A}_{\mathtt{Adv}})(v \ \mathsf{k}\ M \rhd \mathsf{K}_v(\phi \to \phi'))$	9
11.	for all $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, $\mathfrak{s} \models v \ \mathsf{k}\ M \rhd \mathsf{K}_v(\phi \to \phi')$	10
12.	$\mathfrak{s} \models M' \ \mathsf{proofFor}\ (\phi)$ and $\mathfrak{s} \models a \ \mathsf{k}\ M'$	8
13.	$\mathfrak{s} \models \forall(v : \mathtt{A}_{\mathtt{Adv}})(v \ \mathsf{k}\ M' \rhd \mathsf{K}_v(\phi))$	12
14.	for all $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, $\mathfrak{s} \models v \ \mathsf{k}\ M' \rhd \mathsf{K}_v(\phi)$	13
15.	$v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$	hyp.
16.	$\mathfrak{s}' \models v \ \mathsf{k}\ (M, M')$	hyp.
17.	$\mathfrak{s}' \models v \ \mathsf{k}\ M$	16, property of k
18.	$\mathfrak{s} \models v \ \mathsf{k}\ M \rhd \mathsf{K}_v(\phi \to \phi')$	11, 15
19.	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models v \ \mathsf{k}\ M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi \to \phi')$	18
20.	if $\mathfrak{s}' \models v \ \mathsf{k}\ M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi \to \phi')$	19
21.	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi \to \phi')$	17, 20
22.	$\mathfrak{s}' \models v \ \mathsf{k}\ M'$	16, property of k
23.	$\mathfrak{s} \models v \ \mathsf{k}\ M' \rhd \mathsf{K}_v(\phi)$	14, 15
24.	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models v \ \mathsf{k}\ M'$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi)$	23
25.	if $\mathfrak{s}' \models v \ \mathsf{k}\ M'$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi)$	24
26.	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi)$	22, 25
27.	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi')$	21, 26, property of K
28.	if $\mathfrak{s}' \models v \ \mathsf{k}\ (M, M')$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi')$	16, 27

29.	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models v \mathsf{k} (M, M')$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi')$	28
30.	$\mathfrak{s} \models v \mathsf{k} (M, M') \rhd \mathsf{K}_v(\phi')$	29
31.	if $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$ then $\mathfrak{s} \models v \mathsf{k} (M, M') \rhd \mathsf{K}_v(\phi')$	15, 30
32.	for all $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, $\mathfrak{s} \models v \mathsf{k} (M, M') \rhd \mathsf{K}_v(\phi')$	31
33.	$\mathfrak{s} \models \forall(v : \mathtt{A}_{\mathtt{Adv}})(v \mathsf{k} (M, M') \rhd \mathsf{K}_v(\phi'))$	32
34.	$\mathfrak{s} \models (M, M')$ proofFor ϕ'	33
35.	$\mathfrak{s} \models a \mathsf{k} (M, M')$	9, 12, property of k
36.	$\mathfrak{s} \models (M, M')$ proofFor ϕ' and $\mathfrak{s} \models a \mathsf{k} (M, M')$	34, 35
37.	$\mathfrak{s} \models (M, M')$ proofFor $\phi' \wedge a \mathsf{k} (M, M')$	36
38.	there is $M'' \in \mathcal{M}$ s.t. $\mathfrak{s} \models M''$ proofFor $\phi' \wedge a \mathsf{k} M''$	37
39.	$\mathfrak{s} \models \exists m(m$ proofFor $\phi' \wedge a \mathsf{k} m)$	38
40.	$\mathfrak{s} \models \mathsf{P}_a(\phi')$	39
41.	$\mathfrak{s} \models \mathsf{P}_a(\phi')$	7, 40
42.	$\mathfrak{s} \models \mathsf{P}_a(\phi')$	4, 41
43.	if $\mathfrak{s} \models \mathsf{P}_a(\phi)$ then $\mathfrak{s} \models \mathsf{P}_a(\phi')$	2, 42
44.	$\mathfrak{s} \models \mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\phi')$	43
45.	if $\mathfrak{s} \models \mathsf{P}_a(\phi \rightarrow \phi')$ then $\mathfrak{s} \models \mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\phi')$	1, 44
46.	for all $\mathfrak{s}$, if $\mathfrak{s} \models \mathsf{P}_a(\phi \rightarrow \phi')$ then $\mathfrak{s} \models \mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\phi')$	45
47.	$\mathsf{P}_a(\phi \rightarrow \phi') \Rightarrow \mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\phi')$	46, Definition 8
48.	$\models \mathsf{P}_a(\phi \rightarrow \phi') \rightarrow (\mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\phi'))$	47, Lemma 1

Proposition 6 $\models \mathsf{P}_a(\phi) \rightarrow \mathsf{K}_a(\phi)$

Proof.

1.	$\mathfrak{s} \models \mathsf{P}_a(\phi)$	hyp.
2.	$\mathfrak{s} \models \exists m(m$ proofFor $\phi \wedge a \mathsf{k} m)$	1
3.	there is $M \in \mathcal{M}$ s.t. $\mathfrak{s} \models M$ proofFor $\phi \wedge a \mathsf{k} M$	2
4.	$M \in \mathcal{M}$ and $\mathfrak{s} \models M$ proofFor $\phi \wedge a \mathsf{k} M$	hyp.
5.	$\mathfrak{s} \models M$ proofFor ϕ and $\mathfrak{s} \models a \mathsf{k} M$	4
6.	$\mathfrak{s} \models \forall(v : \mathtt{A}_{\mathtt{Adv}})(v \mathsf{k} M \rhd \mathsf{K}_v(\phi))$	5
7.	for all $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, $\mathfrak{s} \models v \mathsf{k} M \rhd \mathsf{K}_v(\phi)$	6
8.	$\mathfrak{s} \models a \mathsf{k} M \rhd \mathsf{K}_a(\phi)$	7
9.	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi)$	8
10.	if $\mathfrak{s} \models a \mathsf{k} M$ then $\mathfrak{s} \circ \mathfrak{s} \models \mathsf{K}_a(\phi)$	9
11.	$\mathfrak{s} \circ \mathfrak{s} \models \mathsf{K}_a(\phi)$	5, 10
12.	$\mathfrak{s} \models \mathsf{K}_a(\phi)$	11[1]
13.	$\mathfrak{s} \models \mathsf{K}_a(\phi)$	3, 12
14.	if $\mathfrak{s} \models \mathsf{P}_a(\phi)$ then $\mathfrak{s} \models \mathsf{K}_a(\phi)$	1, 13

[1] $\circ$ is supposed to preserve uniqueness of process terms and of protocol events in protocol histories, i.e., $\circ$ is supposed to be idempotent

15. for all $\mathfrak{s}$, if $\mathfrak{s} \models \mathsf{P}_a(\phi)$ then $\mathfrak{s} \models \mathsf{K}_a(\phi)$ — 14
16. $\mathsf{P}_a(\phi) \Rightarrow \mathsf{K}_a(\phi)$ — 15, Definition 8
17. $\models \mathsf{P}_a(\phi) \rightarrow \mathsf{K}_a(\phi)$ — 16, Lemma 1

Proposition 7 $\models \mathsf{P}_a(\phi) \rightarrow \phi$

Proof.

1. $\models \mathsf{P}_a(\phi) \rightarrow \mathsf{K}_a(\phi)$ — Proposition 6
2. $\models \mathsf{K}_a(\phi) \rightarrow \phi$ — property of K
3. $\models \mathsf{P}_a(\phi) \rightarrow \phi$ — 1, 2

Proposition 8 $\models \mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\mathsf{P}_a(\phi))$

Proof.

1. $\mathfrak{s} \models \mathsf{P}_a(\phi)$ — hyp.
2. $\mathfrak{s} \models \exists m(m \ \mathsf{proofFor} \ \phi \wedge a \ \mathsf{k} \ m)$ — 1
3. there is $M \in \mathcal{M}$ s.t. $\mathfrak{s} \models M \ \mathsf{proofFor} \ \phi \wedge a \ \mathsf{k} \ M$ — 2
4. $M \in \mathcal{M}$ and $\mathfrak{s} \models M \ \mathsf{proofFor} \ \phi$ and $\mathfrak{s} \models a \ \mathsf{k} \ M$ — 3
5. $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$ — hyp.
6. $\mathfrak{s}' \models v \ \mathsf{k} \ M$ — hyp.
7. $\mathfrak{s}' \circ \mathfrak{s} \models v \ \mathsf{k} \ M$ — 6
8. $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(a \ \mathsf{k} \ M)$ — 4, 7, definition of K
9. $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(M \ \mathsf{proofFor} \ \phi)$ — 4, 7
10. $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\mathsf{P}_a(\phi))$ — 8, 9
11. if $\mathfrak{s}' \models v \ \mathsf{k} \ M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\mathsf{P}_a(\phi))$ — 6, 10
12. for all $\mathfrak{s}'$, if $\mathfrak{s}' \models v \ \mathsf{k} \ M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\mathsf{P}_a(\phi))$ — 11
13. $\mathfrak{s} \models v \ \mathsf{k} \ M \rhd \mathsf{K}_v(\mathsf{P}_a(\phi))$ — 12
14. if $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$ then $\mathfrak{s} \models v \ \mathsf{k} \ M \rhd \mathsf{K}_v(\mathsf{P}_a(\phi))$ — 5, 13
15. for all $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, $\mathfrak{s} \models v \ \mathsf{k} \ M \rhd \mathsf{K}_v(\mathsf{P}_a(\phi))$ — 14
16. $\mathfrak{s} \models \forall(v : \mathtt{A}_{\mathtt{Adv}})(v \ \mathsf{k} \ M \rhd \mathsf{K}_v(\mathsf{P}_a(\phi)))$ — 15
17. $\mathfrak{s} \models M \ \mathsf{proofFor} \ \mathsf{P}_a(\phi)$ and $\mathfrak{s} \models a \ \mathsf{k} \ M$ — 4, 16
18. there is $M \in \mathcal{M}$ s.t. $\mathfrak{s} \models M \ \mathsf{proofFor} \ \mathsf{P}_a(\phi) \wedge a \ \mathsf{k} \ M$ — 17
19. $\mathfrak{s} \models \exists m(m \ \mathsf{proofFor} \ \mathsf{P}_a(\phi) \wedge a \ \mathsf{k} \ m)$ — 18
20. $\mathfrak{s} \models \mathsf{P}_a(\mathsf{P}_a(\phi))$ — 19
21. $\mathfrak{s} \models \mathsf{P}_a(\mathsf{P}_a(\phi))$ — 3, 20
22. if $\mathfrak{s} \models \mathsf{P}_a(\phi)$ then $\mathfrak{s} \models \mathsf{P}_a(\mathsf{P}_a(\phi))$ — 1, 21
23. for all $\mathfrak{s}$, if $\mathfrak{s} \models \mathsf{P}_a(\phi)$ then $\mathfrak{s} \models \mathsf{P}_a(\mathsf{P}_a(\phi))$ — 22
24. $\mathsf{P}_a(\phi) \Rightarrow \mathsf{P}_a(\mathsf{P}_a(\phi))$ — 23, Definition 8
25. $\models \mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\mathsf{P}_a(\phi))$ — 24, Lemma 1

Proposition 9 $\dfrac{\models \phi}{\models a \mathsf{k} M \to \mathsf{P}_a(\phi)}$ *M is a tuple of the key values in ϕ*

Proof.

1.	$\models \phi$ and M is a tuple of the key values in ϕ	hyp.
2.	$\mathfrak{s} \models a \mathsf{k} M$	hyp.
3.	$v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$	hyp.
4.	$\mathfrak{s}' \models v \mathsf{k} M$	hyp.
5.	$\mathfrak{s}' \circ \mathfrak{s} \models v \mathsf{k} M$	4
6.	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi)$	1, 5, epistemic necessitation
7.	if $\mathfrak{s}' \models v \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi)$	4, 6
8.	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models v \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_v(\phi)$	7
9.	$\mathfrak{s} \models v \mathsf{k} M \rhd \mathsf{K}_v(\phi)$	8
10.	if $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$ then $\mathfrak{s} \models v \mathsf{k} M \rhd \mathsf{K}_v(\phi)$	3, 9
11.	for all $v \in \boldsymbol{\mathcal{A}}_{\mathtt{Eve}}$, $\mathfrak{s} \models v \mathsf{k} M \rhd \mathsf{K}_v(\phi)$	10
12.	$\mathfrak{s} \models \forall(v : \mathtt{A}_{\mathtt{Adv}})(v \mathsf{k} M \rhd \mathsf{K}_v(\phi))$	11
13.	$\mathfrak{s} \models M$ proofFor ϕ and $\mathfrak{s} \models a \mathsf{k} M$	2, 12
14.	there is $M \in \mathcal{M}$ s.t. $\mathfrak{s} \models M$ proofFor $\phi \wedge a \mathsf{k} M$	13
15.	$\mathfrak{s} \models \exists m(m$ proofFor $\phi \wedge a \mathsf{k} m)$	14
16.	$\mathfrak{s} \models \mathsf{P}_a(\phi)$	15
17.	if $\mathfrak{s} \models a \mathsf{k} M$ then $\mathfrak{s} \models \mathsf{P}_a(\phi)$	2, 16
18.	for all $\mathfrak{s}$, if $\mathfrak{s} \models a \mathsf{k} M$ then $\mathfrak{s} \models \mathsf{P}_a(\phi)$	17
19.	$a \mathsf{k} M \Rightarrow \mathsf{P}_a(\phi)$	18, Definition 8
20.	$\models a \mathsf{k} M \to \mathsf{P}_a(\phi)$	19, Lemma 1

Proof of Theorem 3

1. $\emptyset \neq [\![M]\!]^{\mathfrak{s}}_a \subset \mathcal{F}/_{\equiv}$: $\emptyset \neq [\![M]\!]^{\mathfrak{s}}_a$ because $[\top] \in [\![M]\!]^{\mathfrak{s}}_a$ due to $\models \mathsf{K}_a(\top)$ (and also $[a \mathsf{k} M] \in [\![M]\!]^{\mathfrak{s}}_a$ due to $\models a \mathsf{k} M \rhd \mathsf{K}_a(a \mathsf{k} M)$); and $[\![M]\!]^{\mathfrak{s}}_a \subset \mathcal{F}/_{\equiv}$ because $[\bot] \in \mathcal{F}/_{\equiv}$, but $[\bot] \notin [\![M]\!]^{\mathfrak{s}}_a$ due to $\not\models \mathsf{K}_a(\bot)$.

2. (a) if $[\phi], [\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$ then $[\phi] \wedge [\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$:

1	$[\phi] \in [\![M]\!]^{\mathfrak{s}}_a$ and $[\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$	hyp.
2	$\mathfrak{s} \models a \mathsf{k} M \rhd \mathsf{K}_a(\phi)$ and $\mathfrak{s} \models a \mathsf{k} M \rhd \mathsf{K}_a(\phi')$	1
3	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi)$ and for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi')$	2
4	for all $\mathfrak{s}'$, (if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi)$) and (if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi')$)	3
5	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\begin{pmatrix} \mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi) \text{ and} \\ \mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi') \end{pmatrix}$	4
6	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi) \wedge \mathsf{K}_a(\phi')$	5
7	$\models (\mathsf{K}_a(\phi) \wedge \mathsf{K}_a(\phi')) \to \mathsf{K}_a(\phi \wedge \phi')$	property of K_a

8	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi \wedge \phi')$	6, 7
9	$\mathfrak{s} \models a \mathsf{k} M \rhd \mathsf{K}_a(\phi \wedge \phi')$	8
10	$[\phi] \wedge [\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$	9

(b) if $[\phi] \in [\![M]\!]^{\mathfrak{s}}_a$ and $[\phi'] \in \mathcal{F}/_{\equiv}$ and $\phi \leq \phi'$ then $[\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$:

1	$[\phi] \in [\![M]\!]^{\mathfrak{s}}_a$ and $[\phi'] \in \mathcal{F}/_{\equiv}$ and $\phi \leq \phi'$	hyp.
2	$\mathfrak{s} \models a \mathsf{k} M \rhd \mathsf{K}_a(\phi)$ and $\phi \Rightarrow \phi'$ and keys$(\phi') \subseteq$ keys(ϕ)	1
3	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi)$	2
4	$\phi \Rightarrow \phi'$ iff $\models \phi \rightarrow \phi'$	Lemma
5	$\models \phi \rightarrow \phi'$	2, 4
6	if $\models \phi \rightarrow \phi'$ then $\models a \mathsf{k} K \rightarrow \mathsf{K}_a(\phi \rightarrow \phi')$ where K designates a tuple built from keys$(\phi \rightarrow \phi')$	prop. of K_a
7	$\models a \mathsf{k} K \rightarrow \mathsf{K}_a(\phi \rightarrow \phi')$	5, 6
8	$\models \mathsf{K}_a(\phi) \rightarrow a \mathsf{k} K'$ were K' designates a tuple built from keys(ϕ)	property of K_a
9	$\mathfrak{s}' \models a \mathsf{k} M$	hyp.
10	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi)$	3, 9
11	$\mathfrak{s}' \circ \mathfrak{s} \models a \mathsf{k} K'$	8, 10
12	$\mathfrak{s}' \circ \mathfrak{s} \models a \mathsf{k} K$	2, 11(, property of k)
13	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi \rightarrow \phi')$	7, 12
14	$\models \mathsf{K}_a(\phi \rightarrow \phi') \rightarrow (\mathsf{K}_a(\phi) \rightarrow \mathsf{K}_a(\phi'))$	property of K_a
15	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi) \rightarrow \mathsf{K}_a(\phi')$	13, 14
16	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi')$	10, 15
17	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi')$	9, 16
18	$\mathfrak{s} \models a \mathsf{k} M \rhd \mathsf{K}_a(\phi')$	17
19	$[\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$	18

3. if $[\phi] \vee [\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$ and $[\phi] \vee [\phi''] \in [\![M]\!]^{\mathfrak{s}}_a$ then $[\phi] \vee ([\phi'] \wedge [\phi'']) \in [\![M]\!]^{\mathfrak{s}}_a$:

1	$[\phi] \vee [\phi'] \in [\![M]\!]^{\mathfrak{s}}_a$ and $[\phi] \vee [\phi''] \in [\![M]\!]^{\mathfrak{s}}_a$	hyp.
2	$\mathfrak{s} \models a \mathsf{k} M \rhd \mathsf{K}_a(\phi \vee \phi')$ and $\mathfrak{s} \models a \mathsf{k} M \rhd \mathsf{K}_a(\phi \vee \phi'')$	1
3	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi \vee \phi')$ and for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi \vee \phi'')$	2
4	$\mathfrak{s}' \models a \mathsf{k} M$	hyp.
5	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi \vee \phi')$ and $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi \vee \phi'')$	3, 4
6	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi \vee \phi') \wedge \mathsf{K}_a(\phi \vee \phi'')$	5
7	$\models (\mathsf{K}_a(\phi \vee \phi') \wedge \mathsf{K}_a(\phi \vee \phi'')) \rightarrow \mathsf{K}_a((\phi \vee \phi') \wedge (\phi \vee \phi''))$	prop. K_a
8	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a((\phi \vee \phi') \wedge (\phi \vee \phi''))$	6, 7
9	$\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi \vee (\phi' \wedge \phi''))$	8
10	for all $\mathfrak{s}'$, if $\mathfrak{s}' \models a \mathsf{k} M$ then $\mathfrak{s}' \circ \mathfrak{s} \models \mathsf{K}_a(\phi \vee (\phi' \wedge \phi''))$	4, 9
11	$\mathfrak{s} \models a \mathsf{k} M \rhd \mathsf{K}_a(\phi \vee (\phi' \wedge \phi''))$	10
12	$[\phi] \vee ([\phi'] \wedge [\phi''])] \in [\![M]\!]^{\mathfrak{s}}_a$	11

Proof of Theorem 4 Analogous to the proof of Theorem 3 because that proof only relies on **S4** (adapted to the cryptographic setting), not full **S5** (adapted to the cryptographic setting), and provability is about **S4**.

Proof of Proposition 3

1. $[\![M]\!]^{\mathfrak{s}}_{\mathsf{P}_a} \subset [\![M]\!]^{\mathfrak{s}}_a$: because $\models \mathsf{P}_a(\phi) \to \mathsf{K}_a(\phi)$ (cf. Proposition 6), but $\not\models \mathsf{K}_a(\phi) \to \mathsf{P}_a(\phi)$
2. there is an order-embedding $e : [\![M]\!]^{\mathfrak{s}}_{\mathsf{P}_a} \hookrightarrow [\![M]\!]^{\mathfrak{s}}_a$: take $e := \mathrm{id}_{[\![M]\!]^{\mathfrak{s}}_a}\langle [\![M]\!]^{\mathfrak{s}}_{\mathsf{P}_a} \rangle$, i.e., the restriction to $[\![M]\!]^{\mathfrak{s}}_{\mathsf{P}_a}$ of the identity on $[\![M]\!]^{\mathfrak{s}}_a$.

Proof of Theorem 5

$[\![\mathfrak{s}]\!]_a = [\![\mathfrak{s}']\!]_a$ iff

$$T\left(\bigcup_{\substack{M \in \mathcal{M} \\ \mathfrak{s} \models a \,\mathsf{k}\, M}} [\![M]\!]^{\mathfrak{s}}_a\right) = T\left(\bigcup_{\substack{M \in \mathcal{M} \\ \mathfrak{s}' \models a \,\mathsf{k}\, M}} [\![M]\!]^{\mathfrak{s}'}_a\right) \quad \text{iff}$$

$$\bigcup_{\substack{M \in \mathcal{M} \\ \mathfrak{s} \models a \,\mathsf{k}\, M}} [\![M]\!]^{\mathfrak{s}}_a = \bigcup_{\substack{M \in \mathcal{M} \\ \mathfrak{s}' \models a \,\mathsf{k}\, M}} [\![M]\!]^{\mathfrak{s}'}_a \quad \text{iff}$$

$$\bigcup_{\substack{M \in \mathcal{M} \\ \mathfrak{s} \models a \,\mathsf{k}\, M}} \{\ [\phi] \mid \phi \in \mathcal{F}\ ;\ \mathfrak{s} \models a \,\mathsf{k}\, M \rhd \mathsf{K}_a(\phi)\ \} =$$
$$\bigcup_{\substack{M \in \mathcal{M} \\ \mathfrak{s}' \models a \,\mathsf{k}\, M}} \{\ [\phi] \mid \phi \in \mathcal{F}\ ;\ \mathfrak{s}' \models a \,\mathsf{k}\, M \rhd \mathsf{K}_a(\phi)\ \} \quad \text{iff}$$

$$\{\ [\phi] \mid M \in \mathcal{M}\ ;\ \mathfrak{s} \models a \,\mathsf{k}\, M\ ;\ \phi \in \mathcal{F}\ ;\ \mathfrak{s} \models a \,\mathsf{k}\, M \rhd \mathsf{K}_a(\phi)\ \} =$$
$$\{\ [\phi] \mid M \in \mathcal{M}\ ;\ \mathfrak{s}' \models a \,\mathsf{k}\, M\ ;\ \phi \in \mathcal{F}\ ;\ \mathfrak{s}' \models a \,\mathsf{k}\, M \rhd \mathsf{K}_a(\phi)\ \} \quad \text{iff}$$

$$\begin{pmatrix} [\text{ for all } M \in \mathcal{M}, \mathfrak{s} \models a \,\mathsf{k}\, M; \text{ for all } \phi \in \mathcal{F}, \mathfrak{s} \models a \,\mathsf{k}\, M \rhd \mathsf{K}_a(\phi) \text{] iff} \\ [\text{ for all } M \in \mathcal{M}, \mathfrak{s}' \models a \,\mathsf{k}\, M; \text{ for all } \phi \in \mathcal{F}, \mathfrak{s}' \models a \,\mathsf{k}\, M \rhd \mathsf{K}_a(\phi) \text{]} \end{pmatrix} \quad \text{iff}$$

$$\begin{pmatrix} [\text{ for all } M \in \mathcal{M}, \mathfrak{s} \models a \,\mathsf{k}\, M; \text{ for all } \phi \in \mathcal{F}, \mathfrak{s} \models \mathsf{K}_a(\phi) \text{] iff} \\ [\text{ for all } M \in \mathcal{M}, \mathfrak{s}' \models a \,\mathsf{k}\, M; \text{ for all } \phi \in \mathcal{F}, \mathfrak{s}' \models \mathsf{K}_a(\phi) \text{]} \end{pmatrix} \quad \text{iff}$$

$$\begin{pmatrix} [\text{ for all } M \in \mathcal{M}, \mathfrak{s} \models \mathsf{K}_a(a \,\mathsf{k}\, M); \text{ for all } \phi \in \mathcal{F}, \mathfrak{s} \models \mathsf{K}_a(\phi) \text{] iff} \\ [\text{ for all } M \in \mathcal{M}, \mathfrak{s}' \models \mathsf{K}_a(a \,\mathsf{k}\, M); \text{ for all } \phi \in \mathcal{F}, \mathfrak{s}' \models \mathsf{K}_a(\phi) \text{]} \end{pmatrix} \quad \text{iff}$$

$$\begin{pmatrix} \text{for all } \phi \in \mathcal{F}, \mathfrak{s} \models \mathsf{K}_a(\phi) \text{ iff} \\ \text{for all } \phi \in \mathcal{F}, \mathfrak{s}' \models \mathsf{K}_a(\phi) \end{pmatrix} \quad \text{iff}$$

$[\text{ for all } \phi \in \mathcal{F}, \mathfrak{s} \models \mathsf{K}_a(\phi) \text{ iff } \mathfrak{s}' \models \mathsf{K}_a(\phi) \text{]}$

Proof of Theorem 6

1. the map $\phi \mapsto \ominus\phi$ (a) order-embeds $[\![\mathfrak{s}]\!]^n_a$ in $[\![\mathfrak{s}]\!]^{n+1}_a$, and (b) is order-continuous:

   (a) $\phi \leq \phi'$ iff $\ominus\phi \leq \ominus\phi'$ because $\models \mathsf{K}_a(\phi) \leftrightarrow \oplus\mathsf{K}_a(\ominus\phi)$

(b) the map $\phi \mapsto \ominus\phi$ is order-continuous because it is order-embedding (thus order-preserving) and $[\![\mathfrak{s}]\!]^n_a$, being topped, satisfies the so-called ascending chain condition [DP02, Page 148]

where $\ominus$ designates the previous-time and $\oplus$ the next-time operator of CPL from linear temporal logic [MP84].

2. $[\![\mathfrak{s}]\!]_a$, $[\![\mathfrak{s}]\!]^n_a$, and $[\![\mathfrak{s}]\!]^*_a$ are topped algebraic $\bigcap$-structures because being (directed) unions of filters they are topped, and closed under intersection and directed unions.

3. $[\![\mathfrak{s}]\!]$, $[\![\mathfrak{s}]\!]^n$, and $[\![\mathfrak{s}]\!]^*$ are pre-CPOs because they are disjoint (and thus lacking a least element) unions of CPOs.

Slogan 22 *Proves should be written like programs — structured.*

Appendix B

Specification Library (Glossary)

Classical propositional and first-order operators

$\top := \texttt{Eve} : \texttt{Adv}$	true
$\bot := \neg\top$	false
$\phi \vee \phi' := \neg(\neg\phi \wedge \neg\phi')$	ϕ or ϕ'
$\phi \to \phi' := \neg\phi \vee \phi'$	if ϕ then ϕ'
$\phi \leftrightarrow \phi' := (\phi \to \phi') \wedge (\phi' \to \phi)$	ϕ if and only if ϕ'
$\exists v(\phi) := \neg\forall v(\neg\phi)$	there is v s.t. ϕ
$\forall(v : \theta)(\phi) := \forall v(v : \theta \to \phi)$	
$\exists(v : \theta)(\phi) := \exists v(v : \theta \wedge \phi)$	

Modal operators

$\mathsf{F}\phi := \neg\mathsf{P}\phi$	it is forbidden that ϕ
$\phi \equiv \phi' := (\phi \trianglerighteq \phi') \wedge (\phi' \trianglerighteq \phi)$	ϕ is epistemically equivalent to ϕ'
$\phi \oplus \phi' := \neg(\neg\phi \otimes \neg\phi')$	ϕ disjunctively separates ϕ'
$\Box\phi := \phi \oplus \bot$	everywhere ϕ
$\Diamond\phi := \neg\Box\neg\phi$	somewhere ϕ
$\phi' \blacktriangleright \phi := \neg(\phi' \triangleright \neg\phi)$	assert ϕ' guarantee ϕ
$* := \ominus\bot$	in the beginning
$\dagger := \oplus\bot$	in the end
$\boxminus\phi := \phi\,\mathsf{S}\,\bot$	so far ϕ
$\diamondminus\phi := \neg\boxminus\neg\phi$	once ϕ
$\mathsf{1}.\phi := \phi \wedge \neg\ominus\diamondminus\phi$	for the first time ϕ
$\boxplus\phi := \phi\,\mathsf{U}\,\bot$	henceforth ϕ
$\diamondplus\phi := \neg\boxplus\neg\phi$	eventually ϕ
$\phi \leq \phi' := (\phi \wedge \diamondplus\phi') \vee (\phi' \wedge \diamondminus\phi)$	ϕ before ϕ'
$\phi \rightsquigarrow \phi' := (\phi \leftrightarrow \diamondplus\phi') \wedge (\phi' \leftrightarrow \diamondminus\phi)$	ϕ correlates ϕ'

Relational symbols

$F = F' \;:=\; F \preccurlyeq F' \wedge F' \preccurlyeq F$	F is equal to F'
$F \prec F' \;:=\; F = F' \wedge \neg\, F \preccurlyeq F'$	F is a strict subterm of F'
$a \;\mathsf{h}\; F \;:=\; \exists v(F \preccurlyeq v \wedge a \;\mathsf{k}\; v)$	a has/possesses F
$a \;\mathsf{tk}\; F \;:=\; a \;\mathsf{h}\; F \wedge \neg\, a \;\mathsf{k}\; F$	a tacitly knows F
$F : \emptyset \;:=\; \bot$	
$F : \mathtt{H}[\theta] \;:=\; \exists(v : \theta)(F = \lceil v \rceil)$	
$F : \mathtt{SC}_{F'}[\theta] \;:=\; \exists(v : \theta)(F = \{\!\lfloor v \rfloor\!\}_{F'})$	
$F : \mathtt{AC}_{p^+}[\theta] \;:=\; \exists(v : \theta)(F = \{\!\lfloor v \rfloor\!\}^+_{p^+})$	
$F : \mathtt{S}_p[\theta] \;:=\; \exists(v : \theta)(F = \{\!\lfloor v \rfloor\!\}^-_p)$	
$F : \mathtt{T}[\theta, \theta'] \;:=\; \exists(v : \theta)\exists(v' : \theta')(F = (v, v'))$	
$F : \theta \cup \theta' \;:=\; F : \theta \vee F : \theta'$	
$F : \theta \cap \theta' \;:=\; F : \theta \wedge F : \theta'$	
$F : \theta \setminus \theta' \;:=\; F : \theta \wedge \neg\, F : \theta'$	
$F : \mathtt{M} \;:=\; \top$	
$F : \mathtt{SC}[\theta] \;:=\; \exists v(F : \mathtt{SC}_v[\theta])$	
$F : \mathtt{AC}[\theta] \;:=\; \exists(v : \mathtt{K}^+)(F : \mathtt{AC}_v[\theta])$	
$F : \mathtt{C}[\theta] \;:=\; F : \mathtt{SC}[\theta] \cup \mathtt{AC}[\theta]$	
$F : \mathtt{S}[\theta] \;:=\; \exists(v : \mathtt{K}^-)(F : \mathtt{S}_v[\theta])$	
$\theta \sqsubseteq \theta' \;:=\; \forall(v : \theta)(v : \theta')$	θ is a subtype of θ'
$\theta = \theta' \;:=\; \theta \sqsubseteq \theta' \wedge \theta' \sqsubseteq \theta$	
$\theta \sqsubset \theta' \;:=\; \theta \sqsubseteq \theta' \wedge \theta' \neq \theta$	
$F \;\mathfrak{k}\!\vdash\; F' \;:=\; \exists v(\{\!\lfloor v \rfloor\!\}_F \preccurlyeq F')$	
$p^+ \;\mathfrak{k}\!\vdash^+\; F \;:=\; \exists v(\{\!\lfloor v \rfloor\!\}^+_{p^+} \preccurlyeq F)$	
$p \;\mathfrak{k}\!\vdash^-\; F \;:=\; \exists v(\{\!\lfloor v \rfloor\!\}^-_p \preccurlyeq F)$	
$F \;\mathfrak{k}\!\vdash^*\; F' \;:=\; F \;\mathfrak{k}\!\vdash\; F' \vee F \;\mathfrak{k}\!\vdash^+\; F' \vee F \;\mathfrak{k}\!\vdash^-\; F'$	F is operational in F'
$F \multimap F' \;:=\; \exists v\exists v'(F \preccurlyeq v \wedge \{\!\lfloor v \rfloor\!\}_{v'} \preccurlyeq F')$	
$F \multimap^+ F' \;:=\; \exists v\exists(p^+ : \mathtt{K}^+)(F \preccurlyeq v \wedge \{\!\lfloor v \rfloor\!\}^+_{p^+} \preccurlyeq F')$	
$F \multimap^- F' \;:=\; \exists v\exists(p : \mathtt{K}^-)(F \preccurlyeq v \wedge \{\!\lfloor v \rfloor\!\}^-_p \preccurlyeq F')$	
$F \multimap^* F' \;:=\; F \multimap F' \vee F \multimap^+ F' \vee F \multimap^- F'$	F is guarded in F'
$k \;\mathsf{sk}\; a \;:=\; \exists b\exists o(b \circlearrowright k.o \wedge a \preccurlyeq o)$	k is a symmetric key for a
$k \;\mathsf{sk}^1\; a \;:=\; k \;\mathsf{sk}\; a \wedge k : \mathtt{K}^1$	k is a session/short-term key for a
$k \;\mathsf{sk}^\infty\; a \;:=\; k \;\mathsf{sk}\; a \wedge k : \mathtt{K}^\infty$	k is a long-term key for a
$p \;\mathsf{prk}\; a \;:=\; \exists b\exists o(b \circlearrowright p.o \wedge a \preccurlyeq o)$	p is a private key for a
$n \;\mathsf{puk}\; a \;:=\; \exists v(v^+ = n \wedge v \;\mathsf{prk}\; a)$	n is a public key for a
$n \;\mathsf{ck}\; a \;:=\; n \;\mathsf{sk}\; a \vee n \;\mathsf{prk}\; a$	n is a confidential key for a

Appendix C

Bibliography

[AB05] M. Abadi and B. Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM*, 52(1), 2005. 3.1.2.2

[Aba00] M. Abadi. Security protocols and their properties. In *Foundations of Secure Computation*. IOS Press, 2000. 3.1.2.2

[ABV01] R. Accorsi, D. Basin, and L. Viganò. Towards an awareness-based semantics for security protocol analysis. In *Proceedings of the Post-CAV Workshop on Logical Aspects of Cryptographic Protocol Verification*, 2001. 3.1.2.2

[ADM03] K. Adi, M. Debbabi, and M. Mejri. A new logic for electronic commerce protocols. *Theoretical Computer Science*, 291(3), 2003. 3.1.2.2

[AF01] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, 2001. 3.1.2.2

[AG99] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The Spi-calculus. *Information and Computation*, 148(1), 1999. 3.1.2.2

[AM01] L. C. Aiello and F. Massacci. Verifying security protocols as planning in logic programming. *ACM Transactions on Computational Logic*, 2(4), 2001. 3.1.2.2

[AN96a] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1), 1996. 4.3, 3, 1

[AN96b] R. Anderson and R. Needham. Programming Satan's computer. In *Computer Science Today: Recent Trends and Developments*, volume 1000 of *LNCS*. Springer-Verlag, 1996. 3.1.1, 3.1.2.2

[AN05] S. Artemov and E. Nogina. Introducing justification into epistemic logic. *Journal of Logic and Computation*, 15(6), 2005. 3.3.4

[AR02] M. Abadi and Ph. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2), 2002. 4

[Art95] S. Artemov. Operational modal logic. Technical Report MSI 95-29, Cornell University, 1995. 3.3.4

[Art01] S. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1), 2001. 3.3.4

[ASW00] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4), 2000. 3.3.4

[AT91] M. Abadi and M. R. Tuttle. A semantics for a logic of authentication. In *Proceedings of the ACM Symposium of Principles of Distributed Computing*, 1991. 3.2.3.4

[Bal01] A. Baltag. Logics for insecure communication. In *Proceedings of the Conference on Theoretical Aspects of Rationality and Knowledge*, 2001. 3.1.2.2

[BAN90] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1), 1990. 3.1.1, 3.1.2.2, 3.3

[Bar99] J. Barwise, editor. *Handbook of Mathematical Logic*. Elsevier, 1977 (1999). 2.4.1

[BC93] P. Bieber and F. Cuppens. *Deontic Logic in Computer Science: Normative System Specification*, chapter Expression of Confidentiality Policies with Deontic Logic. John Wiley & Sons, 1993. 3.1.2.2, 3.1.2.3

[BD04] M. Bozzano and G. Delzanno. Automatic verification of secrecy properties for linear logic specifications of cryptographic protocols. *Journal of Symbolic Computation*, 38(5), 2004. 3.1.2.2

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001. 6

[BEL05] L. Bozga, C. Ene, and Y. Lakhnech. A symbolic decision procedure for cryptographic protocols with time stamps. *The Journal of Logic and Algebraic Programming*, 65, 2005. 3.4.1, 4.2

[BGK06] J. Borgström, O. Grinchtein, and S. Kramer. Timed Calculus of Cryptographic Communication. In *Proceedings of the Workshop on Formal Aspects in Security and Trust*, 2006. 2.5.4

[BGPS00] M. Benerecetti, F. Giunchiglia, M. Panti, and L. Spalazzi. A logic of belief and a model checking algorithm for security protocols. In *Proceedings of the Conference on Formal Description Techniques for Distributed Systems and Communication Protocols*, 2000. 3.1.2.2

[Bie90] P. Bieber. A logic of communication in hostile environment. In *Proceedings of the IEEE Computer Security Foundations Workshop*, 1990. 3.1.2.2

[BKN06] J. Borgström, S. Kramer, and U. Nestmann. Calculus of Cryptographic Communication. In *Proceedings of the LICS-Affiliated Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis*, 2006. 2.5.4, 4.1.1

[BM03] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003. 2.4.2, 3.1.2.2, 3.3.3, 4.3.1

[BMN00] P. Bellini, R. Mattolini, and P. Nesi. Temporal logics for real-time system specification. *ACM Computing Surveys*, 32(1), 2000. 3.4.1

[BPS01] J. A. Bergstra, A. Ponse, and S. A. Smolka, editors. *Handbook of Process Algebra*. Elsevier, 2001. 3.1.2.3

[BPW03] M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. In *Proceedings of the ACM Conference on Computer and Communication Security*, 2003. 3.1.2.3

[BvBW07] P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier, 2007. 2.3.2.1, 2.4.1, 3.2.3.1, 3.3.4, 26

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 2001. 3.1.2.3, 28

[CD05a] M. Cohen and M. Dam. A completeness result for BAN logic. In *Proceedings of the Workshop on Methods for Modalities*, 2005. 3.2.3.4

[CD05b] M. Cohen and M. Dam. Logical omniscience in the semantics of BAN logic. In *Proceedings of the LICS-affiliated Workshop on the Foundations of Computer Security*, 2005. 3.2.3.4

[CGP99] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999. 3.1.2.3

[CJM98] E. Clarke, S. Jha, and W. Marrero. A machine checkable logic of knowledge for specifying security properties of electronic commerce protocols. In *Proceedings of the LICS-Affiliated Workshop on Formal Methods & Security Protocols*, 1998. 3.1.2.2

[CKW07] V. Cortier, R. Küsters, and B. Warinschi. A cryptographic model for branching time security properties — the case of contract signing protocols. Technical report, ETH Zurich, 2007. 3.3.4

[CLC04] H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. *Science of Computer Programming*, 50(1–3), 2004. 2.4.2.3

[Coh03] E. Cohen. First-order verification of cryptographic protocols. *Journal of Computer Security*, 11(2), 2003. 3.1.2.2

[CS97] T. Coffey and P. Saidha. Logic for verifying public-key cryptographic protocols. In *IEE Proceedings — Computers and Digital Techniques*, 1997. 3.1.2.2

[CVB05] C. Caleiro, L. Viganò, and D. Basin. Relating strand spaces and distributed temporal logic for security protocol analysis. *Logic Journal of the Interest Group in Pure and Applied Logic*, 13, 2005. 3.1.2.2, 3.2.3.5

[Dam89] M. F. Dam. *Relevance Logic and Concurrent Composition*. PhD thesis, University of Edinburgh, 1989. 3.1.2.3

[DDM+05] A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In *Proceedings of the EATCS International Colloquium on Automata, Languages and Programming*, 2005. 5.1, 5.1

[DDMP05] A. Datta, A. Derek, J.C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13, 2005. 5.1

[dG95] Ph. de Groote, editor. *The Curry-Howard Isomorphism*. Number 8 in Cahiers du centre de logique. Academia-Erasme, Louvain-la-Neuve (Belgique), 1995. 3.3.4

[DGMF04] C. Dixon, M.-C. F. Gago, and W. van der Hoek M. Fisher. Using temporal logics of knowledge in the formal verification of security protocols. In *Proceedings of the International Symposium on Temporal Representation and Reasoning*, 2004. 3.1.2.2

[Dij72] E. W. Dijkstra. The humble programmer. *Communications of the ACM*, 15(10), 1972. 3.1.1

[DMP03] N. Durgin, J. C. Mitchell, and D. Pavlovic. A compositional logic for proving security properties of protocols. *Journal of Computer Security*, 11(4), 2003. 3.1.2.2, 5.1

[DP02] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990 (2002). 2, 4.3.2, 1b

[DY83] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(12), 1983. 1

[ES00] N. Evans and S. Schneider. Analysing time-dependent security properties in CSP using PVS. In *Proceedings of the European Symposium on Research in Computer Security*, 2000. 3.4.1, 4.2, 4.2

[FHG99] F. J. Th. Fábrega, J. C. Herzog, and J. D. Guttman. Strand spaces: proving security protocols correct. *Journal of Computer Security*, 7(2-3), 1999. 3.2.3.5

[FHJ02] U. Frendrup, H. Hüttel, and J. N. Jensen. Modal logics for cryptographic processes. In *Electronic Notes in Theoretical Computer Science*, volume 68, 2002. 3.1.2.2

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995. 3.1.2.3, 3

[Fit96] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer-Verlag, 1996. 3.1.2.3

[Gib04] R. F. Gibson, Jr., editor. *Quintessence: Basic Readings from the Philosophy of W. V. Quine*. The Belknap Press of Harvard University Press, 2004. 1, 1, 2.1.3, 4

[GJM99] J. A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Proceedings of CRYPTO*, 1999. 3.3.4

[GKWZ03] D.M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications*. Elsevier, 2003. 6

[GL91] O. Grumberg and D. E. Long. Model checking and modular verification. In *Proceedings of CONCUR*, 1991. 28

[GLL01] S. Gnesi, D. Latella, and G. Lenzini. A BRUTUS logic for the Spi-Calculus. In *Proceedings of the IFIP Workshop on Issues in the Theory of Security*, 2001. 3.1.2.2

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28(2), 1984. 5.3.1, 1

[GM95] J. W. Gray and J. D. McLean. Using Temporal Logic to Specify and Verify Cryptographic Protocols (progress report). In *Proceedings of the IEEE Computer Security Foundations Workshop*, 1995. 3.1.2.2

[GM04] R. Gorrieri and F. Martinelli. A simple framework for real-time cryptographic protocol analysis with compositional proof rules. *Science of Computer Programming*, 50(1–3), 2004. 3.4.1, 4.2

[GMP92] J. Glasgow, G. Macewen, and P. Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3), 1992. 3.1.2.2

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1), 1989. 7

[Gol01] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001. 1, 2.4.2, 3.1.2.1, 5.3.1, 2, 3, 4, 6

[Gol04] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004. 2.4.2, 3.1.2.1, 5.3.2, 1, 2, 10

[Gol05] O. Goldreich. *Foundations of Cryptography — A Primer*. now Publishers Inc., 2005. 5.3.1, 3

[Gon92] L. Gong. A security risk of depending on synchronized clocks. *ACM SIGOPS Operating Systems Review*, 26(1), 1992. 3.4.1

[Gro92] A. J. Grove. Semantics for knowledge and communication. In *Proceedings of the Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 1992. 4.3

[GSW06] D. Goldin, S. A. Smolka, and P. Wegner, editors. *Interactive Computation: The New Paradigm*. Springer-Verlag, 2006. 3.2.2, 8

[Hal03] J. Halpern. *Reasoning about Uncertainty*. MIT Press, 2003. 5.1.1

[HJ04] Ch. Haack and A. Jeffrey. Pattern-matching Spi-calculus. In *Proceedings of the Workshop on Formal Aspects in Security and Trust*, 2004. 4.1.2

[HJ05] Ch. Haack and A. Jeffrey. Timed Spi-calculus with types for secrecy and authenticity. In *Proceedings of CONCUR*, 2005. 3.4.1, 4.2

[HO02] J. Halpern and K. O'Neill. Secrecy in multi-agent systems. In *Proceedings of the IEEE Computer Security Foundations Workshop*, 2002. 3.1.2.2, 3.3

[HS04a] M. R. Hansen and R. Sharp. Using interval logics for temporal analysis of security protocols. In *Proceedings of the ACM Workshop on Formal Methods in Security Engineering*, 2004. 3.4.1

[HS04b] D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12, 2004. 11, 4.1.3

[IK06] R. Impagliazzo and B. M. Kapron. Logics for reasoning about cryptographic constructions. *Journal of Computer and Systems Sciences*, 72(2), 2006. 3.1.2.2, 5.1, 5.1

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the ACM Symposium on the Theory of Computation*, 1988. 9

[KM99] M. Kudo and A. Mathuria. An extended logic for analyzing timed-release public-key protocols. In *Proceedings of the Conference on Information, Communications and Signal Processing*, 1999. 3.4.1

[KR04] V. Kuncak and M. Rinard. On spatial conjunction as second-order logic. Technical Report 970, MIT CSAIL, 2004. 19

[Kra03] S. Kramer. A language and a notion of truth for cryptographic properties, 2003. Short presentation at LICS. 2.5.4

[Kra04] S. Kramer. Cryptographic Protocol Logic. In *Proceedings of the LICS/ICALP-Affiliated Workshop on Foundations of Computer Security*, 2004. 2.5.4

[Kra06a] S. Kramer. Logical abstractions for protocol engineering, 2006. presented at the Workshop on Models for Cryptographic Protocols. 2.5.4

[Kra06b] S. Kramer. Timed Cryptographic Protocol Logic, 2006. presented at the Nordic Workshop on Programming Theory. 2.5.4

[Kra07a] S. Kramer. The meaning of a cryptographic message via hypothetical knowledge and provability. In *Proceedings of the Workshop on Logic, Rationality and Interaction*, 2007. 2.5.4

[Kra07b] S. Kramer. The meaning of a cryptographic message via hypothetical knowledge and provability. In *Proceedings of the Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis*, 2007. 2.5.4

[Kraar] S. Kramer. Cryptographic Protocol Logic: Satisfaction for (timed) Dolev-Yao cryptography. *Journal of Logic and Algebraic Programming*, to appear. 2.5.4

[KS06] M. Kurkowski and M. Srebrny. A quantifier-free first-order knowledge logic of authentication. *Fundamenta Informaticæ*, 72, 2006. 3.1.2.2

[KSW98] J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Proceedings of the Workshop on Security Protocols*, 1998. 4

[LA05] G. Lowe and M. Auty. On a calculus for security protocol development. Technical report, Oxford University, 2005. 3.1.2.2

[Lam05] L. Lamport. Real time is really simple. Technical Report MSR-TR-2005-30, Microsoft Research, 2005. 3.4, 4.2, 5.1

[Low97] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the Computer Security Foundations Workshop*, 1997. 3.3.5.1

[LV97] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, second edition, 1997. 15, 17

[LW06] A. Lomuscio and B. Woźna. A complete and decidable security-specialised logic and its application to the TESLA protocol. In *Proceedings of the Conference on Autonomous Agents and Multiagent Systems*, 2006. 3.1.2.2

[McL99] J. McLean. Twenty years of formal methods. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1999. 3.1.1

[MDW94] J.-J. Ch. Meyer, F. P. M. Dignum, and R. J. Wieringa. The Paradoxes of Deontic Logic Revisited: A Computer Science Perspective. Or: Should computer scientists be bothered by the concerns of philosophers ? Technical Report UU-CS-1994-38, Utrecht University, 1994. 3.2.3.3

[Mea03] C. Meadows. Ordering from Satan's menu: a survey of requirements specification for formal analysis of cryptographic protocols. *Science of Computer Programming*, 50(3–22), 2003. 3.1.1

[Mil06] D. Miller. Representing and reasoning with operational semantics. In *Proceedings of the Joint International Conference on Automated Reasoning*, 2006. invited paper. 3.1.2.2

[Mos05] L. S. Moss. *Mathematical Problems from Applied Logic I: Logics for the XXIst Century*, chapter Applied Logic: A Manifesto. Springer-Verlag, 2005. 1

[MP84] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1984. 3.1.2.3, 3.1.2.3, 1

[MRST06] J. C. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. *Theoretical Computer Science*, 353(1–3), 2006. 3.1.2.2, 5.1

[MvOV96] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. 2.4.2, 3.1.1, 3.1.2.1, 3.1.2.3, 3.3.2, 3.3.3, 5.3.2, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24

[Nut97] D. Nute, editor. *Defeasible Denotic Logic*, volume 263 of *Synthese Library*. Kluwer, 1997. 3.2.3.3

[Pau98] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1), 1998. 3.1.2.2

[PR03] R. Parikh and R. Ramanujam. A knowledge based semantics of messages. *Journal of Logic, Language and Information*, 12, 2003. 4.3, 4.3.1

[Rab81] M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981. 9

[Riv90] R. L. Rivest. Cryptography. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*. Elsevier, 1990. 2

[Rog04] Ph. Rogaway. On the role of definitions in and beyond cryptography. In *Proceedings of the Asian Computing Science Conference*, 2004. 2, 3.1.1, 5.3

[RSG+00] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley, 2000. 3.1.2.2

[Sch99] S. Schneider. *Concurrent and Real-Time Systems*. Wiley, 1999. 3.4.1

[Sel01] P. Selinger. Models for an adversary-centric protocol logic. In *Proceedings of the Post-CAV Workshop on Logical Aspects of Cryptographic Protocol Verification*, 2001. 3.1.2.2

[Sho04] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/. 2.1.1

[Sla06] B. H. Slater. Epsilon calculi. *Logic Journal of the Interest Group in Pure and Applied Logic*, 14(4), 2006. 2

[SP03] E. Sumii and B. C. Pierce. Logical relations for encryption. *Journal of Computer Security*, 11(4), 2003. 3.1.2.2

[SS99] P. F. Syverson and S. G. Stubblebine. Group principals and the formalization of anonymity. In *Proceedings of the World Congress On Formal Methods In The Development Of Computing Systems*, 1999. 3.1.2.2, 3.3

[SvO96] P. F. Syverson and P. C. van Oorschot. A unified cryptographic protocol logic. CHACS 5540-227, Naval Research Laboratory, Washington D.C., USA, 1996. 3.1.2.2

[Var01] M. Y. Vardi. Branching vs. linear time: Final showdown. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer-Verlag, 2001. 3.3.4

[Vau05] S. Vaudenay. *A Classical Introduction to Cryptography: Applciations to Communications Security*. Springer, 2005. 6

[vL90] J. van Leeuwen, editor. *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics. Elsevier, 1990. 2.4.1

[vT05] H. C. A. van Tilborg, editor. *Encyclopedia of Cryptography and Security*. Springer, 2005. 2.4.2

[Wan04] F. Wang. Formal verification of timed systems: A survey and perspective. *Proceedings of the IEEE*, 92(8), 2004. 3.4.1, 3.4.1, 3.4.1

[Wit75] L. Wittgenstein. *Tractatus Logico-Philosophicus*. Routledge, English edition, 1961, 1975. 4.3.2

[Zah03] J. Zahnd. *Logique élémentaire. Cours de base pour informaticiens*. PPUR, 1998 (2003). 1

[ZH04] C. Zhou and M. R. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems*. Springer-Verlag, 2004. 3.4.3

[ZHR91] C. Zhou, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5), 1991. 3.4.3

[ZV01] Y. Zhang and V. Varadharajan. A logic for modeling the dynamics of beliefs in cryptographic protocols. In *Proceedings of the Australasian Conference on Computer Science*, 2001. 3.1.2.2

Appendix D

Index

Curriculum Vitæ

Education

2002–2007 **Ph.D.** from **Ecole Polytechnique Fédérale de Lausanne** (EPFL)

1995–2001 **M.Sc.** in **Computer Science** from EPFL

- Undergraduate studies (2 years)
- Sabbatical year (SAirGroup, Zurich)
- Exchange student (1 year) at the **University of Granada** (E)
- Termination of graduate studies at EPFL (1 year)
- Diploma project (4 months) at **Politecnico di Milano** (I)
  Topic: *A Formal Specification for a Real-Time Train Controller* (**embedded systems**)

  Project (academia-to-industry consulting)
  supervisor: Prof. Dino Mandrioli
  client: **Ferrovie dello Stato**, the Italian railway company.

1993–1995 **Swiss Federal Certificate of Maturity** as a free candidate. Working student.

1992–1993 Language studies in France and England (with diplomas)

1988–1992 **Apprenticeship in electronics** (Federal Certificate of Capacity, letter of distinction)
Professional Maturity in parallel (Certificate of Examination).

Work experience

2005–2007 **EPFL**: Research Assistant of Prof. Thomas A. Henzinger

2004 **EPFL**: Lecturer (ad interim) for the course *Elementary Logic* (own initiative)

2001–2005 **EPFL**: Research Assistant of Prof. Uwe Nestmann

1999 **IBM Research**, Zurich: Summer Student (3 months)
Task: prototype implementation of a method for inter-enterprise role-based authorisation (**e-commerce**).

1997–1998 **SAirGroup**, Zurich: Computer Programmer for a fixed time (10 months)
Earning of a premium for exceptional services:

- Migration of logistic software for GateGourmet (SAirGroup)
- Design of a brochure for graduate staff recruiting.

1992–1996 **Hewlett Packard**, Zurich: Promoter in French-speaking Switzerland. Part-time activity.