Logical Concepts in Cryptography[1]

Simon Kramer
Ecole Polytechnique Fédérale de Lausanne (EPFL)
simon.kramer@a3.epfl.ch

January 25, 2007

[1] Chapter 1 and Appendices A and B constitute an invited submission (January 15, 2007) to the Journal of Logic and Algebraic Programming with the title "Cryptographic Protocol Logic: Satisfaction for (Timed) Dolev-Yao Cryptography".

Abstract

This report is about the exploration of logical concepts in cryptography and their linguistic abstraction and model-theoretic combination in a logical system, called CPL (for *Cryptographic Protocol Logic*). The report focuses 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 knowledge, norms, provability, space, and time. SECONDARY CONCEPTS: 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 (logics) and model-based (process algebra) formalisms. We illustrate the expressiveness of CPL on representative *requirements engineering* case studies.

Addendum I. 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 really simple. 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.

Addendum II. In Chapter 2, 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. The argument of this chapter is that in the light of logic, adding PP to a (property-based) 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.

Keywords applied formal logic, cryptographic protocols

Contents

List of Tables

Chapter 1

(Timed) Dolev-Yao Cryptography

1.1 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.

1.1.1 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 1 *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." [AN96]. Indeed, designing a correct cryptographic protocol (i.e., "programming Satan's computer" [AN96]), is extremely more difficult than designing a correct, ordinary computer program (i.e., "programming Murphy's [computer]" [AN96]) of the same size. In fact, at the end of the 1980s,

[1] rather than shared memory

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 (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 2 *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 3 *Cryptographic protocol correctness: a killer application for formal methods.*

1.1.2 Topical context

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

[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."

requirements engineering that adequately abstracts "hard-core" mathematical concepts.

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 macro-defined, in terms of the primitives of the candidate. By intuitive specifications we mean that the conceptual dimensions of the 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.

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]). 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." [AN96]. 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 programming 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 1.3 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 1.2.3). (We recall that a *weak* paradox 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 rely on knowledge, i.e.,

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

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

Slogan 4 *Belief can be used to show the presence of attacks, but, as opposed to knowledge, never to show their absence.*

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 of operational aspects. 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.

Requirements engineering—CPL

Our goal is to supply a formal *synthesis* of (mono-dimensional) concepts in a single, poly-dimensional[5] modal logic, namely CPL[6], 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 5 *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 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:

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

[5] 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."

[6] a preliminary, now outdated version of CPL appeared in the informal proceedings of [Kra04]

[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*

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

Slogan 6 *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 1.2.3). 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 [BKN06, BGK06] whose reduction constraints *can* moreover be checked via CPL-satisfaction; and second, CPL's epistemic 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.

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

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

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). 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].

1.2 Logic

1.2.1 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 1.1. 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 $v \in \mathcal{V}$.*

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 encrypted plain text is by definition cryptographic information whose knowledge is intended to be limited to

Table 1.1: Message language

$$
\begin{array}{lcll}
M & ::= & n & \text{(names, i.e., logical constants)} \\
 & | & \blacksquare & \text{(the abstract message)} \\
 & | & p^{+} & \text{(public keys)} \\
 & | & \lceil M \rceil & \text{(message hashes)} \\
 & | & \{\!|M|\!\}_{M} & \text{(symmetric message ciphers)} \\
 & | & \{\!|M|\!\}^{+}_{p^{+}} & \text{(asymmmetric message ciphers)} \\
 & | & \{\!|M|\!\}^{-}_{p} & \text{(signed messages)} \\
 & | & (M, M) & \text{(message tuples)}
\end{array}
$$

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 $\{\!|M|\!\}_{k}$ but not the shared key k or given $\{\!|M|\!\}^{+}_{p^{+}}$ 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 con-*

structors displayed in Table 1.2. There, β denotes basic, α action, and δ data formulae; and o denotes tuples of agent names (key owners).

Table 1.2: Predicate language

$$\begin{array}{rcl}
\phi, \phi' & ::= & \beta \mid \neg\phi \mid \phi \wedge \phi' \mid \forall v(\phi) \\
 & \mid & \underbrace{\mathsf{P}\phi}_{\text{norms}} \mid \underbrace{\mathsf{K}_a(\phi) \mid \phi \supseteq \phi'}_{\text{knowledge}} \mid \underbrace{\phi \otimes \phi' \mid \phi \rhd \phi'}_{\text{space}} \mid \underbrace{\phi \,\mathsf{S}\, \phi' \mid \ominus\phi \mid \bigoplus\phi \mid \phi \,\mathsf{U}\, \phi'}_{\text{time}} \\
\beta, \beta' & ::= & \alpha \mid \delta \\
\alpha, \alpha' & ::= & a \circlearrowright n.o \mid \underbrace{a \xrightarrow[\mathtt{E\not{v}e}]{F} b \mid a \xleftarrow[\mathtt{E\not{v}e}]{F} b}_{\text{private comm.}} \mid \underbrace{a \xrightarrow[\mathtt{Eve}]{F} b \mid a \xleftarrow[\mathtt{Eve}]{} F}_{\text{public comm.}} \\
\delta, \delta' & ::= & n : \sigma \mid a \,\mathsf{k}\, F \mid F \preccurlyeq F' \mid a@x
\end{array}$$

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*[11].

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", $\supseteq$ "epistemically implies", $\otimes$ "conjunctively separates", $\rhd$ "assume—guarantee", S "since", $\ominus$ "previous", $\bigoplus$ "next", U "until", $a \circlearrowright n.o$ "a freshly generated the name n for owner(s) o", $a \xrightarrow[\mathtt{E\not{v}e}]{F} b$ "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", $a \xleftarrow[\mathtt{E\not{v}e}]{F} b$ "a securely received F as such from b", $a \xrightarrow[\mathtt{Eve}]{F} b$ "a insecurely (i.e., over some public channel) sent off F as such to b", $a \xleftarrow[\mathtt{Eve}]{} F$ "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*[12] 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.

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

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

The conditional $\phi \supseteq \phi'$ 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 plain text *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 A).

1.2.2 Semantics

Our definition of satisfaction[13] 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].

For the purpose of this paper, we presuppose a notion of execution, *for example* [BKN06], $\longrightarrow \subseteq (\mathcal{H} \times \mathcal{P})^2$ (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, 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 $\mathtt{Eve}$. 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 $\mathfrak{h} \ ::= \ \epsilon \ \big| \ \mathfrak{h} \cdot \varepsilon$, where ϵ designates the empty protocol history.

We define satisfaction in a functional style on the structure of formulae. Satisfaction employs *complex* (and thus multiple[14]) truth values. Truth values are complex in the sense that they are tuples of a simple truth value (i.e., 'true'

[13] the concept was invented by Tarski

[14] multi-valued logic was invented by Post

or ‘false’) and a set of those events (the evidence/witnesses) that corroborate that simple truth.

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[15] 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 1.4). 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 1.3)*[16]
- $\circ$ *designates concatenation of histories conserving uniqueness of events*
- $\Sigma := \exists(k : \mathtt{CK})(\mathtt{Eve}\ \mathsf{k}\ k \land \neg\, k\ \mathsf{ck}\ \mathtt{Eve})$ *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 permutation, i.e.,* $\mathfrak{h} \equiv \mathfrak{h}'$ *:iff* $|\mathfrak{h}| = |\mathfrak{h}'|$ *and* $\dot{\mathfrak{h}} = \dot{\mathfrak{h}}'$*.*

The permission modality is primitive rather than macro-defined 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 or the ones for probabilistic

[15] 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)

[16] we could easily account for individual knowledge *modulo an equational theory* of cryptographic 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 $\dfrac{\mathfrak{h} \vdash^{\mathcal{E}}_a M \quad M \equiv M'}{\mathfrak{h} \vdash^{\mathcal{E}}_a M'}$

Table 1.3: 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***

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

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

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

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

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

polynomial-time settings (cf. Section 1.4 and Chapter 2, respectively). 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 [BKN06].

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 ■.

Definition 5 (Cryptographic parsing) *The cryptographic parsing function* $(\!|\cdot|\!)_a^{\mathfrak{h}}$ *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 1.5.*

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 $(\!|\,\{\!|M|\!\}_k\,|\!)_a^{\mathfrak{h}}=\blacksquare=(\!|\,\{\!|M'|\!\}_k\,|\!)_a^{\mathfrak{h}'}$. That is, two different plaintexts (M and M') encrypted under the same symmetric key (k) are parsed to the same (abstract) message (■), if 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 2), encryption is probabilistic anyway, which has precisely the effect of rendering the above ciphers (computationally) indistinguishable to a parsing agent.

Table 1.4: Truth denotation

$$\begin{array}{rcl}
\llbracket a \circlearrowright n.o \rrbracket^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{x}} \{\mathtt{N}(a,x,n,o)\} \cap \dot{\mathfrak{h}} \\
\llbracket a \underset{\mathtt{E\not{v}e}}{\xrightarrow{M}} b \rrbracket^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{x}} \{\mathtt{sO}(a,x,M,b)\} \cap \dot{\mathfrak{h}} \\
\llbracket a \underset{\mathtt{E\not{v}e}}{\xleftarrow{M}} b \rrbracket^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{x}} \{\mathtt{sI}(a,x,M,b)\} \cap \dot{\mathfrak{h}} \\
\llbracket a \underset{\mathtt{Eve}}{\xrightarrow{M}} b \rrbracket^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{x}} \{\mathtt{O}(a,x,M,b)\} \cap \dot{\mathfrak{h}} \\
\llbracket a \underset{\mathtt{Eve}}{\leftarrow} M \rrbracket^i_{\mathfrak{p}} & := & (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \cup_{x \in \boldsymbol{x}} \{\mathtt{I}(a,x,M)\} \cap \dot{\mathfrak{h}} \\
\llbracket n : \sigma \rrbracket^i_{\mathfrak{p}} & := & (n \text{ has type } \sigma, \emptyset) \\
\llbracket a \mathsf{~k~} M \rrbracket^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\ \} \\
\llbracket M \preccurlyeq M' \rrbracket^i_{\mathfrak{p}} & := & (M \text{ is a subterm of } M', \emptyset) \\
\llbracket a@x \rrbracket^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) \\
\llbracket \neg\phi \rrbracket^i_{\mathfrak{p}} & := & (\text{not } \mathsf{v}_\phi, \dot{\mathfrak{h}} \setminus \mathcal{E}_\phi) \quad \text{where } \llbracket \phi \rrbracket^i_{\mathfrak{p}} = (\mathsf{v}_\phi, \mathcal{E}_\phi) \\
\llbracket \phi \wedge \phi' \rrbracket^i_{\mathfrak{p}} & := & (\mathsf{v}_\phi \text{ and } \mathsf{v}_{\phi'}, \mathcal{E}_\phi \cup \mathcal{E}_{\phi'}) \quad \text{where } \begin{array}{l} \llbracket \phi \rrbracket^i_{\mathfrak{p}} = (\mathsf{v}_\phi, \mathcal{E}_\phi) \text{ and} \\ \llbracket \phi' \rrbracket^i_{\mathfrak{p}} = (\mathsf{v}_{\phi'}, \mathcal{E}_{\phi'}) \end{array} \\
\llbracket \forall v(\phi) \rrbracket^i_{\mathfrak{p}} & := & (\text{for all } M \in \mathcal{M},\, \mathsf{v}_M, \bigcup_{M \in \mathcal{M}} \mathcal{E}_M) \quad \text{where } \llbracket \{{}^{M}\!/_{v}\}\phi \rrbracket^i_{\mathfrak{p}} = (\mathsf{v}_M, \mathcal{E}_M) \\
\llbracket \mathsf{P}\phi \rrbracket^i_{\mathfrak{p}} & := & \llbracket \phi \rhd \boxplus(\Sigma \to (\Sigma \not\supseteq \phi)) \rrbracket^i_{\mathfrak{p}} \\
\llbracket \mathsf{K}_a(\phi) \rrbracket^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} \\ ((\!| \mathfrak{s} |\!)^{\mathfrak{h}}_a, (\!| \phi |\!)^{\mathfrak{h}}_a) & \text{otherwise.} \end{cases} \\
\llbracket \phi \supseteq \phi' \rrbracket^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} \llbracket \phi \rrbracket^i_{\mathfrak{p}} = (\mathsf{v}_\phi, \mathcal{E}_\phi) \text{ and} \\ \llbracket \phi' \rrbracket^i_{\mathfrak{p}} = (\mathsf{v}_{\phi'}, \mathcal{E}_{\phi'}) \end{array} \\
\llbracket \phi \otimes \phi' \rrbracket^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 \parallel\!| 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'}) \\
\llbracket \phi \rhd \phi' \rrbracket^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 \parallel\!| P) \models_{\mathcal{E}''} \phi', \bigcup \mathcal{E}'' \cup \bigcup \mathcal{E}') \\
\llbracket \phi \mathsf{~S~} \phi' \rrbracket^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, \\
 & & \bigcup_j \mathcal{E}_j \cup \mathcal{E}_k) \quad \text{where } \llbracket \phi \rrbracket^j_{\mathfrak{p}} = (\mathsf{v}_j, \mathcal{E}_j) \text{ and } \llbracket \phi' \rrbracket^k_{\mathfrak{p}} = (\mathsf{v}_k, \mathcal{E}_k) \\
\llbracket \ominus \phi \rrbracket^i_{\mathfrak{p}} & := & \begin{cases} \llbracket \phi \rrbracket^{i-1}_{\mathfrak{p}} & \text{if } i > 0, \text{ and} \\ (\text{false}, \emptyset) & \text{otherwise.} \end{cases} \\
\llbracket \oplus \phi \rrbracket^i_{\mathfrak{p}} & := & \begin{cases} \llbracket \phi \rrbracket^{i+1}_{\mathfrak{p}} & \text{if } i < |\mathfrak{p}| - 1, \text{ and} \\ (\text{false}, \emptyset) & \text{otherwise.} \end{cases} \\
\llbracket \phi \mathsf{~U~} \phi' \rrbracket^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, \\
 & & \bigcup_j \mathcal{E}_j \cup \mathcal{E}_k) \quad \text{where } \llbracket \phi \rrbracket^j_{\mathfrak{p}} = (\mathsf{v}_j, \mathcal{E}_j) \text{ and } \llbracket \phi' \rrbracket^k_{\mathfrak{p}} = (\mathsf{v}_k, \mathcal{E}_k)
\end{array}$$

Table 1.5: Parsing cryptographic messages

$$
\begin{aligned}
\llparenthesis \lceil M \rceil \rrparenthesis_a^{\mathfrak{h}} &:= \begin{cases} \lceil \llparenthesis M \rrparenthesis_a^{\mathfrak{h}} \rceil & \text{if } \mathfrak{h} \models a \mathrel{\mathsf{k}} M, \text{ and} \\ \blacksquare & \text{otherwise.} \end{cases} \\
\llparenthesis \{\!|M|\!\}_{M'} \rrparenthesis_a^{\mathfrak{h}} &:= \begin{cases} \{\!|\llparenthesis M \rrparenthesis_a^{\mathfrak{h}}|\!\}_{\llparenthesis M' \rrparenthesis_a^{\mathfrak{h}}} & \text{if } \mathfrak{h} \models a \mathrel{\mathsf{k}} M', \text{ and} \\ \blacksquare & \text{otherwise.} \end{cases} \\
\llparenthesis \{\!|M|\!\}_{p^+}^{+} \rrparenthesis_a^{\mathfrak{h}} &:= \begin{cases} \{\!|\llparenthesis M \rrparenthesis_a^{\mathfrak{h}}|\!\}_{p^+}^{+} & \text{if } \mathfrak{h} \models a \mathrel{\mathsf{k}} p \vee (a \mathrel{\mathsf{k}} M \wedge a \mathrel{\mathsf{k}} p^+), \text{ and} \\ \blacksquare & \text{otherwise.} \end{cases} \\
\llparenthesis \{\!|M|\!\}_{p}^{-} \rrparenthesis_a^{\mathfrak{h}} &:= \begin{cases} \{\!|\llparenthesis M \rrparenthesis_a^{\mathfrak{h}}|\!\}_{p}^{-} & \text{if } \mathfrak{h} \models a \mathrel{\mathsf{k}} p^+, \text{ and} \\ \blacksquare & \text{otherwise.} \end{cases} \\
\llparenthesis (M, M') \rrparenthesis_a^{\mathfrak{h}} &:= (\llparenthesis M \rrparenthesis_a^{\mathfrak{h}}, \llparenthesis M' \rrparenthesis_a^{\mathfrak{h}})
\end{aligned}
$$

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

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

  3. $$\frac{\mathfrak{h}_l \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \varepsilon(a,M) \approx_a^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \varepsilon(a,M')} \ \llparenthesis M \rrparenthesis_a^{\mathfrak{h}} = \llparenthesis M' \rrparenthesis_a^{\mathfrak{h}'}$$

- *given that* a *is a legitimate agent,*

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

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

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

  6. $$\frac{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \mathtt{I}(b,x,M) \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \mathtt{I}(b,x,M')} \ \llparenthesis M \rrparenthesis_{\mathtt{Eve}}^{\mathfrak{h}} = \llparenthesis M' \rrparenthesis_{\mathtt{Eve}}^{\mathfrak{h}'}$$

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

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. $\mathfrak{h}_l \approx^{(\mathfrak{h},\mathfrak{h}')}_{\cdot} \mathfrak{h}_r$). 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 $\phi' \in \mathfrak{CPL}$.)

Remark 3 *CPL-satisfaction ("model-checking") is undecidable, as secrecy, being CPL-expressible, is.*

1.2.3 Discussion

Relevant implication

In the terminology of relevant logics, both the spatial conditional $\rhd$ 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 $\rhd$, 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

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

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. In other words, if Eve *knows* M then possibly (not only probably) *because* Eve *knows* k in the given model. (Notice the *indicative* mode!) This is a property of Eve's cryptographic *knowledge* w.r.t. its *actuality*. In contrast, consider the tautology

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

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. Our slogan, applying to $\supseteq$, is:

Slogan 7 *Cryptography deserves a proper, relevant notion of implication.*

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 1.4) and prohibition (cf. Appendix A). 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.

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 that, like in SDL, exist in standard epistemic logic (SEL). 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 $\models \neg\exists v(■ = \{\!|v|\!\}_k)$ (cf. "otherwise"-clause in the truth denotation of $\mathsf{K}_a(\phi)$ in Table 1.4). In a cryptographic setting, epistemic necessitation should — and in CPL does — take the following form [CD05a]:

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

The presence of logical omniscience in SEL is, interestingly, 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 an simplification of the one from [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. Solving the problem of logical omniscience has a price:

Proposition 2 *Logical equivalence is* not *a congruence.*

Proof. see Appendix B

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 1.4.3): $\Box_{a.x}(\phi) := \Box(a@x \rightarrow \phi)$ 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 $@_a[\phi]$ from 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 (negative) message term is insecure output (input). With these intended meanings and $\mathcal{S} := \{\ a.x \mid a \in \boldsymbol{\mathcal{A}}_{\texttt{Eve}}$ 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.

1.3 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 A; the reader is urged to consult it) and that $\alpha(b)$ abbreviates disjunction of name generation, sending, and receiving performed by b.

1.3.1 Trust-related affairs

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

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 $\diamondplus(\alpha(b) \land \mathsf{F}\alpha(b))$.

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 $\mathsf{K}_a(\mathsf{prudent}(b))$.[17]

1.3.2 Confidentiality-related affairs

Secret Sharing 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 $a \mathrel{\mathsf{k}} M \land b \mathrel{\mathsf{k}} M \land \forall(c : \mathtt{A}_{\mathtt{Adv}})(c \mathrel{\mathsf{k}} M \rightarrow (c = a \lor c = b))$.

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

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 $\mathsf{K}_a(\exists(c : \mathtt{A})(\phi(c))) \rightarrow \neg\mathsf{K}_a(\phi(b))$.

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 $M' \sqsupseteq_{(a,b)} M := a \mathrel{\mathsf{k}} M \land b \mathrel{\mathsf{k}} M' \sqsupseteq a \mathrel{\mathsf{k}} M$[18] (when $a = b$ we just write $M' \sqsupseteq_a M$).

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

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

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

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

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

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

1.3.3 Authentication-related affairs

Key Confirmation "[...] one party [a] is assured that a second (possibly unidentified) party [b] actually has possession of a particular secret[19] key [k]." [MvOV96, Page 492], written $k : \mathtt{K} \wedge \mathsf{K}_a(b\ \mathsf{k}\ k)$

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 $k : \mathtt{K} \wedge \mathsf{K}_a(\forall(c : \mathtt{A}_{\mathtt{Adv}})(c\ \mathsf{k}\ k \rightarrow (c = a \vee c = b)))$
- *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 $\mathsf{K}_b(M \supseteq_{(a,b)} M)$.

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

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

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

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

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

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

$$\boxplus\forall m_a \forall m_b((\mathsf{K}_b(a \;\mathsf{authored}\; m_a) \land \mathsf{K}_a(b \;\mathsf{authored}\; m_b)) \to \\ \mathsf{K}_a((m_a, m_b) \;\mathsf{sharedSecret}\; (a, b)))$$

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

$$\boxplus\forall m_a \forall m_b((\mathsf{K}_b(a \;\mathsf{authored}\; m_a) \land \mathsf{K}_b(\mathsf{K}_a(b \;\mathsf{authored}\; m_b))) \to \\ \mathsf{K}_b(\mathsf{K}_a((m_a, m_b) \;\mathsf{sharedSecret}\; (a, b))))$$

Entity Authentication (or *Identification*) (safety) via secret sharing 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

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

  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

$$\boxplus\forall m_a \forall m_b((\mathsf{K}_b(a \;\mathsf{authored}\; m_a) \land \mathsf{K}_b(\mathsf{K}_a(b \;\mathsf{authored}\; m_b))) \to \\ \mathsf{K}_b(\mathsf{K}_a((m_a, m_b) \;\mathsf{sharedSecret}\; (a, b))))$$

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

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

$$\boxplus\forall m_a \forall m_b((\mathsf{K}_a(\mathsf{K}_b(a \;\mathsf{authored}\; m_a)) \land \mathsf{K}_b(\mathsf{K}_a(b \;\mathsf{authored}\; m_b))) \to \\ \mathsf{K}_a(\mathsf{K}_b(\mathsf{K}_a((m_a, m_b) \;\mathsf{sharedSecret}\; (a, b)))))$$

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 *can* not be guaranteed.

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

1.3.4 Commitment-related affairs

Proof

- *cryptographic proof*: datum M is a *cryptographic*[20] proof for proposition ϕ, written M $\mathsf{proofFor}$ ϕ, :iff assuming an arbitrary agent a knows M guarantees that a knows that ϕ is true, written $\forall(a : \mathtt{A}_{\mathtt{Adv}})(a\ \mathsf{k}\ M \rhd \mathsf{K}_a(\phi))$
- *provability*: agent a can prove that proposition ϕ is true, written $\mathsf{P}_a(\phi)$, :iff a knows a proof for ϕ, written $\exists m(m\ \mathsf{proofFor}\ \phi \wedge a\ \mathsf{k}\ m)$

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

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 $\diamondplus((a\ \mathsf{k}\ S_b \wedge b\ \mathsf{k}\ S_a) \vee (\neg\, a\ \mathsf{k}\ S_b \wedge \neg\, b\ \mathsf{k}\ S_a))$

- *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 [$\mathtt{Eve}$] to get the honest player's [a] signature [S_a], without the honest player getting the adversary's signature [$S_{\mathtt{Eve}}$]." [ASW00], written $\boxplus(\mathtt{Eve}\ \mathsf{k}\ S_a \rightarrow \diamondplus(a\ \mathsf{k}\ S_{\mathtt{Eve}}))$
- *Completion*: "[...] it is infeasible for the adversary [...] to prevent [a] and [b] from successfully exchanging their signatures." [ASW00], writtten $\diamondplus(a\ \mathsf{k}\ S_b \wedge b\ \mathsf{k}\ S_a)$
- *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."[21] [GJM99], written $\neg\mathsf{P}_b(a\ \mathsf{authored}\ S_a)\,\mathsf{U}\,b\ \mathsf{authored}\ S_b$

Visibly, commitment-related affairs rely on authentication-related affairs.

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

Theorem 1 *The operator P_a is compliant with the modal system **S4**, adapted to the cryptographic setting.*

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

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

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

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

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

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

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

Hence, (the classical logic) CPL can simulate intuitionistic logic via the following macro-definition:

$$\phi \mapsto \phi' := \exists(a : \mathtt{A}_{\mathtt{Adv}})(\mathsf{P}_a(\phi \rightarrow \phi'))$$

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

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 $(\mathfrak{h}', P') \models M \mathrel{\mathsf{proofFor}} \phi \wedge a \mathrel{\mathsf{k}} M$ 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**. That notion of justification roughly corresponds in our (first-order, epistemic-**S5**) setting to the notion of cryptographic proof.

1.3.5 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

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

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 $(\mathfrak{h}, P) \models \phi' \blacktriangleright \phi$,[22] 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 $(\mathfrak{h}, P) \models \top \rhd \phi$.[23]

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

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

The concept of an exo-condition (endo-condition) is to interactive programs what a pre-condition (post-condition) is to non-interactive programs.[24] Our slogan, especially applying to cryptographic protocols, is:

Slogan 8 *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 compositionality.

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.*

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 9 *Debatable requirements entail debatable attacks.*

Table 1.6 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 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 $x_{\mathtt{Bob}}$, 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

[24] $(\mathfrak{h}, P) \models \phi' \vartriangleright \phi$ 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*!

Table 1.6: Protocol narration for core NSPuK

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

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 [BKN06] by instantiating the protocol template displayed in Table 1.7 via substitution of `Alice` for *init* and `Bob` for *resp*. Features of that language are: a primitive for key look-up, 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. The bottom row defines the protocol template, distributing (via parallel composition) the roles at the corresponding locations $\mathit{init}.x_{\mathit{init}}[\,\cdot\,]$ and $\mathit{resp}.x_{\mathit{resp}}[\,\cdot\,]$, 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 ini-

Table 1.7: Protocol template for core NSPuK

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

tiator 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; $\texttt{Get}_{slf}\ (k_{slf} : \texttt{K}^{-}, slf)\ \texttt{in}$ 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*); $\texttt{In}\ \{\!|(x_{slf}, x_{oth})|\!\}^{+}_{\overline{k_{slf}}}\ \texttt{when}\ x_{oth} : \texttt{X}$ guarded (cf. guard $x_{oth} : \texttt{X}$) input of a message with pattern[25] $\{\!|(x_{slf}, x_{oth})|\!\}^{+}_{\overline{k_{slf}}}$ and binding in variable x_{oth} of the other, apparently responding, agent's nonce; $\texttt{Out}_{oth}\ \{\!|x_{oth}|\!\}^{+}_{k_{oth}}$ 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 1.8. 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 1.8: Prehistory for core NSPuK

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

This completes the definition of the initial state

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

of (our attack scenario for) core NSPuK.

Table 1.9 displays the narration of the actual attack. The attack can be 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:

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

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

   with Eve

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

2. Eve *dis*abling the execution (denial of service) of the regular session initiation
$$\texttt{Alice}.x_{a1}[\,\mathrm{NSPuK}_{\texttt{INIT}}(\texttt{Alice},\texttt{Bob})\,]$$

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
$$\texttt{Bob}.x_{b1}[\,\mathrm{NSPuK}_{\texttt{RESP}}(\texttt{Bob})\,]$$
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 $\mathfrak{h}' := \epsilon \cdot \texttt{N}(\texttt{Eve}, x_{e0}, p_{\texttt{Eve}}, \texttt{Eve}) \cdot \texttt{sO}(\texttt{Eve}, x_{e0}, p^{+}_{\texttt{Eve}}, \texttt{Alice}) \cdot \texttt{sI}(\texttt{Alice}, x_{a0}, p^{+}_{\texttt{Eve}}, \texttt{Eve}) \cdot \texttt{sO}(\texttt{Eve}, x_{e0}, p^{+}_{\texttt{Eve}}, \texttt{Bob}) \cdot \texttt{sI}(\texttt{Bob}, x_{b0}, p^{+}_{\texttt{Eve}}, \texttt{Eve})$. More formally,

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

by which we obtain

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

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 [BKN06].

Table 1.9: Attack narration for NSPuK

1.	$\texttt{Alice} \longrightarrow \texttt{Eve}$	:	$\{\!|(x_{\texttt{Alice}}, \texttt{Alice})|\!\}^{+}_{p^{+}_{\texttt{Eve}}}$
$1'$.	$\texttt{Eve}_{\texttt{Alice}} \longrightarrow \texttt{Bob}$	:	$\{\!|(x_{\texttt{Alice}}, \texttt{Alice})|\!\}^{+}_{p^{+}_{\texttt{Bob}}}$
$2'$.	$\texttt{Bob} \longrightarrow \texttt{Eve}_{\texttt{Alice}}$	:	$\{\!|(x_{\texttt{Alice}}, x_{\texttt{Bob}})|\!\}^{+}_{p^{+}_{\texttt{Alice}}}$
2.	$\texttt{Eve} \longrightarrow \texttt{Alice}$	:	$\{\!|(x_{\texttt{Alice}}, x_{\texttt{Bob}})|\!\}^{+}_{p^{+}_{\texttt{Alice}}}$
3.	$\texttt{Alice} \longrightarrow \texttt{Eve}$	:	$\{\!|x_{\texttt{Bob}}|\!\}^{+}_{p^{+}_{\texttt{Eve}}}$
$3'$.	$\texttt{Eve}_{\texttt{Alice}} \longrightarrow \texttt{Bob}$	:	$\{\!|x_{\texttt{Bob}}|\!\}^{+}_{p^{+}_{\texttt{Bob}}}$

1.4 Extending CPL with real-time to tCPL

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.

1.4.1 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[26] 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 [BGK06] with *dense* time

- Property-based formalisms (logics): *interval*-based [HS04]; time-parametrised epistemic modalities [KM99] and a third-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.*

[26] 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.

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 1.1.2); 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 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 [BKN06]. 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 [BGK06]. 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 1.4.3); (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). Regarding rational versus real numbers: consider that cryptographic messages have finite length, which implies that real numbers, e.g., real-valued time stamps, are not transmittable as such, and that real clocks only have finite precision.

Regarding the timed adversary model: consider that 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 [BKN06] to tC^3 [BGK06].

1.4.2 Extension

The notion of execution from [BGK06], 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 $\mathtt{CV}$, and

$t_b, t_e \in \mathcal{TV} := \mathcal{CV} \cup \{-\infty, \infty\}$ denote *time values* having the associated type $\mathtt{TV}$. 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 $\circlearrowright$ 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:

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

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 1.3) 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 ($\mathfrak{h}' \vdash^{\mathcal{E}}_{\mathtt{Eve}} \{\!|M|\!\}_k$):

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

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

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

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 1.4):

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

5. stipulation that composition of protocol histories preserves the natural order of real-time

1.4.3 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:

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

- *durations* [ZHR91], [ZH04] (cf. Table 1.10)

The cryptographic states of affairs involving qualitative temporal modalities from Section 1.3 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 1.10: Definability of durations

$$\begin{array}{rcl}
\Delta\ \mathsf{duration}_{(t,t')}\ \phi & := & \diamondminus_t(\Delta\ \mathsf{duration}_{t'}\ \phi) \lor \diamondplus_t(\Delta\ \mathsf{duration}_{t'}\ \phi) \\
\Delta\ \mathsf{duration}_{t'}\ \phi & := & (\phi \to \forall t_d(t_d@\mathtt{Eve} \to \\
 & & \qquad\qquad \bigoplus((\phi \to \forall t_m((t_m@\mathtt{Eve} \land t_m \leq t') \to \\
 & & \qquad\qquad\qquad\qquad \Delta - (t_m - t_d)\ \mathsf{duration}_{t'}\ \phi)) \land \\
 & & \qquad\qquad\quad (\neg\phi \to \bigoplus(\Delta\ \mathsf{duration}_{t'}\ \phi))))) \land \\
 & & (\neg\phi \to \bigoplus(\Delta\ \mathsf{duration}_{t'}\ \phi))
\end{array}$$

1.4.4 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 1.11). 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 symmetric 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.

Table 1.11: Protocol narration for WMF

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

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 [BGK06], a timed extension of the one of [BKN06], by instantiating the protocol template displayed in Table 1.12 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 look-up of stored

Table 1.12: Protocol template for WMF

<table>
<tr>
<td>$\mathrm{WMF}_{\texttt{INIT}}(\mathit{slf}, \mathit{srv}, \mathit{oth}, \Delta_v) :=$
$\texttt{Get}_{\mathit{slf}}\ (t_s : \texttt{CV}, ■)\ \texttt{in}$
$\texttt{New}\ (k_{so} : \texttt{K}, ((\mathit{slf}, \mathit{oth}), (t_s, t_s + \Delta_v + \Delta_v))).$
$\texttt{Get}_{\mathit{srv}}\ (k_{ss} : \texttt{K}, (\mathit{slf}, \mathit{srv}))\ \texttt{in}$
$\texttt{Out}_{\mathit{srv}}\ \mathit{slf}.$

$\texttt{Out}_{\mathit{srv}}\ \{\!|((t_s, \mathit{oth}), k_{so})|\!\}_{k_{ss}}.1$</td>
<td>$\mathrm{WMF}_{\texttt{SERV}}(\mathit{slf}, \Delta_v) :=$

$\texttt{In}\ \mathit{fst}\ \texttt{when}\ \mathit{fst} : \texttt{A}.$
$\texttt{Get}_{\mathit{slf}}\ (k_{sf} : \texttt{K}, (\mathit{slf}, \mathit{fst}))\ \texttt{in}$
$\texttt{In}\ \{\!|((t, \mathit{snd}), \mathit{key})|\!\}_{\overline{k_{sf}}}\ \texttt{when}\ t : \texttt{TV} \wedge$
$\quad \exists t_s(t_s : \texttt{TV} \wedge t_s@\mathit{slf} \wedge t + \Delta_v \leq t_s) \wedge$
$\quad \mathit{snd} : \texttt{A} \wedge$
$\quad \mathit{key} : \texttt{K}.$
$\texttt{Get}_{\mathit{slf}}\ (k_{ss} : \texttt{K}, (\mathit{slf}, \mathit{snd}))\ \texttt{in}$
$\texttt{Out}_{\mathit{snd}}\ \{\!|((t_s, \mathit{fst}), \mathit{key})|\!\}_{k_{ss}}.1$</td>
<td>$\mathrm{WMF}_{\texttt{RESP}}(\mathit{slf}, \mathit{srv}, \Delta_v) :=$

$\texttt{Get}_{\mathit{srv}}\ (k_{ss} : \texttt{K}, (\mathit{slf}, \mathit{srv}))\ \texttt{in}$
$\texttt{In}\ \{\!|((t, \mathit{oth}), \mathit{key})|\!\}_{\overline{k_{ss}}}\ \texttt{when}\ t : \texttt{TV} \wedge$
$\quad \exists t_s(t_s : \texttt{TV} \wedge t_s@\mathit{slf} \wedge t + \Delta_v \leq t_s) \wedge$
$\quad \mathit{oth} : \texttt{A} \wedge$
$\quad \mathit{key} : \texttt{K}.1$</td>
</tr>
<tr>
<td colspan="3">$\mathrm{WMF}(\mathit{init}, \mathit{srv}, \mathit{resp}, x_{\mathit{init}}, x_{\mathit{serv}}, x_{\mathit{resp}}, \Delta_v) := \mathit{init}.x_{\mathit{init}}[\,\mathrm{WMF}_{\texttt{INIT}}(\mathit{init}, \mathit{srv}, \mathit{resp}, \Delta_v)\,] \;|||$
$\mathit{srv}.x_{\mathit{srv}}[\,\mathrm{WMF}_{\texttt{SERV}}(\mathit{srv}, \Delta_v)\,] \;|||$
$\mathit{resp}.x_{\mathit{resp}}[\,\mathrm{WMF}_{\texttt{RESP}}(\mathit{resp}, \mathit{srv}, \Delta_v)\,]$</td>
</tr>
</table>

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 $\mathit{init}.x_{\mathit{init}}[\,\cdot\,]$, $\mathit{srv}.x_{\mathit{srv}}[\,\cdot\,]$, and $\mathit{resp}.x_{\mathit{resp}}[\,\cdot\,]$, 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 1.13. 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 1.13: Prehistory for WMF

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

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 1.14 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 1.14: Attack narration for WMF

$1a'$.	$\mathtt{Eve}_{\mathtt{Bob}} \to \mathtt{Trent}$	:	$\mathtt{Bob}$
$1b'$.	$\mathtt{Eve}_{\mathtt{Bob}} \to \mathtt{Trent}$	:	$\{\!|((t_{\mathtt{Trent}}, \mathtt{Alice}), k_{\mathtt{AliceBob}})|\!\}_{k_{\mathtt{BobTrent}}}$
$2'$.	$\mathtt{Trent} \to \mathtt{Eve}_{\mathtt{Alice}}$	:	$\{\!|((t'_{\mathtt{Trent}}, \mathtt{Bob}), k_{\mathtt{AliceBob}})|\!\}_{k_{\mathtt{AliceTrent}}}$
$1a''$.	$\mathtt{Eve}_{\mathtt{Alice}} \to \mathtt{Trent}$	:	$\mathtt{Alice}$
$1b''$.	$\mathtt{Eve}_{\mathtt{Alice}} \to \mathtt{Trent}$	:	$\{\!|((t'_{\mathtt{Trent}}, \mathtt{Bob}), k_{\mathtt{AliceBob}})|\!\}_{k_{\mathtt{AliceTrent}}}$
$2''$.	$\mathtt{Trent} \to \mathtt{Bob}$	:	$\{\!|((t''_{\mathtt{Trent}}, \mathtt{Alice}), k_{\mathtt{AliceBob}})|\!\}_{k_{\mathtt{BobTrent}}}$

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 and responding Bob, mediating Trent.

More formally, let

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

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

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

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 [BGK06].

1.5 Conclusion

Achievements We believe having achieved with CPL an original synthesis of an unprecedented variety of logical concepts that are relevant to the security of communication. In particular, we have

1. defined a cryptographically meaningful (in the sense of Dolev-Yao for the moment) *epistemic modality*

2. invented a cryptographically interesting *epistemic conditional* thanks to the auxiliary invention of *complex truth values*

3. pioneered the *application of spatial logic* to the formalisation of cryptographic states of affairs

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

5. demonstrated the macro-definability of *Gödel's provability modality* within the spatio-epistemic fragment of CPL, and by that, shown that CPL can also simulate intuitionistic logic

6. demonstrated the definability of cryptographically meaningful *deontic modalities* within the spatio-epistemico-temporal fragment of CPL

7. demonstrated that the *addition of (dense-valued) real-time* to an untimed, property-based formalism for cryptographic protocols can be simple and backwards-compatible, when properly conceived.

We have also achieved the formalisation of an unprecedented variety of cryptographic states of affairs in a single logic. 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.

Present work At present, we are extending CPL with a notion of probabilistic polynomial-time computation in order to accommodate CPL to modern (complexity-theoretic) cryptography (cf. Chapter 2).

Future work Further, in case first-order CPL should not suffice for some applications, it would be trivial to extend CPL to the second-order. Just allow (unquoted) message types (denoting sets of messages) as messages, and quantification may range over second-order entities (sets).

Finally, a modal model theory for cryptography based on CPL, a proof system for CPL, the study of decidable fragments, and the actual construction of a Curry-Howard isomorphism between cryptographic protocols and propositions are our main desiderata. Fortunately, and this is a strength of our approach, axiomatisations of each one of CPL's operators except for the epistemic conditional exist, which almost reduces the task of defining a proof system for CPL to finding the laws that correctly axiomatise the *mixing* of operators.

Acknowledgements I would like to thank Mika Cohen for our stimulating discussions about crypto logics and his constructive criticism of my work, and Lawrence S. Moss for his valuable comments.

Chapter 2

Towards Probabilistic Polynomial-time Cryptography

2.1 Introduction

We sketch an Ockham's razor extension of core CPL (cf. Chapter 1) with a notion of probabilistic polynomial-time (PP) computation. We hope to convince the reader that adding a notion of PP-computation to a (property-based) formalism for cryptographic protocols can 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. Chapter 1, [Lam05], and [BGK06, Kra06]), need not be built from scratch nor be heavy-weight.

The 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.

The style of our presentation of concepts is alternative to the traditional style, which is *operational* and based on interactive (Turing) *machines*. In contrast, our style is *logical* and based on *linguistic abstractions* of cryptographic concepts.

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*

[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.

- Model-based: [MRST06] a process algebra for equivalence-based specification and verification.

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. Chapter 1 and [Kra06]) 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.

2.1.1 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 1) 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 2.1).

Table 2.1: 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 (name, logical constant) 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), probability can be an (epistemological) measure of our (subjective) belief of states of affairs as well as an (ontological) measure of their (objective) possibility. The

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

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

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.

2.1.2 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 2.2) and *message denotation*, i.e., message evaluation, (cf. Table 2.3).

Table 2.2: Probabilistic process reduction

Probability theory	CPL
sample space	$\mathcal{P} \times \mathcal{H}$
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	$\{\ (\mathfrak{h}', P') \mid (\mathfrak{h}, P) \longrightarrow (\mathfrak{h}', P') \models \phi\ \}$

In Table 2.2, $\mathcal{P}$ designates the set of process terms (protocol models) P (the potential future), $\mathcal{H}$ the set of protocol histories $\mathfrak{h}$ (a finite word of past protocol events[4] ε, e.g., message output or input and new-key/-nonce generation), $\longrightarrow$ the relation of process reduction (modelling interleaving concurrency of protocol events), and $\models$ CPL's relation of satisfaction. 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 $\mathfrak{s} \in \mathcal{P} \times \mathcal{H}$.

In Table 2.3, $\mathcal{M}$ designates the set of message terms and $[\![\cdot]\!]$ the function of message denotation.

2.1.3 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

[4] not to be confused with the concept of atomic events from probability theory

Table 2.3: Probabilistic message denotation

Probability theory	CPL
sample space	$\{\mathtt{0},\mathtt{1}\}^*$
variable (experiment) X	message term $M \in \mathcal{M}$
atomic event	$[\![M]\!] = s$ where $[\![\cdot]\!] \subseteq \mathcal{M} \times \{\mathtt{0},\mathtt{1}\}^*$
possible value (outcome) of X	$s \in \{\mathtt{0},\mathtt{1}\}^*$ 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	$\{\ s \mid s = [\![M]\!]\ \}$

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.

2.2 Extension

This section describes the extension of CPL to ppCPL. The extension depends on the core described in Chapter 1 (the reader is urged to consult it).

2.2.1 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 $\mathtt{PV}$

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 $a \underset{NGA}{\overset{\iota}{\circlearrowright}} n.o$ 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 1.4)

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 $\mathsf{B}^q_a(\phi)$

2.2.2 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 $ALGO ::= NGA \mid ■$, where $ALGO \in NGA$ if M is a name and $ALGO = ■$ otherwise

2. refinement of individual knowledge *simpliciter* to *PP* individual knowledge (cf. Table 2.4) relying on (stateful) PP message denotation:

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

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

3. let

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

   (a) refinement of propositional knowledge *simpliciter*

$$\llbracket \mathsf{K}_a(\phi) \rrbracket_{\mathfrak{p}}^i := (\text{for all } \mathfrak{s}, \text{ if } \mathfrak{s} \in \mathcal{K}_a(\mathfrak{p}, i) \text{ then } \mathfrak{s}' \models_{\mathcal{E}'} \phi',\ \mathcal{E}'_{(\mathfrak{s},\phi)})$$
$$\text{where } (\mathfrak{s}', \phi') := \begin{cases} (\mathfrak{s}, \phi) & \text{if } \mathfrak{s} = \mathfrak{p}@i, \text{ and} \\ (\llparenthesis\, \mathfrak{s} \,\rrparenthesis_a^{\mathfrak{p}@i}, \llparenthesis\, \phi \,\rrparenthesis_a^{\mathfrak{p}@i}) & \text{otherwise.} \end{cases}$$

   to *PP* propositional knowledge

$$\llbracket \mathsf{K}_a(\phi) \rrbracket_{\mathfrak{p}}^i := (\text{for all } \mathfrak{s}, \text{ if } \ \mathfrak{s} \in \mathcal{K}_a(\mathfrak{p}, i)$$
$$\text{then } \ \mathfrak{s}' \models_{\mathcal{E}'} \phi' \text{ and there is a polynomial}$$
$$p : \mathbb{N} \to \mathbb{N} \text{ s.t. } |\mathfrak{s}| \leq p(|\mathfrak{p}@i|),\ \mathcal{E}'_{(\mathfrak{s},\phi)})$$
$$\text{where } (\mathfrak{s}', \phi') := \begin{cases} (\mathfrak{s}, \phi) & \text{if } \mathfrak{s} = \mathfrak{p}@i, \text{ and} \\ (\llparenthesis\, \mathfrak{s} \,\rrparenthesis_a^{\mathfrak{p}@i}, \llparenthesis\, \phi \,\rrparenthesis_a^{\mathfrak{p}@i}) & \text{otherwise.} \end{cases}$$

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

$$\llbracket \mathsf{B}_a^q(\phi) \rrbracket_{\mathfrak{p}}^i := (q = \tfrac{|\mathcal{B}_a(\mathfrak{p},i)|}{|\mathcal{K}_a(\mathfrak{p},i)|} \text{ and}$$
$$\text{for all } \mathfrak{s}, \text{ if } \mathfrak{s} \in \mathcal{B}_a(\mathfrak{p}, i) \text{ then } \mathfrak{s}' \models_{\mathcal{E}'} \phi',\ \mathcal{E}'_{(\mathfrak{s},\phi)})$$
$$\text{where } (\mathfrak{s}', \phi') := \begin{cases} (\mathfrak{s}, \phi) & \text{if } \mathfrak{s} = \mathfrak{p}@i, \text{ and} \\ (\llparenthesis\, \mathfrak{s} \,\rrparenthesis_a^{\mathfrak{p}@i}, \llparenthesis\, \phi \,\rrparenthesis_a^{\mathfrak{p}@i}) & \text{otherwise.} \end{cases}$$

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

Table 2.4: 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

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

2.3 Case studies

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

Table 2.5: 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 $(P, \mathfrak{h}) \models \phi$ (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 ($\phi(M)$)	

2.3.1 Fundamental concepts

This section is in the spirit of [Gol01].

Definition 10 (Random propositional guessing)

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

Definition 11 (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 12 (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 1 (One-way function)
"[...] a function that is easy to compute but hard to invert." [Gol01, Page 32]
Ease of computation $\models a \mathrel{\mathsf{k}} M \rightarrow a \mathrel{\mathsf{k}} f(M)$

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

┘

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 2.4). 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 2 (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 $a \mathsf{k} f(M) \wedge \neg a \mathsf{k} M$.* $\lrcorner$

Notice that we identify agents with feasible algorithms!

Formalisation 3 (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 $\neg(M = M')$.* $\lrcorner$

Definition 13 (Cryptographic evidence and proof)

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

Conjecture 1

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

Formalisation 4 (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]*

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

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

⌟

Definition 14 (Interactive provability)

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

Our (macro-defined) operator for interactive provability is a generalisation to the interactive setting of our macro-definition of Gödel's provability modality (cf. Chapter 1).

Proposition 3 $\models \mathsf{IP}_{(a,b)}(\phi) \rightarrow \mathsf{P}_a(\phi)$

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 5 (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 6 (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]

$$\begin{array}{l} \mathsf{ZK}_{(a,b)}(\phi) := \mathsf{IP}_{(a,b)}(\mathsf{K}_a(\exists m'(\mathsf{K}_b(m' \ \mathsf{proofFor} \ \phi))) \land \\ \qquad\qquad\qquad\qquad \neg\exists m''(\mathsf{K}_a(\mathsf{K}_b(m'' \ \mathsf{evidenceFor} \ \phi)))) \end{array}$$

⌟

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* 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 $a = \mathtt{Eve}$. 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]*

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

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

Formalisation 7 (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]*

$$
\begin{array}{rcl}
A & ::= & (a, \blacksquare) \mid (b, A) \\[1ex]
M \ \mathsf{iProofFor}_{(a,A)}\ \phi & := & M \ \mathsf{iProofFor}^{a}_{(a,A)}\ \phi \\
(M, \blacksquare)\ \mathsf{iProofFor}^{c}_{(a,\blacksquare)}\ \phi & := & c \ \mathsf{k}\ M \wedge M \ \mathsf{proofFor}\ \phi \\
(M, (M', I))\ \mathsf{iProofFor}^{c}_{(a,(b,A))}\ \phi & := & M' \supseteq_{(a,b)} M \wedge (M', I)\ \mathsf{iProofFor}^{c}_{(b,A)}\ \phi
\end{array}
$$

⌟

Definition 15 (Quotient proof)

$$
\begin{array}{rcl}
M \mid_a M' & := & \neg(a \ \mathsf{k}\ M \supseteq a \ \mathsf{k}\ M') \wedge \\
 & & \neg(a \ \mathsf{k}\ M' \supseteq a \ \mathsf{k}\ M) \\
(M, M')\ \mathsf{disjointEvidenceFor}\ \phi & := & \forall a(\mathsf{K}_a(\phi) \rhd (\mathsf{K}_a(\phi) \supseteq a \ \mathsf{k}\ (M, M') \wedge \\
 & & \qquad M \mid_a M'))
\end{array}
$$

$$
\begin{array}{rcl}
(M, \blacksquare)\ \mathsf{mutuallyDisjointEvidenceFor}\ \phi & := & M \ \mathsf{evidenceFor}\ \phi \\
(M, \overline{M})\ \mathsf{mutuallyDisjointEvidenceFor}\ \phi & := & (M, \overline{M})\ \mathsf{disjointEvidenceFor}\ \phi \wedge \\
 & & \overline{M}\ \mathsf{mutuallyDisjointEvidenceFor}\ \phi \\[1ex]
M \ \mathsf{quotientProofFor}\ \phi & := & M \ \mathsf{proofFor}\ \phi \wedge \\
 & & M \ \mathsf{mutuallyDisjointEvidenceFor}\ \phi
\end{array}
$$

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 16 (Multi-prover Zero-Knowledge)

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

where

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

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

Formalisation 8 (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)

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

$\lrcorner$

2.3.2 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 17 (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]*

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

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

Formalisation 9 (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 10 (Ciphertext-only attack)
"[...] the adversary (or cryptanalyst) tries [$\supseteq$] to deduce the decryption key [k (symmetric) resp. p (private)] or plaintext [M] by only [$\equiv$] observing ciphertext [$m_1, \ldots, m_n$]." [MvOV96, Page 41]

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

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

⌟

Formalisation 11 (Known-plaintext attack)
"[...] the adversary has a quantity of plaintext [$m_1, \ldots, m_n$] and corresponding ciphertext." [MvOV96, Page 41]

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

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

Formalisation 12 (Chosen-plaintext attack)
"[...] the adversary chooses [$\triangleright$] plaintext [m] and is then given [$\blacktriangleright$] corresponding ciphertext. Subsequently [$\diamondplus$], the adversary uses any information deduced [$\supseteq$] in order to recover plaintext [M] corresponding to previously unseen ciphertext." [MvOV96, Page 41]

$$\begin{aligned}&\exists(k : \mathtt{K})(\neg\,\mathtt{Eve}\ \mathsf{k}\ [M]_k \ \land\\ &\qquad \exists m(\mathtt{Eve}\ \mathsf{k}\ m \ \triangleright\\ &\qquad\quad (\mathtt{Eve}\ \mathsf{k}\ [m]_k \blacktriangleright \diamondplus(M \supseteq_{\mathtt{Eve}} (m, [m]_k)))))\end{aligned}$$

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

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

Formalisation 13 (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

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

Then

$$\begin{array}{l} \exists(p : \mathtt{K}^-)(\neg\, \mathtt{Eve}\ \mathsf{k}\ [M]_{p^+} \wedge \\ \qquad \exists m(\exists m'(m'\ \mathsf{aCPC}_{\mathtt{Eve}}^{p^+}\ \mathtt{Eve}\ \mathsf{k}\ m) \triangleright \\ \qquad\qquad (\mathtt{Eve}\ \mathsf{k}\ [m]_{p^+} \blacktriangleright \diamondplus(M \supseteq_{\mathtt{Eve}} (m, [m]_{p^+}))))) \end{array}$$

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

Formalisation 14 (Chosen-ciphertext attack)

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

$$\begin{array}{l} \exists(k : \mathtt{K})(\neg\, \mathtt{Eve}\ \mathsf{k}\ [M]_k \wedge \\ \qquad \exists m(\mathtt{Eve}\ \mathsf{k}\ [m]_k \triangleright \\ \qquad\qquad (\mathtt{Eve}\ \mathsf{k}\ m \blacktriangleright \diamondplus(M \supseteq_{\mathtt{Eve}} (m, [m]_k))))) \end{array}$$

⌟

Formalisation 15 (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

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

Then

$$\begin{aligned}&\exists(k:\mathtt{K})(\neg\,\mathtt{Eve}\ \mathsf{k}\ [M]_k\ \land\\ &\qquad\exists m(\exists m'(m'\ \mathsf{sCCC}^{k}_{\mathtt{Eve}}\ \mathtt{Eve}\ \mathsf{k}\ m)\ \rhd\\ &\qquad\qquad(\mathtt{Eve}\ \mathsf{k}\ [m]_k \blacktriangleright \diamondplus(M \supseteq_{\mathtt{Eve}} (m,[m]_k)))))\end{aligned}$$

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 16 (Key-only attack)

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

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

⌟

Formalisation 17 (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]

$$\begin{aligned}&\exists v\exists(a:\mathtt{A})(v^{+}\ \mathsf{puk}\ a \land \mathtt{Eve}\ \mathsf{k}\ v^{+}\ \land\\ &\qquad\exists m_1\cdots\exists m_n(\mathtt{Eve}\ \mathsf{k}\ ]m_1[_v \land\cdots\land \mathtt{Eve}\ \mathsf{k}\ ]m_n[_v\ \land\\ &\qquad\qquad\mathtt{Eve}\ \mathsf{k}\ m_1 \land\cdots\land \mathtt{Eve}\ \mathsf{k}\ m_n))\end{aligned}$$

⌟

Formalisation 18 (Chosen-message attack)

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

$$\begin{aligned}&\exists v\exists(a:\mathtt{A})(v^{+}\ \mathsf{puk}\ a \land \mathtt{Eve}\ \mathsf{k}\ v^{+}\ \land\\ &\qquad\exists m_1\cdots\exists m_n((\mathtt{Eve}\ \mathsf{k}\ m_1 \land\cdots\land \mathtt{Eve}\ \mathsf{k}\ m_n)\ \rhd\\ &\qquad\qquad(\mathtt{Eve}\ \mathsf{k}\ ]m_1[_v \land\cdots\land \mathtt{Eve}\ \mathsf{k}\ ]m_n[_v)))\end{aligned}$$

⌟

Formalisation 19 (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^{p}_{a} an inductively-defined macro expressing the realisation of such a chain for agent a

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

Then

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

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 20 (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]

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

⌟

Formalisation 21 (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]

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

⌟

Formalisation 22 (Universal forgery)
"An adversary is able to create a valid signature for an arbitrary [∀] message chosen a priori."

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

⌟

Formalisation 23 (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]

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

⌟

Appendix A

Specification library

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'
$\phi \rightarrow \phi' \;:=\; \neg\phi \vee \phi'$	if ϕ then ϕ'
$\phi \leftrightarrow \phi' \;:=\; (\phi \rightarrow \phi') \wedge (\phi' \rightarrow \phi)$	ϕ if and only if ϕ'
$\exists v(\phi) \;:=\; \neg\forall v(\neg\phi)$	there is v s.t. ϕ
$\forall(v : \theta)(\phi) \;:=\; \forall v(v : \theta \rightarrow \phi)$	
$\exists(v : \theta)(\phi) \;:=\; \exists v(v : \theta \wedge \phi)$	

Modal operators

$\mathsf{F}\phi \;:=\; \neg\mathsf{P}\phi$	it is forbidden that ϕ
$\phi \equiv \phi' \;:=\; (\phi \supseteq \phi') \wedge (\phi' \supseteq \phi)$	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
$\phi' \blacktriangleright \phi \;:=\; \neg(\phi' \rhd \neg\phi)$	assert ϕ' guarantee ϕ
$* \;:=\; \bigominus\bot$	in the beginning
$\dagger \;:=\; \bigoplus\bot$	in the end
$\boxminus\phi \;:=\; \phi\,\mathsf{S}\,\bot$	so far ϕ
$\diamondminus\phi \;:=\; \neg\boxminus\neg\phi$	once ϕ
$\mathsf{1}.\phi \;:=\; \phi \wedge \neg\bigominus\diamondminus\phi$	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'
$\phi \leadsto \phi' \;:=\; (\phi \leftrightarrow \diamondplus\phi') \wedge (\phi' \leftrightarrow \diamondminus\phi)$	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'

$a \mathrel{\mathsf{h}} F \;:=\; \exists v(F \preccurlyeq v \wedge a \mathrel{\mathsf{k}} v)$	a has/possesses F
$a \mathrel{\mathsf{tk}} F \;:=\; a \mathrel{\mathsf{h}} F \wedge \neg\, a \mathrel{\mathsf{k}} F$	a tacitly knows F
$F : \emptyset \;:=\; \bot$	
$F : \mathtt{H}[\theta] \;:=\; \exists(v : \theta)(F = \lceil v \rceil)$	
$F : \mathtt{SC}_{F'}[\theta] \;:=\; \exists(v : \theta)(F = \{\!\lvert v \rvert\!\}_{F'})$	
$F : \mathtt{AC}_{p^+}[\theta] \;:=\; \exists(v : \theta)(F = \{\!\lvert v \rvert\!\}^{+}_{p^+})$	
$F : \mathtt{S}_{p}[\theta] \;:=\; \exists(v : \theta)(F = \{\!\lvert v \rvert\!\}^{-}_{p})$	
$F : \mathtt{T}[\theta, \theta'] \;:=\; \exists(v : \theta)\exists(v' : \theta')(F = (v, v'))$	
$F : \theta \cup \theta' \;:=\; F : \theta \vee F : \theta'$	
$F : \theta \cap \theta' \;:=\; F : \theta \wedge F : \theta'$	
$F : \theta \setminus \theta' \;:=\; F : \theta \wedge \neg\, F : \theta'$	
$F : \mathtt{M} \;:=\; \top$	
$F : \mathtt{SC}[\theta] \;:=\; \exists v(F : \mathtt{SC}_{v}[\theta])$	
$F : \mathtt{AC}[\theta] \;:=\; \exists(v : \mathtt{K}^{+})(F : \mathtt{AC}_{v}[\theta])$	
$F : \mathtt{C}[\theta] \;:=\; F : \mathtt{SC}[\theta] \cup \mathtt{AC}[\theta]$	
$F : \mathtt{S}[\theta] \;:=\; \exists(v : \mathtt{K}^{-})(F : \mathtt{S}_{v}[\theta])$	
$\theta \sqsubseteq \theta' \;:=\; \forall(v : \theta)(v : \theta')$	θ is a subtype of θ'
$\theta = \theta' \;:=\; \theta \sqsubseteq \theta' \wedge \theta' \sqsubseteq \theta$	
$\theta \sqsubset \theta' \;:=\; \theta \sqsubseteq \theta' \wedge \theta' \neq \theta$	
$F \multimapinv F' \;:=\; \exists v(\{\!\lvert v \rvert\!\}_{F} \preccurlyeq F')$	
$p^{+} \multimapinv^{+} F \;:=\; \exists v(\{\!\lvert v \rvert\!\}^{+}_{p^+} \preccurlyeq F)$	
$p \multimapinv^{-} F \;:=\; \exists v(\{\!\lvert v \rvert\!\}^{-}_{p} \preccurlyeq F)$	
$F \multimapinv^{*} F' \;:=\; F \multimapinv F' \vee F \multimapinv^{+} F' \vee F \multimapinv^{-} F'$	F is operational in F'
$F \multimap F' \;:=\; \exists v \exists v'(F \preccurlyeq v \wedge \{\!\lvert v \rvert\!\}_{v'} \preccurlyeq F')$	
$F \multimap^{+} F' \;:=\; \exists v \exists(p^{+} : \mathtt{K}^{+})(F \preccurlyeq v \wedge \{\!\lvert v \rvert\!\}^{+}_{p^+} \preccurlyeq F')$	
$F \multimap^{-} F' \;:=\; \exists v \exists(p : \mathtt{K}^{-})(F \preccurlyeq v \wedge \{\!\lvert v \rvert\!\}^{-}_{p} \preccurlyeq F')$	
$F \multimap^{*} F' \;:=\; F \multimap F' \vee F \multimap^{+} F' \vee F \multimap^{-} F'$	F is guarded in F'
$k \mathrel{\mathsf{sk}} a \;:=\; \exists b \exists o(b \circlearrowright k.o \wedge a \preccurlyeq o)$	k is a symmetric key for a
$k \mathrel{\mathsf{sk}^{1}} a \;:=\; k \mathrel{\mathsf{sk}} a \wedge k : \mathtt{K}^{1}$	k is a session/short-term key for a
$k \mathrel{\mathsf{sk}^{\infty}} a \;:=\; k \mathrel{\mathsf{sk}} a \wedge k : \mathtt{K}^{\infty}$	k is a long-term key for a
$p \mathrel{\mathsf{prk}} a \;:=\; \exists b \exists o(b \circlearrowright p.o \wedge a \preccurlyeq o)$	p is a private key for a
$n \mathrel{\mathsf{puk}} a \;:=\; \exists v(v^{+} = n \wedge v \mathrel{\mathsf{prk}} a)$	n is a public key for a
$n \mathrel{\mathsf{ck}} a \;:=\; n \mathrel{\mathsf{sk}} a \vee n \mathrel{\mathsf{prk}} a$	n is a confidential key for a

Appendix B

Proofs

Proposition 2

Proof. by counterexample:

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

Hence, $\models \mathsf{K}_a(\texttt{Eve} : \texttt{Adv})$, i.e., $\mathsf{K}_a(\texttt{Eve} : \texttt{Adv})$ is a logical truth (tautology), but $\not\models \mathsf{K}_a(k : \texttt{K})$, i.e., $\mathsf{K}_a(k : \texttt{K})$ 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 4 $\models \mathsf{P}_a(\phi \rightarrow \phi') \rightarrow (\mathsf{P}_a(\phi) \rightarrow \mathsf{P}_a(\phi'))$

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.	$\mathfrak{s} \models (M, M') \ \mathsf{proofFor}\ \phi'$	33
35.	$\mathfrak{s} \models a \ \mathsf{k}\ (M, M')$	9, 12, property of k
36.	$\mathfrak{s} \models (M, M') \ \mathsf{proofFor}\ \phi'$ and $\mathfrak{s} \models a \ \mathsf{k}\ (M, M')$	34, 35
37.	$\mathfrak{s} \models (M, M') \ \mathsf{proofFor}\ \phi' \wedge a \ \mathsf{k}\ (M, M')$	36
38.	there is $M'' \in \mathcal{M}$ s.t. $\mathfrak{s} \models M'' \ \mathsf{proofFor}\ \phi' \wedge a \ \mathsf{k}\ M''$	37
39.	$\mathfrak{s} \models \exists m(m \ \mathsf{proofFor}\ \phi' \wedge a \ \mathsf{k}\ m)$	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.	$\mathfrak{s} \models \mathsf{P}_a(\phi) \to \mathsf{P}_a(\phi')$	43
45.	if $\mathfrak{s} \models \mathsf{P}_a(\phi \to \phi')$ then $\mathfrak{s} \models \mathsf{P}_a(\phi) \to \mathsf{P}_a(\phi')$	1, 44
46.	for all $\mathfrak{s}$, if $\mathfrak{s} \models \mathsf{P}_a(\phi \to \phi')$ then $\mathfrak{s} \models \mathsf{P}_a(\phi) \to \mathsf{P}_a(\phi')$	45
47.	$\mathsf{P}_a(\phi \to \phi') \Rightarrow \mathsf{P}_a(\phi) \to \mathsf{P}_a(\phi')$	46, Definition 8
48.	$\models \mathsf{P}_a(\phi \to \phi') \to (\mathsf{P}_a(\phi) \to \mathsf{P}_a(\phi'))$	47, Lemma 1

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

Proof.

1.	$\mathfrak{s} \models \mathsf{P}_a(\phi)$	hyp.
2.	$\mathfrak{s} \models \exists m(m \;\mathsf{proofFor}\; \phi \land a \;\mathsf{k}\; m)$	1
3.	there is $M \in \mathcal{M}$ s.t. $\mathfrak{s} \models M \;\mathsf{proofFor}\; \phi \land a \;\mathsf{k}\; M$	2
4.	$M \in \mathcal{M}$ and $\mathfrak{s} \models M \;\mathsf{proofFor}\; \phi \land a \;\mathsf{k}\; M$	hyp.
5.	$\mathfrak{s} \models M \;\mathsf{proofFor}\; \phi$ 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
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.	$\models \mathsf{P}_a(\phi) \to \mathsf{K}_a(\phi)$	16, Lemma 1

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

Proof.

1.	$\models \mathsf{P}_a(\phi) \to \mathsf{K}_a(\phi)$	Proposition 5
2.	$\models \mathsf{K}_a(\phi) \to \phi$	property of K
3.	$\models \mathsf{P}_a(\phi) \to \phi$	1, 2

[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

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

Proof.

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

Proof.

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

Bibliography

[AB05] M. Abadi and B. Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM*, 52(1), 2005.

[Aba00] M. Abadi. Security protocols and their properties. In *Foundations of Secure Computation*. IOS Press, 2000.

[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.

[ADM03] K. Adi, M. Debbabi, and M. Mejri. A new logic for electronic commerce protocols. *Theoretical Computer Science*, 291(3), 2003.

[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.

[AG99] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The Spi-calculus. *Information and Computation*, 148(1), 1999.

[AM01] L. C. Aiello and F. Massacci. Verifying security protocols as planning in logic programming. *ACM Transactions on Computational Logic*, 2(4), 2001.

[AN96] R. Anderson and R. Needham. Programming Satan's computer. In *Computer Science Today: Recent Trends and Developments*, volume 1000 of *LNCS*. Springer-Verlag, 1996.

[AN05] S. Artemov and E. Nogina. Introducing justification into epistemic logic. *Journal of Logic and Computation*, 15(6), 2005.

[AR02] M. Abadi and Ph. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2), 2002.

[ASW00] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4), 2000.

[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.

[Bal01] A. Baltag. Logics for insecure communication. In *Proceedings of the Conference on Theoretical Aspects of Rationality and Knowledge*, 2001.

[BAN90] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1), 1990.

[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.

[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.

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[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.

[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.

[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.

[Bie90] P. Bieber. A logic of communication in hostile environment. In *Proceedings of the IEEE Computer Security Foundations Workshop*, 1990.

[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.

[BM03] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer, 2003.

[BMN00] P. Bellini, R. Mattolini, and P. Nesi. Temporal logics for real-time system specification. *ACM Computing Surveys*, 32(1), 2000.

[BPS01] J. A. Bergstra, A. Ponse, and S. A. Smolka, editors. *Handbook of Process Algebra*. Elsevier, 2001.

[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.

[Can01] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 2001.

[CD05a] M. Cohen and M. Dam. A completeness result for BAN logic. In *Proceedings of the Workshop on Methods for Modalities*, 2005.

[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.

[CGP99] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 1999.

[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.

[Coh03] E. Cohen. First-order verification of cryptographic protocols. *Journal of Computer Security*, 11(2), 2003.

[CS97] T. Coffey and P. Saidha. Logic for verifying public-key cryptographic protocols. In *IEE Proceedings — Computers and Digital Techniques*, 1997.

[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.

[Dam89] M. F. Dam. *Relevance Logic and Concurrent Composition.* PhD thesis, University of Edinburgh, 1989.

[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.

[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.

[dG95] Ph. de Groote, editor. *The Curry-Howard Isomorphism.* Number 8 in Cahiers du centre de logique. Academia-Erasme, Louvain-la-Neuve (Belgique), 1995.

[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.

[Dij72] E. W. Dijkstra. The humble programmer. *Communications of the ACM*, 15(10), 1972.

[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.

[DY83] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(12), 1983.

[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.

[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.

[FHJ02] U. Frendrup, H. Hüttel, and J. N. Jensen. Modal logics for cryptographic processes. In *Electronic Notes in Theoretical Computer Science*, volume 68, 2002.

[FHMV95] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge.* MIT Press, 1995.

[Fit96] M. Fitting. *First-Order Logic and Automated Theorem Proving.* Springer-Verlag, 1996.

[GJM99] J. A. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Proceedings of CRYPTO*, 1999.

[GKWZ03] D.M. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. *Many-Dimensional Modal Logics: Theory and Applications.* Elsevier, 2003.

[GL91] O. Grumberg and D. E. Long. Model checking and modular verification. In *Proceedings of CONCUR*, 1991.

[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.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28(2), 1984.

[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.

[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.

[GMP92] J. Glasgow, G. Macewen, and P. Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3), 1992.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1), 1989.

[Gol01] O. Goldreich. *Foundations of Cryptography: Basic Tools.* Cambridge University Press, 2001.

[Gol04] O. Goldreich. *Foundations of Cryptography: Basic Applications.* Cambridge University Press, 2004.

[Gol05] O. Goldreich. *Foundations of Cryptography — A Primer.* now Publishers Inc., 2005.

[Gon92] L. Gong. A security risk of depending on synchronized clocks. *ACM SIGOPS Operating Systems Review*, 26(1), 1992.

[GSW06] D. Goldin, S. A. Smolka, and P. Wegner, editors. *Interactive Computation: The New Paradigm.* Springer-Verlag, 2006.

[Hal03] J. Halpern. *Reasoning about Uncertainty.* MIT Press, 2003.

[HJ05] Ch. Haack and A. Jeffrey. Timed Spi-calculus with types for secrecy and authenticity. In *Proceedings of CONCUR*, 2005.

[HO02] J. Halpern and K. O'Neill. Secrecy in multi-agent systems. In *Proceedings of the IEEE Computer Security Foundations Workshop*, 2002.

[HS04] 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.

[IK06] R. Impagliazzo and B. M. Kapron. Logics for reasoning about cryptographic constructions. *Journal of Computer and Systems Sciences*, 72(2), 2006.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the ACM Symposium on the Theory of Computation*, 1988.

[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.

[Kra04] S. Kramer. Cryptographic Protocol Logic. In *Proceedings of the LICS/ICALP-Affiliated Workshop on Foundations of Computer Security*, 2004.

[Kra06] S. Kramer. Timed Cryptographic Protocol Logic, 2006. presented at the Nordic Workshop on Programming Theory.

[KS06] M. Kurkowski and M. Srebrny. A quantifier-free first-order knowledge logic of authentication. *Fundamenta Informaticæ*, 72, 2006.

[KSW98] J. Kelsey, B. Schneier, and D. Wagner. Protocol interactions and the chosen protocol attack. In *Proceedings of the Workshop on Security Protocols*, 1998.

[LA05] G. Lowe and M. Auty. On a calculus for security protocol development. Technical report, Oxford University, 2005.

[Lam05] L. Lamport. Real time is really simple. Technical Report MSR-TR-2005-30, Microsoft Research, 2005.

[Low97] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of the Computer Security Foundations Workshop*, 1997.

[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.

[McL99] J. McLean. Twenty years of formal methods. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1999.

[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.

[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.

[Mil06] D. Miller. Representing and reasoning with operational semantics. In *Proceedings of the Joint International Conference on Automated Reasoning*, 2006. invited paper.

[MP84] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer, 1984.

[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.

[MvOV96] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[Nut97] D. Nute, editor. *Defeasible Denotic Logic*, volume 263 of *Synthese Library.* Kluwer, 1997.

[Pau98] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1), 1998.

[Rab81] M. O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, 1981.

[Riv90] R. L. Rivest. Cryptography. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity.* Elsevier, 1990.

[Rog04] Ph. Rogaway. On the role of definitions in and beyond cryptography. In *Proceedings of the Asian Computing Science Conference*, 2004.

[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.

[Sch99] S. Schneider. *Concurrent and Real-Time Systems.* Wiley, 1999.

[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.

[Sla06] B. H. Slater. Epsilon calculi. *Logic Journal of the Interest Group in Pure and Applied Logic*, 14(4), 2006.

[SP03] E. Sumii and B. C. Pierce. Logical relations for encryption. *Journal of Computer Security*, 11(4), 2003.

[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.

[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.

[Wan04] F. Wang. Formal verification of timed systems: A survey and perspective. *Proceedings of the IEEE*, 92(8), 2004.

[ZH04] C. Zhou and M. R. Hansen. *Duration Calculus: A Formal Approach to Real-Time Systems.* Springer-Verlag, 2004.

[ZHR91] C. Zhou, C. A. R. Hoare, and A. P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5), 1991.

[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.

Index