Logical Concepts in Cryptography

Simon Kramer

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

Abstract. This paper 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 paper 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, space, and time. Secondary concepts: knowledge de dicto and knowledge de re, confidentiality norms, truth-functional and relevant implication, multiple and complex truth values. The contribution of the paper is conceptual refinement and unification in a single, comprehensive logical language, namely CPL. We illustrate the expressiveness of CPL on representative *requirements engineering* case studies. The distinguishing feature of CPL is that it unifies and refines a variety of (not all!) existing approaches. This feature is the result of our *wholistic conception* of property-based (logics) and model-based (process algebra) formalisms.

Keywords applied formal logic, information security, logical modelling of cryptographic protocols, combination of modal logics and process algebra, requirements engineering

1 Introduction

The definition of a cryptographic protocol begins (and "ends" if this stage is not mastered[1]) 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 negations of necessary conditions for the requirements to hold. Protocol definition and in particular requirements engineering are engineering tasks (the spirit of [2]). In contrast, the definition of cryptographic operators is a scientific task (the spirit of [3, 4]) requiring profound expertise from different fields of discrete mathematics. Protocol engineers do (and should) not have

[1] "[...] 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." [1]

(to have) this expertise. For example, it is legitimate for a protocol engineer to "abstract" negligible probabilities and consider them as what they are — negligible. Ideally, engineers should only have to master a single, common, and formal language for requirements engineering that adequately abstracts "hard-core" mathematical concepts. 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. Our task shall be to synthesise the relevant logical concepts in cryptography into a cryptographic protocol logic in the tradition of temporal[2] logic [5] (cf. [6] for an effort of similar ambition but in the tradition of dynamic[3] logic [7]).

We briefly survey requirements engineering — the practice of the specification — of cryptographic protocols. Protocol designers commonly specify 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 intended *goals* (or *global* properties) in *natural language* [8]. Informal specifications present two major drawbacks: they do not have a well-defined, and thus a well-understood *meaning*; and 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 [9]; process calculi: CSP [10], applied π-Calculus [11], Spi-Calculus [12], and [13]). Examples of such encodings are equations between protocol instantiations, and predicates defined inductively on the traces those instantiations may exhibit [14]. However, such encodings present four major drawbacks: (1) they have to be found; 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. On the other hand, process calculi are ideal *design formalisms.* That is, they offer — due to their minimalist, linguistic representation of modelling concepts and their mathematical, operational semantics — a win-win situation between the (pedantic) rigour of machine models and the (practical) usability of programming languages.

Still, informal language and programming languages are inadequate for expressing and comparing cryptographic properties. It is our belief that only a logical 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 [15], a unification of several variants of BAN-logic [16]; over general-purpose propositional, modal, program, and first- and higher-order logics used for the special purpose of cryptographic protocol analysis: *propositional* ("logic programming") [17, 18]; *modal*: deontic [19], doxastic [20, 21], epistemic [22, 23], linear [24], temporal [25]; *program*: dynamic [26, 27,

[2] more precisely, poly-dimensional (i.e., norms, knowledge, space, time) mono-modal
[3] more precisely, mono-dimensional (i.e., time) poly-modal (action parameters)

6], Hoare-style [28]; *first-order* [29–31]; *higher-order* [32, 33]; to combinations thereof: doxastic-epistemic [34], doxastic-temporal [35], distributed temporal [36], dynamic-epistemic [37], epistemic-temporal [38] first-order-temporal [39], dynamic-epistemic-temporal [40], deontic-epistemic-temporal [41].

All these logics have elucidated important aspects of cryptographic communication and proved the relevance of logical concepts to the modelling of that communication. In particular, mere enunciation of maybe the three most fundamental cryptographic goals, 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[4] enunciations in natural language of these goals are the following (cf. Section 2.1 for their formalisations 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 protocol participant a w.r.t. participant 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; for the same reason (standard) deontic logic is inadequate (cf. Section 2.3). And doxastic logic is inadequate due to its inadequacy for the above goals as these crucially rely on knowledge, i.e., necessarily true, and not possibly false, belief (no control of the epistemic error). 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 concepts at the price of high extension costs.

Our goal is to supply a formal *synthesis* of logical concepts in a single, multi-dimensional[5] modal logic, namely CPL[6], that yields requirements that are *intuitive* but *abstract* w.r.t. particular models of cryptography. First, we believe that the formal method for any science is ultimately logic, as defined by a relation of satisfaction (*model*-theoretic approach[7], effectuated via *model checking* [45]) or a relation of deduction (*proof*-theoretic approach, effectuated via *automated theorem proving* [46]). Second, given that requirements engineering is mainly about meaning, i.e., understanding and formalising properties, we believe that

[4] as a matter of fact unique definitions of these goals do not exist (yet)

[5] cf. [42] for a research monograph on multi-dimensional modal logic (an active research area), characterised in [43] 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 [44]

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

a model-theoretic approach is — at least at a first stage — more suitable than a proof-theoretic approach. By 'intuitive' we mean that the conceptual dimensions of the requirement are apparent in distinctive forms in the formula that expresses the requirement — succinctly.

We argue that higher-order logic (at least beyond the second order) and set theory are unsuitable as front-end formalisms for requirements engineering purposes. They 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 are not just luxury, but the very — and difficult to distil — essence and a measure of its *pragmatics*, i.e., practical usefulness. The application domain of cryptographic protocols is conceptually very rich. A suitable requirements engineering formalism must hardwire and organise this conceptual variety in its semantics and provide succinct and intuitive linguistic abstractions (syntax) for them. 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 (back-ends) for such special-purpose formalisms (object logics). 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 [19]); propositional *knowledge* ("knows that"), i.e., knowledge of cryptographic states of affairs, (cf. epistemic logic [47]); execution *space* (cf. spatial logic [48]); and execution *time* (cf. temporal logic [5]). That is, CPL *unifies* first-order and four modal logics in a single, multi-dimensional 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 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 [49], in a novel way. First, CPL's temporal accessibility relation (the semantics of its temporal modalities) can be defined by an event-trace generating process (reduction) calculus, e.g., C^3 [50, 51] whose execution constraints are moreover checkable 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 process equivalence, which can be used for the model-based (process-algebraic and complementary to property-based) formulation of protocol requirements.

A cryptographic protocol involves the concurrent interaction of participants that are physically separated by — and exchange messages across — an unreliable and insecure transmission medium. Expressing properties of concurrent interaction requires temporal operators [5]. The physical separation by an unreliable and insecure transmission medium in turn demands the epistemic and deontic modalities. To see why, consider that the existence of such a separating medium introduces an *uncertainty* among protocol participants about the *trustworthiness* of the execution of protocol actions (sending and receiving) and the contents of exchanged messages, both w.r.t. *actuality* (an epistemic concern) and *legitimacy* (a deontic concern). It is exactly the role of a cryptographic protocol to re-establish 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) [2, 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 inner correctness even when composed with other protocols, i.e., a (totally) correct protocol should be *stable under different execution contexts* [52, 53].

2 Logic

2.1 Syntax

The language $\mathcal{F}$ of CPL is parametric in the language $\mathcal{M}$ of its individuals, i.e., protocol messages. It is chiefly relational, and functional in exactly the language $\mathcal{M}$ of protocol 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 $\mathcal{M}$ of individuals.

Definition 1 (Protocol messages). *Protocol messages $M \in \mathcal{M}$ have the following structure. $M ::= n$ (names, logical constants) $\big|$ ■ (the abstract message) $\big|$ p^+ (public keys) $\big|$ $\lceil M \rceil$ (message hashes) $\big|$ $\{\!|M|\!\}_M$ (symmetric message ciphers) $\big|$ $\{\!|M|\!\}^+_{p^+}$ (asymmetric message ciphers) $\big|$ $\{\!|M|\!\}^-_p$ (signed messages) $\big|$ (M, M) (message tuples).*

Names $n \in \boldsymbol{\mathcal{N}}$ are participant names $a, b \in \boldsymbol{\mathcal{P}}$, the (for the moment Dolev-Yao [54]) adversary's name `Eve`*, symmetric 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 $v \in \mathcal{V}$.*

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 6), 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 succinctness of statements about the structure of messages.

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

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

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 $\mathtt{P}_{\mathtt{Adv}} := \mathtt{P} \cup \mathtt{Adv}$, $\mathtt{K} := \mathtt{K}^1 \cup \mathtt{K}^\infty$, $\mathtt{CK} := \mathtt{K} \cup \mathtt{K}^-$, $\mathtt{K}^* := \mathtt{CK} \cup \mathtt{K}^+$, and $\mathtt{N} := \mathtt{P}_{\mathtt{Adv}} \cup \mathtt{K}^* \cup \mathtt{X}$.

Definition 3 (Formulae). *The set of formulae $\mathcal{F}$ contains precisely those propositions that are the closed predicates formed with the operators of Table 1. There, β denotes basic, α action, and δ data formulae; a, b, c denote participants and x, x' nonces; and k denotes a symmetric key and p a private key.*

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*.

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/necessarily implies", $\otimes$ "conjunctively separates", $\rhd$ "assume—guarantee", S "since", $\ominus$ "previous", $\oplus$ "next", and U "until", $a.x \circlearrowright x'$ "a

Table 1. Predicate language

$$
\begin{aligned}
\phi, \phi' &::= \beta \mid \neg\phi \mid \phi \wedge \phi' \mid \forall v(\phi) \\
&\mid \mathsf{P}\phi \mid \mathsf{K}_a(\phi) \mid \phi \supseteq \phi' \mid \phi \otimes \phi' \mid \phi \rhd \phi' \mid \phi \,\mathsf{S}\, \phi' \mid \ominus\phi \mid \oplus\phi \mid \phi \,\mathsf{U}\, \phi' \\
\beta, \beta' &::= \alpha \mid \delta \\
\alpha, \alpha' &::= a.x \circlearrowleft x' \mid a.x \circlearrowleft k.\{b,c\} \mid a.x \circlearrowleft p.b \\
&\mid a.x \xrightarrow[\mathsf{E}\not{\mathsf{v}}\mathsf{e}]{F} b \mid a.x \xrightarrow[\mathsf{Eve}]{F} b \mid a.x \xleftarrow[\mathsf{E}\not{\mathsf{v}}\mathsf{e}]{F} b \mid a.x \xleftarrow[\mathsf{Eve}]{} F \\
\delta, \delta' &::= n : \sigma \mid a \;\mathsf{k}\; F \mid F \preccurlyeq F'
\end{aligned}
$$

freshly generated the nonce x' in session x", $a.x \circlearrowleft k.\{b,c\}$ "a freshly generated the symmetric key k for b and c in session x", $a.x \circlearrowleft p.b$ "a freshly generated the private key p for b in session x", $a.x \xrightarrow[\mathsf{E}\not{\mathsf{v}}\mathsf{e}]{F} b$ "a securely (i.e., in a way unobservable by the adversary) sent off F *as such* (i.e., not only as a strict sub-term of another message) to b in session x", $a.x \xrightarrow[\mathsf{Eve}]{F} b$ "a insecurely (i.e., in a way observable by the adversary) sent off F as such to b in session x", $a.x \xleftarrow[\mathsf{E}\not{\mathsf{v}}\mathsf{e}]{F} b$ "a securely (i.e., not through the adversary) received F as such from b in session x", $a.x \xleftarrow[\mathsf{Eve}]{} F$ "a insecurely (i.e., possibly from the adversary) received F as such in session x", : "has type", k "knows", and $\preccurlyeq$ "is a subterm of".

Our language is *1-sorted* because protocol participants are referred to by their name and names are transmittable data, i.e., messages. K expresses knowledge *de dicto* (or *propositional* knowledge). In contrast, k expresses knowledge *de re* (or *individual* knowledge). Knowledge de re 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 participants to synthesise those pieces from previously analysed pieces. By 'understanding of the purpose' we mean (1) knowledge of the *structure* for compound, and (2) knowledge of the *identity* for atomic (names) information. Note that such understanding requires that there be a *minimal redundancy* in that information. The conditional $\phi \supseteq \phi'$ is epistemic or necessary 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 distribution/production) and the other satisfies ϕ' (e.g., key use/consumption). The formula $\phi \rhd \phi'$ is satisfied by a model if and only if for all models that satisfy ϕ the parallel composition of both models satisfies ϕ' (cf. total/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 the structure of messages. It is actually macro-definable in terms of typing of atomic data, equality (itself macro-definable), and existential quantification (cf. Appendix B).

We exemplify the expressiveness of CPL on a selection of tentative formalisations of fundamental cryptographic states of affairs. To the best of our knowledge, no other existing crypto logic is sufficiently expressive to allow for the *definition* of the totality of these properties. We invite the reader to validate our formalisations on the criteria of intuitiveness and succinctness. Note that the formalisations employ macro-defined predicates (cf. Appendix B) and that $\alpha(b)$ abbreviates disjunction of name generation, sending, and receiving performed by b.

Honesty b is *honest*, written $\mathsf{honest}(b)$, :iff b does never knowingly perform a forbidden action, written $\neg\diamondplus(\alpha(b) \wedge \mathsf{F}\alpha(b) \wedge \mathsf{K}_b(\mathsf{F}\alpha(b)))$.

Prudency b is *prudent*, written $\mathsf{prudent}(b)$, :iff b does never perform a forbidden action, written $\neg\diamondplus(\alpha(b) \wedge \mathsf{F}\alpha(b))$ or equivalently $\boxplus(\alpha(b) \to \mathsf{P}\alpha(b))$.

Trust a *trusts* b, written $a\ \mathtt{trusts}\ b$, :iff a knows that b is prudent, written $\mathsf{K}_a(\mathsf{prudent}(b))$.

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

Perfect Forward Secrecy "[. . .] compromise of long-term keys does not compromise past session keys." [2, Page 496], written $\neg\diamondplus(\exists(k : \mathtt{K}^1)(\mathtt{Eve}\ \mathsf{k}\ k) \supseteq \exists(k : \mathtt{K}^\infty)(\mathtt{Eve}\ \mathsf{k}\ k))$[8]

Anonymity b is anonymous to a in state of affairs $\phi(b)$:iff if a knows that some participant is involved in ϕ then a cannot identify that participant with b, written $\mathsf{K}_a(\exists(c : \mathtt{P})(\phi(c))) \to \neg\mathsf{K}_a(\phi(b))$.

Known-key attack "[. . .] an adversary obtains some keys used previously and then uses this information to determine new keys." [2, Page 41], written $\exists(v : \mathtt{CK})(\mathtt{Eve}\ \mathsf{k}\ v \wedge (\exists(v' : \mathtt{CK})(v' \neq v \wedge \mathtt{Eve}\ \mathsf{k}\ v') \supseteq \mathtt{Eve}\ \mathsf{k}\ v))$

Key confirmation for a w.r.t. b ". . . one party is assured that a second (possibly unidentified) party actually has possession of a particular secret key." [2, Page 492], written $k : \mathtt{K} \wedge \mathsf{K}_a(b\ \mathsf{k}\ k)$

Implicit key authentication for a w.r.t. b ". . . one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key." [2, Page 492], written $k : \mathtt{K} \wedge \mathsf{K}_a(\forall(c : \mathtt{P}_{\mathtt{Adv}})(c\ \mathsf{k}\ k \to (c = a \vee c = b)))$

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

[8] A material conditional would not do here because the antecedent and the consequent are *epistemically* — and thus not truth-functionally — *related* via key corruption, i.e., the *derivation* of a session key from a corrupted long-term key.

Non-repudiation of authorship M proves that b authored M' :iff assuming an arbitrary a knows M guarantees that a knows that b authored M', written $\forall(a:\mathtt{P})(a \;\mathsf{k}\; M \rhd \mathsf{K}_a(b \;\mathsf{a}\; M'))$.

Compositional protocol correctness protocol (plug-in) P in (initial) state $\mathfrak{h}$ satisfies property ϕ provided that P is used with a (parallel) protocol (environment) satisfying φ, written $(P,\mathfrak{h}) \models \varphi \rhd \phi$.

2.2 Semantics

Our definition of satisfaction is *anchored* (or *rooted*) and defined on protocol states, i.e., tuples $(P,\mathfrak{h}) \in \mathcal{P}\times\mathcal{H}$ of a protocol model P (i.e., a parallel-composable process term) and a protocol history $\mathfrak{h}$ (i.e., an event trace). For the purpose of this paper, we presuppose a notion of execution (e.g., [50]) $\longrightarrow \subseteq (\mathcal{P}\times\mathcal{H})\times(\mathcal{P}\times\mathcal{H})$ (or relation of *temporal accessibility* in the jargon of modal logic) producing protocol events and chaining them up to form protocol histories. Protocol events have the following form: generation of a nonce x' in session x by a, written $\mathtt{N}(a,x,x')$; generation of a fresh symmetric key k for b and c in session x by a, written $\mathtt{N}(a,x,k,(b,c))$; generation of a fresh private key p for b in session x by a, written $\mathtt{N}(a,x,p,b)$; insecure input of M in session x by a, written $\mathtt{I}(a,x,M)$; secure input of M from b in session x by a, written $\mathtt{sI}(a,x,M,b)$; insecure output of M to b in session x by a, written $\mathtt{O}(a,x,M,b)$; and secure output of M to b in session x 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.

Definition 4 (Protocol histories). *Protocol histories $\mathfrak{h}\in\mathcal{H}$ are simply finite words of protocol events ε, i.e., event traces:*

$$\mathfrak{h} \quad ::= \quad \epsilon \;\Big|\; \mathfrak{h}\cdot\varepsilon$$

where ϵ denotes the empty protocol history.

We define satisfaction in a functional style on the structure of formulae. Satisfaction employs *complex* (and thus multiple) truth values. Truth values are complex in the sense that they are tuples of a simple truth value (i.e., ‘true’ or ‘false’) and a set of those events (the evidence) that are necessary to corroborate that simple truth.

Definition 5 (Satisfaction). *Let $\models \subseteq (\mathcal{P}\times\mathcal{H})\times\mathcal{F}$ denote satisfaction of a formula $\phi\in\mathcal{F}$ by a protocol state $\mathfrak{s}\in\mathcal{P}\times\mathcal{H}$ (the anchor/root of an implicit execution path model for ϕ):*

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

where $\mathrm{paths}(\mathfrak{s})$ *denotes the set of paths* $\mathfrak{p}$ *achored/rooted in* $\mathfrak{s}$ *and induced by* $\longrightarrow$*, and* $[\![\cdot]\!]$ *denotes a function of truth denotation from formulae to complex truth values (cf. Table 2). There,*

- $\mathfrak{p}@i$ *denotes the state* $(P, \mathfrak{h})$ *at position* i *in path* $\mathfrak{p}$*.*
- $\dot{\mathfrak{h}}$ *denotes the set derived from protocol history* $\mathfrak{h}$*.*
- $\mathfrak{h} \vdash_a M$ *denotes the derivation of the individual knowledge* M *by participant* a *from* a*'s view on* $\mathfrak{h}$*, i.e., the extraction, analysis, and synthesis (cf. Appendix A) of the data that* a *has generated, received, or sent in* $\mathfrak{h}$*.*
- $\circ$ *denotes concatenation of protocol histories.*
- $\mathrm{E}_a^M(\mathfrak{h})$ *denotes the set of* earliest *events in* $\mathfrak{h}$ *that corroborate truth of the formula* $a \mathsf{~k~} M$*:*

$$\mathrm{E}_a^M(\mathfrak{h}) := \mathrm{E}_a^M(\mathfrak{h}, \epsilon)$$
$$\mathrm{E}_a^M(\mathfrak{h} \cdot \varepsilon, \mathfrak{h}') := \begin{cases} \mathrm{E}_a^M(\mathfrak{h}, \mathfrak{h}') & \textit{if } \mathfrak{h} \circ \mathfrak{h}' \vdash_a M, \textit{ and} \\ \mathrm{E}_a^M(\mathfrak{h}, \mathfrak{h}' \cdot \varepsilon) \cup \{\varepsilon\} & \textit{otherwise.} \end{cases}$$
$$\mathrm{E}_a^M(\epsilon, \mathfrak{h}') := \emptyset$$

- $\Sigma := \exists(k : \mathtt{CK})(\mathtt{Eve} \mathsf{~k~} k \land \neg\, k \mathsf{~ck~} \mathtt{Eve})$ *denotes a state formula expressing the* fundamental state of violation *in a cryptographic setting, namely the one where the adversary has come to know a confidential key not of her own.*
- $\approx_a \subseteq (\mathcal{P} \times \mathcal{H}) \times (\mathcal{P} \times \mathcal{H})$ *denotes the relation of* epistemic accessibility *associated with the modality* K_a*; it is defined hereafter.*
- $(\!|\cdot|\!)_a^{\cdot}$ *denotes a unary function (inspired by [55]) 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.*
- $\equiv$ *denotes a relation of* structural equivalence *defined on process terms and on event traces. On process terms, it denotes the smallest equivalence relation expressing associativity and commutativity of processes. On event traces, it denotes permutation, i.e.,* $\mathfrak{h} \equiv \mathfrak{h}'$ *:iff* $|\mathfrak{h}| = |\mathfrak{h}'|$ *and* $\dot{\mathfrak{h}} = \dot{\mathfrak{h}}'$*, where* $|\cdot|$ *denotes a length function.*

Permission is not macro-defined because we want to highlight that each new notion of fundamental state of violation will give rise to a new notion of permission. 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 (K_a) of CPL, but also as the definitional basis for the observational process equivalence of C^3 [50]. 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 6 (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 3.*

Table 2. Truth denotation

$$
\begin{aligned}
\llbracket a.x \circlearrowright x' \rrbracket^i_{\mathfrak{p}} &:= (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \{\mathtt{N}(a, x, x')\} \cap \dot{\mathfrak{h}} \\
\llbracket a.x \circlearrowright k.\{b, c\} \rrbracket^i_{\mathfrak{p}} &:= (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \{\mathtt{N}(a, x, k, \{b, c\})\} \cap \dot{\mathfrak{h}} \\
\llbracket a.x \circlearrowright p.b \rrbracket^i_{\mathfrak{p}} &:= (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \{\mathtt{N}(a, x, p, b)\} \cap \dot{\mathfrak{h}} \\
\llbracket a.x \xrightarrow[\mathtt{E\not{v}e}]{M} b \rrbracket^i_{\mathfrak{p}} &:= (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \{\mathtt{sO}(a, x, M, b)\} \cap \dot{\mathfrak{h}} \\
\llbracket a.x \xrightarrow[\mathtt{Eve}]{M} b \rrbracket^i_{\mathfrak{p}} &:= (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \{\mathtt{O}(a, x, M, b)\} \cap \dot{\mathfrak{h}} \\
\llbracket a.x \xleftarrow[\mathtt{E\not{v}e}]{M} b \rrbracket^i_{\mathfrak{p}} &:= (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \{\mathtt{sI}(a, x, M, b)\} \cap \dot{\mathfrak{h}} \\
\llbracket a.x \xleftarrow[\mathtt{Eve}]{} M \rrbracket^i_{\mathfrak{p}} &:= (\mathcal{E} \neq \emptyset, \mathcal{E}) \quad \text{where } \mathcal{E} := \{\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}} &:= (\mathfrak{h} \vdash_a M, \mathrm{E}^M_a(\mathfrak{h})) \\
\llbracket M \preccurlyeq M' \rrbracket^i_{\mathfrak{p}} &:= (M \text{ is a subterm of } M', \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, \textstyle\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 \rightarrow (\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{p}@i}_a, (\!| \phi |\!)^{\mathfrak{p}@i}_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 } \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 is } Q \in \mathcal{P} \text{ and } Q' \in \mathcal{P} \text{ s.t. } P \equiv Q \mathbin{|||} Q' \text{ and } (Q, \mathfrak{h}) \models_{\mathcal{E}_\phi} \phi \\
& \text{and } (Q', \mathfrak{h}) \models_{\mathcal{E}_{\phi'}} \phi', \, \mathcal{E}_\phi \cup \mathcal{E}_{\phi'}) \\
\llbracket \phi \rhd \phi' \rrbracket^i_{\mathfrak{p}} &:= (\text{for all } (Q, \mathfrak{h}') \in \mathcal{P} \times \mathcal{H} \text{ and } \mathfrak{h}'' \equiv \mathfrak{h}' \circ \mathfrak{h}, \text{ if } \ (Q, \mathfrak{h}') \models_{\mathcal{E}'} \phi \ \text{ then} \\
& (Q \mathbin{|||} P, \mathfrak{h}'') \models_{\mathcal{E}''} \phi', \textstyle\bigcup \mathcal{E}'' \cup \bigcup \mathcal{E}') \\
\llbracket \phi \mathbin{\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, \\
& \textstyle\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 \mathbin{\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, \\
& \textstyle\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{aligned}
$$

Table 3. Parsing on cryptographic messages

$$\llparenthesis\, \ulcorner M \urcorner \,\rrparenthesis_a^{\mathfrak{h}} := \begin{cases} \ulcorner \llparenthesis\, M \,\rrparenthesis_a^{\mathfrak{h}} \urcorner & \text{if } \mathfrak{h} \models a \;\mathsf{k}\; M\text{, and} \\ \blacksquare & \text{otherwise.} \end{cases}$$

$$\llparenthesis\, \{\!|M|\!\}_{M'} \,\rrparenthesis_a^{\mathfrak{h}} := \begin{cases} \{\!|\llparenthesis\, M \,\rrparenthesis_a^{\mathfrak{h}}|\!\}_{\llparenthesis\, M' \,\rrparenthesis_a^{\mathfrak{h}}} & \text{if } \mathfrak{h} \models a \;\mathsf{k}\; M'\text{, and} \\ \blacksquare & \text{otherwise.} \end{cases}$$

$$\llparenthesis\, \{\!|M|\!\}_{p^+}^{+} \,\rrparenthesis_a^{\mathfrak{h}} := \begin{cases} \{\!|\llparenthesis\, M \,\rrparenthesis_a^{\mathfrak{h}}|\!\}_{p^+}^{+} & \text{if } \mathfrak{h} \models a \;\mathsf{k}\; p \lor (a \;\mathsf{k}\; M \land a \;\mathsf{k}\; p^+)\text{, and} \\ \blacksquare & \text{otherwise.} \end{cases}$$

$$\llparenthesis\, \{\!|M|\!\}_{p}^{-} \,\rrparenthesis_a^{\mathfrak{h}} := \begin{cases} \{\!|\llparenthesis\, M \,\rrparenthesis_a^{\mathfrak{h}}|\!\}_{p}^{-} & \text{if } \mathfrak{h} \models a \;\mathsf{k}\; p^+\text{, and} \\ \blacksquare & \text{otherwise.} \end{cases}$$

$$\llparenthesis\, (M, M') \,\rrparenthesis_a^{\mathfrak{h}} := (\llparenthesis\, M \,\rrparenthesis_a^{\mathfrak{h}}, \llparenthesis\, M' \,\rrparenthesis_a^{\mathfrak{h}})$$

Definition 7 (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 participant 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 participant,*

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`,

4. $$\frac{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \hat{\varepsilon}(b) \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r} \; \mathtt{Eve} \neq b \qquad \frac{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \hat{\varepsilon}(b)} \; \mathtt{Eve} \neq b$$
5. $$\frac{\mathfrak{h}_l \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r}{\mathfrak{h}_l \cdot \mathtt{I}(b,x,M) \approx_{\mathtt{Eve}}^{(\mathfrak{h},\mathfrak{h}')} \mathfrak{h}_r \cdot \mathtt{I}(b,x,M')} \; \llparenthesis\, M \,\rrparenthesis_{\mathtt{Eve}}^{\mathfrak{h}} = \llparenthesis\, M' \,\rrparenthesis_{\mathtt{Eve}}^{\mathfrak{h}'}$$

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

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 agents a including $\mathtt{Eve}$, $\approx_a \subseteq \mathcal{H} \times \mathcal{H}$ 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 [47]
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 8 (Structurally indistinguishable protocol states). *Let* P_1 *and* P_2 *denote two cryptographic processes, i.e., models of cryptographic protocols, of some set* $\mathcal{P}$. *Then two protocol states* $(P_1, \mathfrak{h}_1)$ *and* $(P_2, \mathfrak{h}_2)$ *are* structurally indistinguishable from the viewpoint of *an agent* a, *written* $(P_1, \mathfrak{h}_1) \approx_a (P_2, \mathfrak{h}_2)$, *:iff* $\mathfrak{h}_1 \approx_a \mathfrak{h}_2$.

2.3 Discussion

In the terminology of relevant logics, both the spatial conditional $\rhd$ and the epistemic conditional $\supseteq$ are *relevant* (as opposed to *truth-functional*) 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 *hypothetical*) information is represented by the adjoint state $(Q, \mathfrak{h}')$. In $\supseteq$, the relevant (and *actual*) information is represented by the event subset $\mathcal{E}_{\phi'}$. As an example, consider

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

which states *what* primary knowledge, namely k, *is required* for $\mathtt{Eve}$ *to derive* the (secondary) knowledge M in the given model. In other words, if $\mathtt{Eve}$ *knew* k then $\mathtt{Eve}$ *would know* M in the given model. In contrast, consider

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

which states *how the* secondary *knowledge* M *can actually be derived* from the primary knowledge in a given model. In other words, if $\mathtt{Eve}$ knows M then *because* $\mathtt{Eve}$ knows k in the given model. We believe that $\rhd$ and $\supseteq$ are (perhaps *the*) two natural — and incidentally, relevant — notions of implication for cryptography.

A particularly interesting use of the spatial and the epistemic conditional is the definition of a cryptographically meaningful notion of permission (cf. Table 2) and prohibition (cf. Appendix B). Our definition says that it is permitted that ϕ is true if and only if if ϕ *were* true then whenever the fundamental state of violation *would be* reached, it *would not be* due to ϕ being true. This (reductionistic) notion of permission is inspired by [56, 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 [56] 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 [57]. Consider that it must be obligatory that (1) the fundamental state of violation be never reached during protocol execution, and (2) protocol participants 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 the fundamental state of violation.

Our semantics for the epistemic modality reconciles the cryptographically intuitive but incomplete semantics from [58] with the complete (but less computational), renaming semantics from [59]. We achieve this by casting the cryptographic intuition from [58] in a simple (rule-based) and computational formulation of epistemic accessibility. Similarly to [58], we parse unintelligible data in an agent's a *individual* knowledge M into abstract messages $\blacksquare$. In addition, and inspired by [60, 59], 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, i.e., the fact that an agent a knows all logical truths 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 $(P, \epsilon) \models \mathsf{K}_a(\exists v(\{\!|M|\!\}_k = \{\!|v|\!\}_k))$ "in" SEL but truthfully $(P, \epsilon) \not\models \mathsf{K}_a(\exists v(\{\!|M|\!\}_k = \{\!|v|\!\}_k))$ in CPL because $\models \neg\exists v(\blacksquare = \{\!|v|\!\}_k)$ (cf. "otherwise"-clause in the truth denotation of $\mathsf{K}_a(\phi)$ in Table 2). For further discussion see [60]. Note that our truth condition for the epistemic modality is an enhancement of the one from [60, 59] in the sense that we are able to 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., $\models \mathsf{K}_a(\phi) \to \phi$, due to the reflexivity of its associated accessibility relation.

3 Conclusion

We believe having accomplished with CPL an original synthesis of an unprecedented variety of logical concepts that are relevant to the logical modelling of cryptographic 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, (3) pioneered the application of spatial logic to cryptographic concerns, and (4) shown that cryptographically meaningful deontic modalities are definable with a combination of epistemic and spatial conditional. At present, we are extending CPL with weak real time for time stamps and timed keys. Then, we are planning to extend CPL with probability for propositional knowledge and computational complexity for individual knowledge. We would like to be able to express that an agent a knows that ϕ is true with probability p, written $\mathsf{K}_a^p(\phi)$, and that a knows M provided she has probabilistic polynomial time, or greater computing power, written $a\ \mathsf{k}_{\text{ppt}}^{\leq}\ M$ and $a\ \mathsf{k}_{\text{ppt}}^{>}\ M$ respectively. This would lead to a complexity-theoretic notion of propositional and individual knowledge for our logic (cf. [61] for a logic with a complexity-theoretic notion of possession). 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 proof system for CPL is also one of our a desiderata.

Acknowledgement I would like to thank Mika Cohen for our stimulating discussions about crypto logics and his constructive criticism of this paper.

References

1. Meadows, C.: Ordering from Satan's menu: a survey of requirements specification for formal analysis of cryptographic protocols. Science of Computer Programming **50**(3–22) (2003)
2. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1996)
3. Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press (2001)
4. Goldreich, O.: Foundations of Cryptography: Basic Applications. Cambridge University Press (2004)
5. Manna, Z., Pnueli, A.: The Temporal Logic of Reactive and Concurrent Systems: Specification. Springer (1984)
6. Durgin, N., Mitchell, J.C., Pavlovic, D.: A compositional logic for proving security properties of protocols. Journal of Computer Security **11**(4) (2003)
7. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press (2000)
8. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer (2003)
9. Sumii, E., Pierce, B.C.: Logical relations for encryption. Journal of Computer Security **11**(4) (2003)

10. Ryan, P., Schneider, S., Goldsmith, M., Lowe, G., Roscoe, B.: The Modelling and Analysis of Security Protocols: the CSP Approach. Addison-Wesley (2000)
11. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: Proceedings of the ACM Symposium on Principles of Programming Languages. (2001)
12. Abadi, M., Gordon, A.D.: A calculus for cryptographic protocols: The Spi-calculus. Information and Computation **148**(1) (1999)
13. Mitchell, J.C., Ramanathan, A., Scedrov, A., Teague, V.: A probabilistic polynomial-time process calculus for the analysis of cryptographic protocols. Theoretical Computer Science **353**(1–3) (2006)
14. Abadi, M.: Security protocols and their properties. In: Foundations of Secure Computation, IOS Press (2000)
15. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. ACM Transactions on Computer Systems **8**(1) (1990)
16. Syverson, P.F., van Oorschot, P.C.: A unified cryptographic protocol logic. CHACS 5540-227, Naval Research Laboratory, Washington D.C., USA (1996)
17. Aiello, L.C., Massacci, F.: Verifying security protocols as planning in logic programming. ACM Transactions on Computational Logic **2**(4) (2001)
18. Abadi, M., Blanchet, B.: Analyzing security protocols with secrecy types and logic programs. Journal of the ACM **52**(1) (2005)
19. Bieber, P., Cuppens, F.: Expression of Confidentiality Policies with Deontic Logic. In: Deontic Logic in Computer Science: Normative System Specification. John Wiley & Sons (1993)
20. Accorsi, R., Basin, D., Viganò, L.: Towards an awareness-based semantics for security protocol analysis. In: Proceedings of the Post-CAV Workshop on Logical Aspects of Cryptographic Protocol Verification. (2001)
21. Zhang, Y., Varadharajan, V.: A logic for modeling the dynamics of beliefs in cryptographic protocols. In: Proceedings of the Australasian Conference on Computer Science. (2001)
22. Syverson, P.F., Stubblebine, S.G.: Group principals and the formalization of anonymity. In: Proceedings of the World Congress On Formal Methods In The Development Of Computing Systems. (1999)
23. Halpern, J., O'Neill, K.: Secrecy in multi-agent systems. In: Proceedings of the IEEE Computer Security Foundations Workshop. (2002)
24. Bozzano, M., Delzanno, G.: Automatic verification of secrecy properties for linear logic specifications of cryptographic protocols. Journal of Symbolic Computation **38**(5) (2004)
25. Gray, J.W., McLean, J.D.: Using Temporal Logic to Specify and Verify Cryptographic Protocols (progress report). In: Proceedings of the IEEE Computer Security Foundations Workshop. (1995)
26. Frendrup, U., Hüttel, H., Jensen, J.N.: Modal logics for cryptographic processes. In: Electronic Notes in Theoretical Computer Science. Volume 68. (2002)
27. Adi, K., Debbabi, M., Mejri, M.: A new logic for electronic commerce protocols. Theoretical Computer Science **291**(3) (2003)
28. Lowe, G., Auty, M.: On a calculus for security protocol development. Technical report, Oxford University (2005)
29. Clarke, E., Jha, S., Marrero, W.: 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)

30. Selinger, P.: Models for an adversary-centric protocol logic. In: Proceedings of the Post-CAV Workshop on Logical Aspects of Cryptographic Protocol Verification. (2001)
31. Cohen, E.: First-order verification of cryptographic protocols. Journal of Computer Security **11**(2) (2003)
32. Paulson, L.C.: The inductive approach to verifying cryptographic protocols. Journal of Computer Security **6**(1) (1998)
33. Impagliazzo, R., Kapron, B.M.: Logics for reasoning about cryptographic constructions. Journal of Computer and Systems Sciences **72**(2) (2006)
34. Coffey, T., Saidha, P.: Logic for verifying public-key cryptographic protocols. In: IEE Proceedings — Computers and Digital Techniques. (1997)
35. Benerecetti, M., Giunchiglia, F., Panti, M., Spalazzi, L.: A logic of belief and a model checking algorithm for security protocols. In: Proceedings of FORTE. (2000)
36. Caleiro, C., Viganò, L., Basin, D.: Towards a metalogic for security protocol analysis. In: Proceedings of the Workshop on Combination of Logics. (2004)
37. Baltag, A.: Logics for insecure communication. In: Proceedings of the Conference on Theoretical Aspects of Rationality and Knowledge. (2001)
38. Dixon, C., Gago, M.C.F., M. Fisher, W.v.d.H.: Using temporal logics of knowledge in the formal verification of security protocols. In: Proceedings of the International Symposium on Temporal Representation and Reasoning. (2004)
39. Gnesi, S., Latella, D., Lenzini, G.: A BRUTUS logic for the Spi-Calculus. In: Proceedings of the IFIP Workshop on Issues in the Theory of Security. (2001)
40. Bieber, P.: A logic of communication in hostile environment. In: Proceedings of the IEEE Computer Security Foundations Workshop. (1990)
41. Glasgow, J., Macewen, G., Panangaden, P.: A logic for reasoning about security. ACM Transactions on Computer Systems **10**(3) (1992)
42. Gabbay, D., Kurucz, A., Wolter, F., Zakharyaschev, M.: Many-Dimensional Modal Logics: Theory and Applications. Elsevier (2003)
43. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press (2001)
44. Kramer, S.: Cryptographic Protocol Logic. In: Proceedings of the LICS/ICALP-Affiliated Workshop on Foundations of Computer Security. (2004)
45. Clarke, Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
46. Fitting, M.: First-Order Logic and Automated Theorem Proving. Springer-Verlag (1996)
47. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press (1995)
48. Dam, M.F.: Relevance Logic and Concurrent Composition. PhD thesis, University of Edinburgh (1989)
49. Bergstra, J.A., Ponse, A., Smolka, S.A., eds.: Handbook of Process Algebra. Elsevier (2001)
50. Borgström, J., Kramer, S., Nestmann, U.: Calculus of Cryptographic Communication. In: Proceedings of the LICS-Affiliated Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis. (2006)
51. Borgström, J., Grinchtein, O., Kramer, S.: Timed Calculus of Cryptographic Communication. In: Proceedings of the Workshop on Formal Aspects in Security and Trust. (2006)
52. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings of the IEEE Symposium on Foundations of Computer Science. (2001)

53. Backes, M., Pfitzmann, B., Waidner, M.: A universally composable cryptographic library. In: Proceedings of the ACM Conference on Computer and Communication Security. (2003)
54. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Transactions on Information Theory **29**(12) (1983)
55. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology **15**(2) (2002)
56. Meyer, J.J.C., Dignum, F.P.M., Wieringa, R.J.: 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)
57. Nute, D., ed.: Defeasible Denotic Logic. Volume 263 of Synthese Library. Kluwer (1997)
58. Abadi, M., Tuttle, M.R.: A semantics for a logic of authentication. In: Proceedings of the ACM Symposium of Principles of Distributed Computing. (1991)
59. Cohen, M., Dam, M.: A completeness result for BAN logic. In: Proceedings of the Workshop on Methods for Modalities. (2005)
60. Cohen, M., Dam, M.: Logical omniscience in the semantics of BAN logic. In: Proceedings of the LICS-affiliated Workshop on the Foundations of Computer Security. (2005)
61. Datta, A., Derek, A., Mitchell, J.C., Shmatikov, V., Turuani, M.: Probabilistic polynomial-time semantics for a protocol security logic. In: Proceedings of the EATCS International Colloquium on Automata, Languages and Programming. (2005)

A Auxiliary definitions

Data extraction

$$\frac{}{\mathfrak{h}\cdot\varepsilon(a,\ldots)\vdash_a (a,\ldots)} \qquad \frac{\mathfrak{h}\vdash_a M}{\mathfrak{h}\cdot\varepsilon\vdash_a M}$$

Data synthesis ***Data analysis***

$$\frac{\mathfrak{h}\vdash_a M \quad \mathfrak{h}\vdash_a M'}{\mathfrak{h}\vdash_a (M,M')} \qquad \frac{\mathfrak{h}\vdash_a (M,M')}{\mathfrak{h}\vdash_a M} \qquad \frac{\mathfrak{h}\vdash_a (M,M')}{\mathfrak{h}\vdash_a M'}$$

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

$$\frac{\mathfrak{h}\vdash_a M \quad \mathfrak{h}\vdash_a M'}{\mathfrak{h}\vdash_a \{\!|M|\!\}_{M'}} \qquad \frac{\mathfrak{h}\vdash_a \{\!|M|\!\}_{M'} \quad \mathfrak{h}\vdash_a M'}{\mathfrak{h}\vdash_a M}$$

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

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

B Specification library

$\top$	$:=$	$\mathtt{Eve} : \mathtt{Adv}$	true
$\bot$	$:=$	$\neg\top$	false
$\phi \vee \phi'$	$:=$	$\neg(\phi \wedge \phi')$	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 a v s.t. ϕ
$\forall(v:\theta)(\phi)$	$:=$	$\forall v(v:\theta \rightarrow \phi)$	
$\exists(v:\theta)(\phi)$	$:=$	$\exists v(v:\theta \wedge \phi)$	
$\mathsf{F}\phi$	$:=$	$\neg\mathsf{P}\phi$	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
$\boxminus\phi$	$:=$	$\phi \,\mathsf{S}\, \bot$	so far ϕ
$\diamondminus\phi$	$:=$	$\neg \boxminus \neg\phi$	once ϕ
$\mathsf{1}.\phi$	$:=$	$\phi \wedge \neg\ominus\diamondminus\phi$	for the first time ϕ
$\boxplus\phi$	$:=$	$\phi \,\mathsf{U}\, \bot$	henceforth ϕ
$\diamondplus\phi$	$:=$	$\neg \boxplus \neg\phi$	eventually ϕ
$\phi \leq \phi'$	$:=$	$(\phi \wedge \diamondplus\phi') \vee (\phi' \wedge \diamondminus\phi)$	ϕ before ϕ'
$\phi \rightsquigarrow \phi'$	$:=$	$(\phi \leftrightarrow \diamondplus\phi') \wedge (\phi' \leftrightarrow \diamondminus\phi)$	ϕ' is being caused by ϕ
$\phi \rightarrowtail \phi'$	$:=$	$\boxplus(\phi \rightsquigarrow \phi')$	ϕ causes ϕ'
$F = F'$	$:=$	$F \preccurlyeq F' \wedge F' \preccurlyeq F$	F is equal to F'
$F \prec F'$	$:=$	$F = F' \wedge \neg\, F \preccurlyeq F'$	F is a strict subterm of F'
$a \;\mathsf{h}\; F$	$:=$	$\exists v(F \preccurlyeq v \wedge a \;\mathsf{k}\; v)$	a has/possesses F
$a \;\mathsf{tk}\; F$	$:=$	$a \;\mathsf{h}\; F \wedge \neg\, a \;\mathsf{k}\; F$	a tacitly knows F
$F : \emptyset$	$:=$	$\bot$	
$F : \mathtt{H}[\theta]$	$:=$	$\exists(v:\theta)(F = \lceil v \rceil)$	
$F : \mathtt{SC}_k[\theta]$	$:=$	$\exists(v:\theta)(F = \{\!\vert v \vert\!\}_k)$	
$F : \mathtt{AC}_{p^+}[\theta]$	$:=$	$\exists(v:\theta)(F = \{\!\vert v \vert\!\}^{+}_{p^+})$	
$F : \mathtt{S}_p[\theta]$	$:=$	$\exists(v:\theta)(F = \{\!\vert v \vert\!\}^{-}_{p})$	
$F : \mathtt{T}[\theta,\theta']$	$:=$	$\exists(v:\theta)\exists(v':\theta')(F = (v,v'))$	
$F : \theta \cup \theta'$	$:=$	$F : \theta \vee F : \theta'$	
$F : \theta \cap \theta'$	$:=$	$F : \theta \wedge F : \theta'$	
$F : \theta \setminus \theta'$	$:=$	$F : \theta \wedge \neg\, F : \theta'$	
$F : \mathtt{M}$	$:=$	$\top$	
$F : \mathtt{SC}[\theta]$	$:=$	$\exists(v:\mathtt{K})(F : \mathtt{SC}_v[\theta])$	
$F : \mathtt{AC}[\theta]$	$:=$	$\exists(v:\mathtt{K}^+)(F : \mathtt{AC}_v[\theta])$	
$F : \mathtt{C}[\theta]$	$:=$	$F : \mathtt{SC}[\theta] \cup \mathtt{AC}[\theta]$	
$F : \mathtt{S}[\theta]$	$:=$	$\exists(v:\mathtt{K}^-)(F : \mathtt{S}_v[\theta])$	
$\theta \sqsubseteq \theta'$	$:=$	$\forall(v:\theta)(v:\theta')$	θ is a subtype of θ'

$\theta = \theta'$	$:=$	$\theta \sqsubseteq \theta' \wedge \theta' \sqsubseteq \theta$	
$\theta \sqsubset \theta'$	$:=$	$\theta \sqsubseteq \theta' \wedge \theta' \neq \theta$	
$k \mathrel{\varepsilon\!-} F$	$:=$	$\exists v(\{\!\vert v\vert\!\}_k \preccurlyeq F)$	
$p^+ \mathrel{\varepsilon\!-}^+ F$	$:=$	$\exists v(\{\!\vert v\vert\!\}^+_{p^+} \preccurlyeq F)$	
$p \mathrel{\varepsilon\!-}^- F$	$:=$	$\exists v(\{\!\vert v\vert\!\}^-_p \preccurlyeq F)$	
$n \mathrel{\varepsilon\!-}^* F$	$:=$	$n \mathrel{\varepsilon\!-} F \vee n \mathrel{\varepsilon\!-}^+ F \vee n \mathrel{\varepsilon\!-}^- F$	n is operational in F
$F \mathrel{-\!\!3} F'$	$:=$	$\exists v \exists (k : \mathtt{K})(F \preccurlyeq v \wedge \{\!\vert v\vert\!\}_k \preccurlyeq F')$	
$F \mathrel{-\!\!3}^+ F'$	$:=$	$\exists v \exists (p^+ : \mathtt{K}^+)(F \preccurlyeq v \wedge \{\!\vert v\vert\!\}^+_{p^+} \preccurlyeq F')$	
$F \mathrel{-\!\!3}^- F'$	$:=$	$\exists v \exists (p : \mathtt{K}^-)(F \preccurlyeq v \wedge \{\!\vert v\vert\!\}^-_p \preccurlyeq F')$	
$F \mathrel{-\!\!3}^* F'$	$:=$	$F \mathrel{-\!\!3} F' \vee F \mathrel{-\!\!3}^+ F' \vee F \mathrel{-\!\!3}^- F'$	F is guarded in F'
$n \mapsto x$	$:=$	$\Diamond\!\!\!\!-\,1.\exists m(n \mathrel{\varepsilon\!-}^* m \wedge \exists(a, b : \mathtt{P})(a.x \xrightarrow[\mathtt{Eve}]{m} b))$	n is for session x
$a \mathrel{\mathsf{a}} F$	$:=$	$\Diamond\!\!\!\!-\,(a \mathrel{\mathsf{k}} F \wedge \forall(b : \mathtt{P}_{\mathtt{Adv}})(b \mathrel{\mathsf{k}} F \rightarrow b = a))$	a authored F
$k \mathrel{\mathsf{sk}} a$	$:=$	$\exists(b, c : \mathtt{P}_{\mathtt{Adv}})\exists(x : \mathtt{X})(b.x \circlearrowright k.\{a, c\} \vee$ $b.x \circlearrowright k.\{c, a\})$	k is a shared key for a
$k \mathrel{\mathsf{sk}^1} a$	$:=$	$k \mathrel{\mathsf{sk}} a \wedge k : \mathtt{K}^1$	k is a session 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
$n \mathrel{\mathsf{prk}} a$	$:=$	$\exists(b : \mathtt{P})\exists(x : \mathtt{X})(b.x \circlearrowright n.a)$	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 private key for a
$n \mathrel{\mathsf{ck}} a$	$:=$	$n \mathrel{\mathsf{sk}} a \vee n \mathrel{\mathsf{prk}} a$	n is a confid. key for a