Formal Proofs of Cryptographic Security of Diffie-Hellman-based Protocols *

Arnab Roy[1], Anupam Datta[2], John C. Mitchell[1]

[1] Stanford University, Stanford, CA
{arnab, mitchell}@cs.stanford.edu
[2] Carnegie Mellon University, Pittsburgh, PA
danupam@cmu.edu

Abstract. We present axioms and inference rules for reasoning about Diffie-Hellman-based key exchange protocols and use these rules to prove authentication and secrecy properties of two important protocol standards, the Diffie-Hellman variant of Kerberos, and IKEv2, the revised standard key management protocol for IPSEC. The new proof system is sound for an accepted semantics used in cryptographic studies. In the process of applying our system, we uncover a deficiency in Diffie-Hellman Kerberos that is easily repaired.

1 Introduction

Diffie-Hellman key exchange (DHKE) is one of the earliest public-key concepts [28]. It allows two parties without a prior shared secret to jointly create one that is independent of past and future keys, and is therefore widely used in many network security protocols. In this paper, we develop axioms for reasoning about protocols that use Diffie-Hellman key exchange, prove these axioms sound using cryptographic reduction arguments, and use the axiom system to formally prove authentication and secrecy theorems for two significant standardized protocols. The two protocols we consider are Diffie-Hellman Key Exchange for initial authentication in Kerberos V5 [43] (which we refer to as DHINIT) and IKEv2 [34], the IPSEC key exchange standard. Kerberos is widely used in Microsoft Windows networking and other applications, while IKEv2 is part of IPSEC which is widely used for virtual private networks. The authentication and secrecy theorems, for probabilistic polynomial-time execution and standard cryptographic protocol attacks, have not been proved before to the best of our knowledge. In analyzing DHINIT, we also discover that the KAS is *not* authenticated to the client after the first stage, but we are able to prove formally in our logic that authentication is nonetheless achieved at a later stage; we also suggest a change to the protocol to ensure authentication after the first stage. In analyzing IKEv2,

* This work was partially supported by the NSF Science and Technology Center TRUST and U.S. Army Research Office contract on Perpetually Available and Secure Information Systems (DAAD19-02-1-0389) to CMU's CyLab.

which replaces the controversial[3] Internet Key Exchange (IKEv1) protocol using concepts from an intermediate protocol called Just Fast Keying (JFK) [3], we consider the IKEv2 mode in which signatures are used for authentication and Diffie-Hellman exponentials are never reused.

The axioms presented in this paper are used in Protocol Composition Logic (PCL) [24, 26, 41, 25, 39]. Our formalization uses the characterization of "good key" from [27], but improves on previous work in several respects: (i) we fix a bug in the **DH** axiom in [27] by using the "DHStrongSecretive" formulas developed in the paper, (ii) we present a general inductive method for proving secrecy conditions for Diffie-Hellman key exchange, and (iii) we present axioms for reasoning from ciphertext integrity assumptions. These three innovations are essential for the formal proofs for DHINIT and IKEv2, which could not be carried out in the system of [27]. In addition, the present soundness proofs are based on a new cryptographic definition and associated theorems about the joint security of multiple encryption schemes keyed using random or DHKE-keys. This paper complements [39] and completes the development of formal cryptographically sound proofs for three modes of Kerberos V5 ([42] contains technical details).

Most demonstrated approaches for proving security of complex network protocols, of the scale that appear in IEEE and IETF standards, use a simplified model of protocol execution based on symbolic computation and highly idealized cryptography [9, 16, 19, 24]. However, proofs about symbolic computation do not provide the same level of assurance as proofs about probabilistic polynomial-time attacks. Several groups of researchers have therefore developed methods for deriving cryptographic meaning from properties of symbolic protocol execution [7, 6, 18, 22, 31, 32, 38]. These methods involve showing that the behavior of a symbolic abstraction, under symbolic attacks, yields the same significant failures as a finer-grained execution under finer-grained probabilistic polynomial-time attack. However, such equivalence theorems rely on strong cryptographic assumptions, and there are no known suitable symbolic abstractions of Diffie-Hellman exponentiation. In addition, there are theoretical negative results that suggest that correspondence theorems may be impossible for symmetric encryption if a protocol might reveal a secret key [17, 23], or for hash functions or exclusive-or [5, 8]. In contrast, computational PCL reasons directly about properties of probabilistic polynomial-time execution of protocols, under attack by a probabilistic polynomial-time adversary, without explicit formal reasoning about probability or complexity. In addition, different axioms depend on different cryptographic assumptions, allowing us to consider which assumptions are actually necessary for each property we establish. As currently formulated in the RFC, Kerberos requires a party to sign only its own Diffie-Hellman exponential. We prove this is sufficient, using axioms that depend on the *random oracle* assumption [12]. However, we are not able to give a formal proof using alternate axioms that

[3] In the proceedings version of this paper we had used the phrase "seriously flawed". We acknowledge that this can be interpreted as a criticism about the cryptographic core of the protocol, which was not our intention. The controversy was with the complete protocol, *not* the cryptographic core itself.

do not depend on random oracles. On the other hand, the alternate axioms are sufficient to prove authentication if we modify the protocol slightly so that the KAS signs both the Diffie-Hellman exponentials, as is done in IKEv2 and JFK.

Two related studies are a symbolic proof for Kerberos (without DHKE) [4] and a cryptographic reduction proof for JFK [3]. In the Kerberos analysis, a correspondence between symbolic computation and cryptographic models [7] is used to draw cryptographic conclusions. This requires a separate verification that a "commitment problem" does not occur in the protocol (see [4]), and does not extend to Diffie-Hellman. The JFK proof is interesting and informative, with suggestions in [3] that "analysis based on formal methods would be a useful complement," but simpler than the proof of DHINIT since JFK digitally signs Diffie-Hellman values differently. More generally, Abadi and Rogaway [1] initiated computationally sound symbolic analysis of static equivalence, with extensions and completeness explored in [37, 2]; a recent extension to Diffie-Hellman appears in [15], covering only *passive adversaries,* not the stronger active adversaries used in the present paper. Protocol Composition Logic [24] was used in a case study of 802.11i [29], has previous computational semantics [26], and was used to study protocol composition and key exchange [27]. In other studies of DHKE, [30] uses a symbolic model, while [36] imposes nonstandard protocol assumptions. The cryptographic primitives used in Kerberos are analyzed in [14].

Section 2 summarizes Protocol Composition Logic (PCL), with section 3 presenting the proof system and computational soundness theorem. Kerberos DHINIT and IKEv2 are analyzed in sections 4 and 5, respectively. Conclusions are in section 6.

2 Background

This section contains a brief summary of aspects of Protocol Composition Logic (PCL) used in the rest of this paper. Additional background appears in [24, 26, 41, 25, 39].

Modelling protocols A protocol is given by a set of roles, each specifying a sequence of actions to be executed by an honest agent. Protocol roles may be written using a process language in which each role defines a sequential process, and concurrency arises as a consequence of concurrent execution of any number of instances of protocol roles. The set of role actions include generating a new nonce, signing or encrypting a messages, communicating over the network, and decrypting or verifying a signature through pattern matching. A role may depend on so-called input parameters, such as the intended recipient of messages sent by an instance of the role, or the recipient's public encryption key. An example protocol is presented in Section 4.

Protocol execution may be characterized using probabilistic polynomial-time oracle Turing machines [13]. In this approach, an initial configuration is defined by choosing a number of principals (agents), assigning one or more role instances to each principal, designating some subset of the principals as honest, and choosing encryption keys as needed. Protocol execution then proceeds by allowing a

probabilistic polynomial-time adversary to control protocol execution by interacting with honest principals (as oracles). This gives the adversary complete control over the network, but keys and random nonces associated with honest parties are not given directly to the adversary unless they are revealed in the course of protocol execution.

Each protocol, initial configuration, and choice of probabilistic polynomial-time adversary gives rise to a probability distribution on polynomial-length executions. A *trace* records all actions executed by honest principals and the attacker during one execution (run) of the protocol. Since honest principals execute roles defined by symbolic programs, we may define traces so that they record symbolic descriptions of the actions of honest parties and a mapping of symbolic variables to bitstrings values manipulated by the associated Turing machine. Since an attacker is not given by a symbolic program, a trace only records the send-receive actions of the attacker, not its internal actions. Traces also include the random bits (used by the honest parties, the adversary and available to an additional probabilistic polynomial-time algorithm called the distinguisher), as well as a few other elements used in defining semantics of formulas over collections of traces [26].

Protocol logic, proof system, cryptographic soundness The syntax of PCL and the informal descriptions of the principal predicates are given in [25, 39]. Most protocol proofs use formulas of the form $\theta[P]_X\phi$, which are similar to Hoare triples. Informally, $\theta[P]_X\phi$ means that after actions P are executed by the thread X, starting from any state where formula θ is true, formula ϕ is true about the resulting state. Formulas θ and ϕ typically combine assertions about temporal order of actions (useful for stating authentication) and assertions about knowledge (useful for stating secrecy).

Intuitively, a formula is true about a protocol if, as we increase the security parameter and look at the resulting probability distributions on traces, the probability of the formula failing is negligible (i.e., bounded above by the reciprocal of any polynomial). We may define the meaning of a formula φ on a set T of equi-probable computational traces as a subset $T' \subseteq T$ that respects φ in some specific way. For example, an action predicate such as Send selects a set of traces in which a send occurs (by the indicated agent). More precisely, the semantics $[\![\varphi]\!]\,(T, D, \epsilon)$ of a formula φ is inductively defined on the set T of traces, with distinguisher D and tolerance ϵ. The distinguisher and tolerance are only used in the semantics of the secrecy predicates Indist and $\mathsf{GoodKeyAgainst}$, where they determine whether the distinguisher has more than a negligible chance of distinguishing the given value from random or winning an IND-CCA game (standard in the cryptographic literature), respectively. We say a protocol Q satisfies a formula φ, written $Q \models \varphi$ if, for all adversaries and sufficiently large security parameters, the *probability* that φ "holds" is asymptotically overwhelming. A precise inductive semantics is given in [26].

Protocol proofs are written using a formal proof system, which includes axioms and proof rules that capture essential properties of cryptographic primitives such as signature and encryption schemes. In addition, the proof system incorpo-

rates axioms and rules for first-order reasoning, temporal reasoning, knowledge, and a form of inductive invariant rule called the "honesty" rule. The induction rule is essential for combining facts about one role with inferred actions of other roles. An axiom about a cryptographic primitive is generally proved sound by a cryptographic reduction argument that relies on some cryptographic assumption about that primitive. As a result, the mathematical import of a formal proof in PCL may depend on a set of cryptographic assumptions, namely those assumptions required to prove soundness of the actual axioms and rules that are used in the proof. In some cases, there may be different ways to prove a single PCL formula, some relying on one set of cryptographic assumptions, and another proof relying on another set of cryptographic assumptions.

3 Proof System

Section 3.1 contains new axioms and rules for reasoning about Diffie-Hellman key exchange. Section 3.2 summarizes the concept of *secretive protocol* and proof rules taken from [39] that are used in this paper to establish secrecy properties. However, we give new soundness proofs for these axioms, based on an extension of standard multiparty encryption schemes [10] to allow for multiple public and symmetric encryption schemes keyed using random or Diffie-Hellman based keys. The associated cryptographic definitions and theorems are presented in Section 3.3.

3.1 Diffie-Hellman Axioms

In this section we formalize reasoning about how individual threads treat DH exponentials in an appropriate way. We introduce the predicate $\mathsf{DHGood}(X, m, x)$, where x is a nonce used to compute a DH exponential, to capture the notion that thread X only uses certain safe actions to compute m from values that it has generated or received over the network. For example, axioms **DH2** and **DH3** say that a message m is DHGood if it has just been received, or if it is just computed by exponentiating the known group generator g with the nonce x. Axiom **DH4** states that the pair of two DHGood terms is also DHGood.

Note that unlike the symbolic model, it is not well defined in the computational model to say "m contains x". That is why our proof systems for secrecy in the symbolic model [41] and computational model [39] are different - the computational system does induction on actions rather than structure of terms. The need to look at the structure of m is obviated by the way the reduction to games like IND-CCA works. The high level intuition is that a consistent simulation of the protocol can be performed while doing the reduction, if the terms to be sent

to the adversary are "good".

DH0 $\mathsf{DHGood}(X, a, x)$, for a of any atomic type, except nonce, *viz.* name or key

DH1 $\mathsf{New}(Y, n) \wedge n \neq x \supset \mathsf{DHGood}(X, n, x)$

DH2 $[\texttt{receive}\ m;]_X\ \mathsf{DHGood}(X, m, x)$

DH3 $[m := \texttt{expg}\ x;]_X\ \mathsf{DHGood}(X, m, x)$

DH4 $\mathsf{DHGood}(X, m_0, x) \wedge \mathsf{DHGood}(X, m_1, x)\ [m := m_0.m_1;]_X\ \mathsf{DHGood}(X, m, x)$

DH5 $\mathsf{DHGood}(X, m, x)\ [m' := \texttt{symenc}\ m, k;]_X\ \mathsf{DHGood}(X, m', x)$

DH6 $\mathsf{DHGood}(X, m, x)\ [m' := \texttt{hash}\ m;]_X\ \mathsf{DHGood}(X, m', x)$

The formula $\mathsf{SendDHGood}(X, x)$ indicates that thread X sent out only "DH-Good" terms w.r.t. the nonce x. $\mathsf{DHSecretive}\ (X, Y, k)$ means that there exist nonces x, y such that threads X, Y respectively generated them, sent out "DH-Good" terms and X generated the key k from g^{xy}. $\mathsf{DHStrongSecretive}(X, Y, k)$ asserts a stronger condition - that threads X and Y only used each other's DH exponentials to generate the shared secret (The predicate $\mathsf{Exp}(X, gx, y)$ means thread X exponentiates gx to the power y). The formula $\mathsf{SharedSecret}(X, Y, k)$ means that the key k satisfies IND-CCA key usability against any thread other than X or Y, particularly against any adversary. Formally,

$$\mathsf{SendDHGood}(X, x) \equiv \forall m.\ \mathsf{Send}(X, m) \supset \mathsf{DHGood}(X, m, x)$$

$$\mathsf{DHSecretive}(X, Y, k) \equiv \exists x, y.\ \mathsf{New}(X, x) \wedge \mathsf{SendDHGood}(X, x) \wedge \mathsf{New}(Y, y) \wedge \mathsf{SendDHGood}(Y, y) \wedge \mathsf{KeyGen}(X, k, x, g^y)$$

$$\mathsf{DHStrongSecretive}(X, Y, k) \equiv \exists x, y.\ \mathsf{New}(X, x) \wedge \mathsf{SendDHGood}(X, x) \wedge \mathsf{New}(Y, y) \wedge \mathsf{SendDHGood}(Y, y) \wedge \mathsf{KeyGen}(X, k, x, g^y) \wedge (\mathsf{Exp}(X, gy, x) \supset gy = g^y) \wedge (\mathsf{Exp}(Y, gx, y) \supset gx = g^x)$$

$$\mathsf{SharedSecret}(X, Y, k) \equiv \forall Z.\ \mathsf{GoodKeyAgainst}(Z, k) \vee Z = X \vee Z = Y$$

The following axioms hold for the above definition of SendGood:

SDH0 $\mathsf{Start}(X) \supset \mathsf{SendDHGood}(X, x)$

SDH1 $\mathsf{SendDHGood}(X, x)\ [\mathsf{a}]_X\ \mathsf{SendDHGood}(X, x)$, where a is not a send action

SDH2 $\mathsf{SendDHGood}(X, x)\ [\texttt{send}\ m;]_X\ \mathsf{DHGood}(X, m, x) \supset \mathsf{SendDHGood}(X, x)$

The following axioms relate the $\mathsf{DHStrongSecretive}$ property, which is trace based, to computational notions of security. The first axiom, which depends on the DDH (Decisional Diffie-Hellman) assumption and IND-CCA security of the encryption scheme, states a secrecy property - if threads X and Y are DHStrongSecretive w.r.t. the key k, then k satisfies IND-CCA key usability. The second axiom, which depends on the DDH assumption and INT-CTXT (ciphertext integrity [11, 33]) security of the encryption scheme, states that with the same DHStrongSecretive property, if someone decrypts a term with the key k successfully, then it must have been encrypted with the key k by either X or Y. Both the axioms are proved sound by cryptographic reductions to the primitive security games.

DH $\mathsf{DHStrongSecretive}(X, Y, k) \Rightarrow \mathsf{SharedKey}(X, Y, k)$

CTXGS $\mathsf{DHStrongSecretive}(X, Y, k) \wedge \mathsf{SymDec}(Z, E_{sym}[k](m), k) \supset \mathsf{SymEnc}(X, m, k) \vee \mathsf{SymEnc}(Y, m, k)$

If the weaker property $\mathsf{DHSecretive}(X, Y, k)$ holds then we can establish an axiom similar to **CTXGS**, but we have to model the key generation function as a random oracle and the soundness proof (presented in [42]) is very different. The intuition behind this requirement is that if the threads do not use each other's intended DH exponentials then there could, in general, be related key attacks; the random oracle obviates this possibility.

$$\textbf{CTXG} \quad \mathsf{DHSecretive}(X, Y, k) \wedge \mathsf{SymDec}(Z, E_{sym}[k](m), k) \supset \mathsf{SymEnc}(X, m, k) \vee \mathsf{SymEnc}(Y, m, k)$$

The earlier paper [27] overlooked the subtle difference between the DHStrongSecretive and DHSecretive predicates. Specifically, in order to prove the axiom **DH** sound without the random oracle model, it is necessary to ensure that both parties use only each other's DH exponentials to generate keys—a condition guaranteed by DHStrongSecretive, but not DHSecretive or the variant considered in [27].

To provide some sense of the soundness proofs, we sketch the proof for the **CTXGS** axiom. The axiom is sound if the set (given by the semantics) $[\![\mathsf{DHStrongSecretive}(X, Y, k) \wedge \mathsf{SymDec}(Z, E_{sym}[k](m), k) \supset \mathsf{SymEnc}(X, m, k) \vee \mathsf{SymEnc}(Y, m, k)]\!](T, D, \epsilon)$ includes almost all traces in the set T generated by any probabilistic poly-time adversary $\mathcal{A}$. Assume that this is not the case: Let E be the event that an honest principal decrypts a ciphertext c with the key k such that c was not produced by X or Y by encryption with the key k; there exists an adversary $\mathcal{A}$ who forces E to occur in a non-negligible number of traces. Using $\mathcal{A}$, we will construct an adversary $\mathcal{A}'$ who breaks DDH, thereby arriving at a contradiction.

Suppose $\mathcal{A}'$ is given a DDH instance (g^a, g^b, g^c). It has to determine whether $c = ab$. Let the DH nonces used by X, Y be x, y respectively. $\mathcal{A}'$ simulates execution of the protocol to $\mathcal{A}$ by using g^a, g^b as the computational representations of g^x, g^y respectively. Whenever a symbolic step ($k' :=$ `dhkeygen` $m, x;$) comes up, $\mathcal{A}'$ behaves in the following manner: since $\mathsf{DHStrongSecretive}(X, Y, k)$ holds, m has to be equal to g^b, then k' is assigned the value g^c; Likewise for the action ($k' :=$ `dhkeygen` $m, y;$). After the protocol simulation, if the event E has occurred then output "$c = ab$", otherwise output "$c \neq ab$". The advantage of $\mathcal{A}'$ in winning the DDH game is:

$$\mathbf{Adv}^{\mathrm{DDH}}(\mathcal{A}') = \Pr[E|c = ab] - \Pr[E|c \neq ab]$$

By the assumption about $\mathcal{A}$, the first probability is non-negligible. The second probability is negligible because the encryption scheme is INT-CTXT secure. Hence the advantage of $\mathcal{A}'$ in breaking DDH is non-negligible. The SendDHGood predicate that DHStrongSecretive implies, ensures that the protocol simulation can be carried out consistently. Intuitively, this is ensured as long as the protocol simulator has to manipulate received messages, g^x, g^y (but not x, y directly) and key messages with g^{xy} to construct messages to be sent out. Axioms $\mathbf{DH0} - \mathbf{6}$ are used to formally establish that the protocol has this property.

3.2 Secretive Protocols

In this section, we adapt the concept of *secretive protocol,* a trace-based condition implying computational secrecy [40, 39], to permit keys generated from DHKE. While the proof rules remain identical, the soundness proofs are significantly different and involve a reduction to a multi-scheme IND-CCA game that we introduce in Section 3.3 of this paper. This definition allows the use of multiple encryption schemes keyed using randomly generated keys or keys output from a DHKE. A secretive protocol with respect to a nonce s and set of keys $\mathcal{K}$ is a protocol which generates *secretive traces*, defined below, with overwhelming probability.

Definition 1 (Secretive Trace). *A trace is a* secretive trace *with respect to s and $\mathcal{K}$ if the following properties hold for every thread belonging to honest principals:*

- *a thread which generates nonce s, ensures that it is encrypted with a key k in the set $\mathcal{K}$ in any message sent out.*
- *whenever a thread decrypts a message with a key k in $\mathcal{K}$, which was produced by encryption with key k by an honest party, and parses the decryption, it ensures that the results are encrypted with some key k' with $k' \in \mathcal{K}$ in any message sent out.*

To account for DH keys in the set $\mathcal{K}$, we wish to establish that DH keys are used in a "safe" manner by the protocol, formally captured by the predicate $\mathsf{DHStrongSecretive}$. Following [39], the predicate $\mathsf{Good}(X, m, s, \mathcal{K})$ asserts that the thread X constructed the term m in accordance with the rules allowing a *secretive protocol* with respect to nonce s and set of keys $\mathcal{K}$ to send out m. The formula $\mathsf{SendGood}(X, s, \mathcal{K})$ asserts that all messages that thread X sends out are good and $\mathsf{Secretive}(s, \mathcal{K})$ asserts that all honest threads only send out good messages. The axioms characterizing these predicates are same as in [39] and are omitted here. The induction rule $\mathbf{IND}_{GOOD}$ states that if all honest threads executing some basic sequence (i.e. a fragment of a role pausing before the next receive, denoted P) in the protocol (denoted $\mathcal{Q}$) locally construct good messages to be sent out, given that they earlier also did so, then we can conclude $\mathsf{Secretive}(s, \mathcal{K})$. A set of basic sequences (BS) of a role is any partition of the sequence of actions in a role such that if any element sequence has a `receive` then its only at its begining.

$$\mathbf{IND}_{GOOD} \quad \forall \rho \in \mathcal{Q}. \forall P \in BS(\rho). \quad \frac{\mathsf{SendGood}(X, s, \mathcal{K})\ [P]_X\ \Phi \supset \mathsf{SendGood}(X, s, \mathcal{K})}{\mathcal{Q} \vdash \Phi \supset \mathsf{Secretive}(s, \mathcal{K})}\ (*)$$

$(*)$: $[P]_X$ does not capture free variables in Φ, $\mathcal{K}$, s, and Φ is a prefix closed trace formula.

Now we relate the concept of a secretive protocol, which is trace-based, to complexity theoretic notions of security. We define a level-0 key to be either a pre-shared secret, a public key or a DH Key. To apply the results here the DHStrongSecretive property has to hold for a DH key k for some pair of honest

threads. A nonce is established to be a level-1 key when the protocol is proved to be a *secretive protocol* with respect to the nonce and a set of level-0 keys. This concept is extended further to define level-2 keys and so on.

For a set of keys $\mathcal{K}$ of levels ≤ 1, $\mathcal{C}(\mathcal{K})$ is the union of all the level-0 keys in $\mathcal{K}$ and the union of all the level-0 keys protecting the level-1 keys in $\mathcal{K}$. The formula $\mathsf{InInitSet}(X, s, \mathcal{K})$ asserts X is either the generator of nonce s or a possessor of some key in $\mathcal{C}(\mathcal{K})$. $\mathsf{GoodInit}(s, \mathcal{K})$ asserts that all such threads belong to honest principals. The formula $\mathsf{GoodKeyFor}$ lets us state that secrets established by secretive protocols, where possibly the secrets are also used as keys, are good keys against everybody except the set of principals who either generated the secret or are in possession of a key protecting the secret. For level-0 keys which we want to claim as being possessed only by honest principals we use the formula $\mathsf{GoodKey}$. For protocols employing an IND-CCA secure encryption scheme, the following axiom is sound:

$$\textbf{GK} \quad \mathsf{Secretive}(s, \mathcal{K}) \wedge \mathsf{GoodInit}(s, \mathcal{K}) \Rightarrow \mathsf{GoodKeyFor}(s, \mathcal{K})$$

If the encryption scheme is both IND-CCA and INT-CTXT secure then following axioms are sound:

$$\begin{aligned} \textbf{CTX0} \quad & \mathsf{GoodKey}(k) \wedge \mathsf{SymDec}(Z, E_{sym}[k](m), k) \supset \\ & \exists X.\, \mathsf{SymEnc}(X, m, k), \text{ for level-0 key } k. \\ \textbf{CTXL} \quad & \mathsf{Secretive}(s, \mathcal{K}) \wedge \mathsf{GoodInit}(s, \mathcal{K}) \wedge \mathsf{SymDec}(Z, E_{sym}[s](m), s) \supset \\ & \exists X.\, \mathsf{SymEnc}(X, m, s) \end{aligned}$$

The following axiom states that if a protocol is secretive with respect to s and $\mathcal{K}$, then the only keys, under which a message containing s openly is found encrypted in a "good" message, are in the set $\mathcal{K}$:

$$\begin{aligned} \textbf{SDEC} \quad & \mathsf{Secretive}(s, \mathcal{K}) \wedge \mathsf{SymDec}(X, E_{sym}[k](m), k) \wedge \\ & \mathsf{Good}(X, E_{sym}[k](m), s, \mathcal{K}) \wedge \mathsf{ContainsOpen}(m, s) \supset k \in \mathcal{K} \end{aligned}$$

The predicate $\mathsf{ContainsOpen}(m, a)$ asserts that a can be obtained from m by a series of unpairings only.

The **soundness theorem** is proved by showing that every axiom is a valid formula and that all proof rules preserve validity. The soundness proofs for the four axioms above are sketched in [42]; they proceed by reduction to the multiple encryption scheme game defined in the next section.

Theorem 1 (Soundness). $\forall \mathcal{Q}, \varphi.$ *if* $\mathcal{Q} \vdash \varphi$ *then* $\mathcal{Q} \vDash \varphi$

3.3 Joint Security of Multiple Encryption Schemes

A public-key encryption scheme $\mathcal{ES}$ is a triplet $(\mathcal{KG}, \mathcal{E}, \mathcal{D})$ such that $\mathcal{KG}(I)$ generates a pair of keys (ek, dk), where I is some initial information, ek is the public key and dk is the private key, and $\mathcal{E}$ and $\mathcal{D}$ are the encryption and decryption functions respectively. In [10], Bellare, Boldyreva and Micali analyzed the security of a single public-key encryption scheme in a setting where more than one independent keys are used. The security of an encryption scheme is defined in

terms of a game between an adversary and a challenger. In the *chosen plaintext* (IND-CPA) setting, the adversary has access to a *left-or-right* encryption oracle $\mathcal{E}_{ek}(LR(\cdot,\cdot,b))$, which takes a pair of equal length messages m_0, m_1 from the adversary and returns the encryption of m_b with the key ek, the bit b being unknown to the adversary. In the *chosen ciphertext* (IND-CCA) setting, the adversary has, in addition, access to a decryption oracle $\mathcal{D}_{dk}(\cdot)$, with the caveat that it cannot query for the decryption of a ciphertext it received as an answer to a previous encryption oracle query.

In this section, we extend their definition to settings involving multiple encryption schemes. Consider a sequence of n, not necessarily distinct, encryption schemes $\langle \mathcal{ES}^i \mid 1 \leq i \leq n \rangle$, possibly consisting of public-key and symmetric-key encryption schemes with either pre-shared keys or setup by a Diffie-Hellman exchange. For notational uniformity we define $ek_i = dk_i$ for symmetric key schemes, both equal to the secret key. For Diffie-Hellman schemes, $ek_i = dk_i = keygen(g^{xy})$ where g^x and g^y are the public DH values. Let DH be the set of Diffie-Hellman public values (g^x, g^y) for those keys which are generated by a DH exchange and PK be the set of public-keys among the ek_i's. In the *multi-scheme* setting we let the adversary have access to n encryption and decryption oracles with their corresponding public informations (PK and DH), all using the *same* challenger bit b for encryption. Security in this setting is defined below.

Definition 2 (Multi Scheme Indistinguishability). *The experiment MS-IND-CCA, for adversary A, is defined as:*

$$\begin{array}{l}\textbf{Experiment } \mathbf{Exp}^{\textit{MS-IND-CCA}}_{\langle \mathcal{ES} \rangle, I}(A, b) \\ \quad \texttt{For } i = 1, \cdots, n \texttt{ do } (ek_i, dk_i) \leftarrow \mathcal{KG}^i(I) \texttt{ EndFor} \\ \quad d \leftarrow A^{\mathcal{E}^1_{ek_1}(LR(\cdot,\cdot,b)),\ldots,\mathcal{E}^n_{ek_n}(LR(\cdot,\cdot,b)),\mathcal{D}^1_{dk_1}(\cdot),\ldots,\mathcal{D}^n_{dk_n}(\cdot)}(I, PK, DH) \\ \quad \texttt{Return } d\end{array}$$

A query to any LR oracle consists of two messages of equal *length and that for each $i = 1, \ldots, n$ adversary A does not query $\mathcal{D}_{dk_i}(\cdot)$ on an output of $\mathcal{E}^i_{ek_i}(LR(\cdot,\cdot,b))$. The* advantage *of A is defined as:*

$$\mathbf{Adv}^{\textit{MS-IND-CCA}}_{\langle \mathcal{ES} \rangle, I}(A) = \Pr[\mathbf{Exp}^{\textit{MS-IND-CCA}}_{\langle \mathcal{ES} \rangle, I}(A, 0) = 0] - \Pr[\mathbf{Exp}^{\textit{MS-IND-CCA}}_{\langle \mathcal{ES} \rangle, I}(A, 1) = 0]$$

The sequence of encryption schemes $\langle \mathcal{ES}^i \mid 1 \leq i \leq n \rangle$ is MS-IND-CCA *secure if the advantage of any probabilistic poly-time adversary A is negligible in the security parameter.*

The definition of MS-IND-CPA is similar, with the decryption oracles dropped. We prove that individual security of the encryption schemes implies joint security.

Theorem 2 (IND-CPA(CCA) $\rightarrow$ MS-IND-CPA(CCA)). *If encryption schemes $\mathcal{ES}_1$, $\mathcal{ES}_2$, ..., $\mathcal{ES}_n$ are individually IND-CPA(CCA)secure, then the sequence of schemes $\langle \mathcal{ES}_1, \mathcal{ES}_2, \ldots, \mathcal{ES}_n \rangle$ is MS-IND-CPA(CCA) secure.*

4 Kerberos with DHINIT

In this section, we formally model Kerberos with DHINIT and prove that it satisfies computational authentication and secrecy properties under standard assumptions about the cryptographic primitives. Authentication proofs for each stage of Kerberos rely on the secrecy guarantees of keys set up in earlier stages, while the secrecy proofs similarly rely on previously proved authentication guarantees, an alternation first pointed out in [20]. Since the later stages of DHINIT are the same as those of Basic Kerberos [35], we obtain proofs for the complete protocol by appealing to security proofs and composition theorems in a compatible setting [39].

We find, perhaps surprisingly, that the KAS is *not* authenticated to the client after the first stage and suggest a fix to the protocol to avoid this problem. Our counterexample is similar in flavor to the attack found on Kerberos V5 with public-key initialization by [19]. In addition, we use an axiom that relies on *random oracles* to complete the proof of the security properties. We also develop an alternative proof, using only axioms that hold in the standard model, for a variant of the protocol that requires the KAS to sign both the Diffie-Hellman exponentials. We leave open whether this discrepancy arises from a security flaw in DHINIT or a limitation of our current proof.

4.1 Modelling the Protocol

The Kerberos protocol involves four roles—the Client, the Kerberos Authentication Server (KAS), the Ticket Granting Server (TGS), and the application server. The KAS and the TGS share a long term symmetric key as do the TGS and the application server. Mutual authentication and key establishment between the client and the application server is achieved by using this chain of trust. We write $k^{type}_{X,Y}$ to refer to long term symmetric keys, where X and Y are the principals sharing the key and *type* indicates their roles, e.g. $t \rightarrow k$ for TGS and KAS and $s \rightarrow t$ for application server and TGS. Kerberos runs in three stages with the client role participating in all three. The client program for the first stage and the KAS program are given below but the complete formal description of the protocol is given in [42].

Client $= (C, \hat{K}, \hat{T}, \hat{S}, t)$ [

new n_1; new $\tilde{n_1}$;

new x; $gx :=$ expg x;

$chksum :=$ hash $\hat{C}.\hat{T}.n_1$;

$sigc :=$ sign "$Auth$"$.chksum.\tilde{n_1}.gx, sk_C$;

send $Cert_C.sigc.\hat{C}.\hat{T}.n_1$;

receive $Cert_K.sigk.\hat{C}.tgt.enc_{kc}$;

verify $sigk$, "$DHKey$"$.gy.\tilde{n_1}, vk_K$;

$k :=$ dhkeygen gy, x;

$text_{kc} :=$ symdec enc_{kc}, k;

match $text_{kc}$ as $AKey.n_1.\hat{T}$;

$\cdots stage\ boundary \cdots$

$]_C$

KAS $= (K)$ [

receive $Cert_C.sigc.\hat{C}.\hat{T}.n_1$;

verify $sigc$, "$Auth$"$.chksum.\tilde{n_1}.gx, vk_C$;

$chk :=$ hash $\hat{C}.\hat{T}.n_1$;

match chk as $chksum$;

new $AKey$;

new y; $gy :=$ expg y;

$k :=$ dhkeygen gx, y;

$sigk :=$ sign "$DHKey$"$.gy.\tilde{n_1}, sk_K$;

$tgt :=$ symenc $AKey.\hat{C}, k^{t \to k}_{T,K}$;

$enc_{kc} :=$ symenc $AKey.n_1.\hat{T}, k$;

send $Cert_K.sigk.\hat{C}.tgt.enc_{kc}$;

$]_K$

The client C and KAS K carry out a Diffie-Hellman key exchange protocol authenticated by digital signatures to set up a key $AKey$ to be used as a session key between the client and the TGS in the next stage. (In Basic Kerberos, this phase is simpler; it relies on a preshared key between C and K.) The first few actions of the client are explained as follows: it generates three random numbers $n_1, \tilde{n_1}, x$ using `new` actions. It then generates the Diffie-Hellman exponential gx and sends a message to the KAS K containing its signature over the exponential and a few other fields including the identities of the TGS $\hat{T}$ and itself. In the second stage, the client gets a new session key ($SKey$ - Service Key) and a service ticket (st) to converse with the application server S which takes place in the third stage. The control flow of Kerberos exhibits a staged architecture where once one stage has been completed successfully, the subsequent stages can be performed multiple times or aborted and started over for handling errors.

4.2 Security Properties and Proofs

Table 1 lists the authentication and secrecy properties of Kerberos with DHINIT that we want to prove. The authentication properties are of the form that a message of a certain format was indeed sent by some thread of the expected principal. The secrecy properties state that a putative secret is a good key for certain principals. For example, $AUTH^{client}_{kas}$ states that when C finishes executing the **Client** role, some thread of $\hat{K}$ indeed sent the expected message with probability asymptotically close to one; SEC^{client}_{akey} states that the authorization key is "good" after execution of the **Client** role by C. The other security properties are analogous. More specifically, $\mathsf{GoodKeyAgainst}(\mathsf{X}, \mathsf{k})$ [27] intuitively means that if k were used instead of a random key to key an IND-CCA encryption scheme, then the advantage of X in the corresponding security game would be negligible. The motivation for using this definition is that stronger conditions such as key indistinguishability fail to hold as soon as the key is used; key indistinguishability is also not necessary to establish reasonable security properties of practical protocols (see [27] for further discussion). We abbreviate the honesty assumptions by defining $\mathsf{Hon}(\hat{X_1}, \hat{X_2}, \cdots, \hat{X_n}) \equiv \mathsf{Honest}(\hat{X_1}) \wedge \mathsf{Honest}(\hat{X_2}) \wedge \cdots \mathsf{Honest}(\hat{X_n})$.

$$
\begin{aligned}
SEC_k &: \mathsf{Hon}(\hat{C},\hat{K}) \supset (\mathsf{GoodKeyAgainst}(X,k) \vee \hat{X} \in \{\hat{C},\hat{K}\})\\
SEC_{akey} &: \mathsf{Hon}(\hat{C},\hat{K},\hat{T}) \supset (\mathsf{GoodKeyAgainst}(X,AKey) \vee \hat{X} \in \{\hat{C},\hat{K},\hat{T}\})\\
SEC_{skey} &: \mathsf{Hon}(\hat{C},\hat{K},\hat{T},\hat{S}) \supset (\mathsf{GoodKeyAgainst}(X,SKey) \vee \hat{X} \in \{\hat{C},\hat{K},\hat{T},\hat{S}\})\\
AUTH_{kas} &: \exists \eta.\ \mathsf{Send}((\hat{K},\eta), Cert_K.SIG[sk_K](\text{“}DHKey\text{”}.gy.\tilde{n_1}).E_{sym}[k^{t\to k}_{T,K}](AKey.\hat{C}).\\
&\qquad E_{sym}[k](AKey.n_1.\hat{T}))\\
AUTH_{tgs} &: \exists \eta.\ \mathsf{Send}((\hat{T},\eta), \hat{C}.E_{sym}[k^{s\to t}_{S,T}](SKey.\hat{C}).E_{sym}[AKey](SKey.n_2.\hat{S}))
\end{aligned}
$$

$$
SEC^{client}_k : [\mathbf{Client}]_C\ SEC_k \qquad SEC^{kas}_k : [\mathbf{KAS}]_K\ SEC_k
$$

$$
\begin{aligned}
SEC^{client}_{akey} &: [\mathbf{Client}]_C\ SEC_{akey} & AUTH^{client}_{kas} &: [\mathbf{Client}]_C\ \mathsf{Hon}(\hat{C},\hat{K}) \supset AUTH_{kas}\\
SEC^{kas}_{akey} &: [\mathbf{KAS}]_K\ SEC_{akey} & AUTH^{tgs}_{kas} &: [\mathbf{TGS}]_T\ \mathsf{Hon}(\hat{T},\hat{K})\\
SEC^{tgs}_{akey} &: [\mathbf{TGS}]_T\ SEC_{akey} & &\qquad \supset \exists n_1,k,gy,\tilde{n_1}.\ AUTH_{kas}\\
& & AUTH^{client}_{tgs} &: [\mathbf{Client}]_C\ \mathsf{Hon}(\hat{C},\hat{K},\hat{T}) \supset AUTH_{tgs}\\
SEC^{client}_{skey} &: [\mathbf{Client}]_C\ SEC_{skey} & AUTH^{server}_{tgs} &: [\mathbf{Server}]_S\ \mathsf{Hon}(\hat{S},\hat{T})\\
SEC^{tgs}_{skey} &: [\mathbf{TGS}]_T\ SEC_{skey} & &\qquad \supset \exists n_2,AKey.\ AUTH_{tgs}
\end{aligned}
$$

Table 1. DHINIT Security Properties

The following protocol execution demonstrates that $AUTH^{client}_{kas}$ does *not* hold after the first stage of the client role.

$$
\begin{aligned}
C \longrightarrow K(I) &: Cert_C.SIG[sk_C](\text{“}Auth\text{”}.HASH(\hat{C}.\hat{T}.n_1).\tilde{n_1}.gx).\hat{C}.\hat{T}.n_1\\
I \longrightarrow K &: Cert_I.SIG[sk_I](\text{“}Auth\text{”}.HASH(\hat{I}.\hat{T}.n_1).\tilde{n_1}.gx).\hat{I}.\hat{T}.n_1\\
K \longrightarrow I \longrightarrow C &: Cert_K.SIG[sk_K](\text{“}DHKey\text{”}.gy.\tilde{n_1}).\\
&\qquad E_{sym}[k^{t\to k}_{T,K}](AKey.\hat{I}).E_{sym}[k](AKey.n_1.\hat{T})
\end{aligned}
$$

C cannot parse the incorrect $tgt:\ E_{sym}[k^{t\to k}_{T,K}](AKey.\hat{I})$, as it does not have the key $k^{t\to k}_{T,K}$. Consequently, after interacting with the KAS the client is not guaranteed that the KAS thinks it interacted with the client. This problem can be easily fixed by requiring the KAS to include the client's identity inside the signature. However, the subsequent interaction with the TGS does ensure that the KAS indeed intended communication with the given client.

Theorem 3 (KAS Authentication). *On execution of the* **Client** *role by a principal, it is guaranteed with asymptotically overwhelming probability that the intended KAS indeed sent the expected response assuming that both the client and the KAS are honest, the signature scheme is CMA-secure, the encryption scheme is IND-CCA and INT-CTXT secure, and the Decisional Diffie-Hellman (DDH) assumption holds. A similar result also holds for a principal executing the* **TGS** *role. Formally,* $KERBEROS \vdash AUTH^{client}_{kas}, AUTH^{tgs}_{kas}$.

The axiomatic proof is in [42]. The key steps of the proof are the following: (a) the client C verifies the KAS K's signature on its Diffie-Hellman public

value (gy) and the client's nonce ($\tilde{n_1}$) and infers using the **SIG** axiom that the KAS did produce the signature; (b) a program invariant (proved using the honesty rule **HON**) is used to infer that the KAS observed the client's nonce and produced the DH exponential gy by exponentiating some nonce y; (c) the next few proof steps establish that the Diffie-Hellman key k can be used as an encryption key only by C and K by proving that $\mathsf{DHSecretive}(\mathsf{X}, \mathsf{Y}, \mathsf{k})$ holds and then using the axiom **CTXG**; note that this step requires the use of the random oracle model since the soundness of **CTXG** depends on that; (d) since the client decrypted the ciphertext $E_{sym}[k](AKey.n_1.\hat{T})$ and the client did not produce it itself, we therefore infer that it must have been produced by the KAS. At this point, we are assured that the KAS agrees on $\hat{T}, gx, n$ and $AKey$. However, it still does not agree on the identity of the client. It turns out, as we will see in Theorem 4, that this partial authentication is sufficient to prove the secrecy of the authentication key ($AKey$) from the client's perspective. Now, stronger authentication properties are proved from the second stage of the protocol once the client decrypts the message $E_{sym}[AKey]$ $(SKey.n_2.\hat{S})$. We infer that some thread of $\hat{C}, \hat{K}$ or $\hat{T}$ must have produced the encryption because of ciphertext integrity. Using an invariant to reason about the special form of this ciphertext, we conclude that the encrypting thread must have received a tgt containing $AKey$ and meant for itself. Since we have proved the secrecy of $AKey$ already under the keys k and $k_{T,K}^{t \to k}$, we infer that this tgt must be keyed with one of k and $k_{T,K}^{t \to k}$ the holders of which—$\hat{C}$, $\hat{T}$ and $\hat{K}$—are honest. This reasoning is formally captured in the axiom **SDEC**. Now we use the honesty rule to infer that if an honest thread encrypted this message then it must have generated $AKey$; we know that thread is K. At this point, we conclude that the TGS agrees on the identity of the KAS. The proof that the TGS agrees on the identity of the client is similar.

Theorem 4 (Authentication Key Secrecy). *On execution of the* **Client** *role by a principal, the Authentication Key is guaranteed to be good, in the sense of IND-CCA security, assuming that the client, the KAS and the TGS are all honest, the signature scheme is CMA-secure, the encryption scheme is IND-CCA and INT-CTXT secure, and the DDH assumption holds. Similar results hold for principals executing the* **KAS** *and* **TGS** *roles. Formally, KERBEROS* $\vdash SEC_{akey}^{client}, SEC_{akey}^{kas}, SEC_{akey}^{tgs}$.

The axiomatic proof is in [42]. The main idea is to prove by induction over the steps of the protocol that $AKey$ occurs on the network only as an encryption key or as a payload protected by encryption with the Diffie-Hellman key k or the pre-shared key $k_{T,K}^{t \to k}$. Formally, this step is carried out using the secrecy induction rule IND_{GOOD}. We therefore infer that $AKey$ is good for use as an encryption key using the axiom **GK**.

Since AKey is protected by both the DH key k and the symmetric key $k_{T,K}^{t \to k}$, therefore, we had to formulate a reduction to a multi party IND-CCA game where some of the keys can be symmetric, with either pre-shared keys or those generated by DHKE in section 3.3. Although not required for this paper, we

considered the further generalization of also considering public keys, since that didn't involve additional innovation.

We prove additional authentication and secrecy properties about the later stages of the protocol. Since the later stages of DHINIT are the same as those in basic Kerberos, we leverage the composition theorems in prior work to reuse existing proofs [39].

Theorem 5 (TGS Authentication). *On execution of the* **Client** *role by a principal, it is guaranteed with asymptotically overwhelming probability that the intended TGS indeed sent the expected response assuming that the client, the KAS and the TGS are all honest, the signature scheme is CMA-secure, the encryption scheme is IND-CCA and INT-CTXT secure, and the DDH assumption holds. Similar result holds for a principal executing the* **Server** *role. Formally,* $KERBEROS \vdash AUTH_{tgs}^{client}, AUTH_{tgs}^{server}$.

Theorem 6 (Service Key Secrecy). *On execution of the* **Client** *role by a principal, the Service Key is guaranteed to be good, in the sense of IND-CCA security, assuming that the client, the KAS, the TGS and the application server are all honest, the signature scheme is CMA-secure, the encryption scheme is IND-CCA and INT-CTXT secure, and the DDH assumption holds. Similar result holds for a principal executing the* **TGS** *role. Formally,* $KERBEROS \vdash SEC_{skey}^{client}, SEC_{skey}^{tgs}$.

5 IKEv2

IKEv2 [34] is a complex protocol used to negotiate a security association at the beginning of an IPSec session. We consider the mode in which Diffie-Hellman exponentials are never reused and signatures are used for authentication. We formally model this mode of IKEv2 and provide the first formal proof that it satisfies computational authentication and security guarantees in the standard model; full details are in [42]. A significant difference from DHINIT is that the IKEv2 proofs do not require the random oracle model. At a high-level, this difference arises because in IKEv2 honest parties authenticate their own as well as their peer's Diffie-Hellman exponential using signatures. This enables us to prove the $\mathsf{DHStrongSecretive}(\mathsf{X}, \mathsf{Y}, \mathsf{k})$ property and use the **CTXGS** axiom in our proofs. Recall that in the DHINIT proofs we could only prove the weaker $\mathsf{DHSecretive}(\mathsf{X}, \mathsf{Y}, \mathsf{k})$ property and hence had to use the **CTXG** axiom, which is sound only in the random oracle model. However, the key derivation function needs to satisfy certain properties (described in [42] based on issues identified in [21]).

The security properties of IKEv2, listed in Table 2, state that on completion of a thread executing one of the roles, the shared keys sk_i and sk_r satisfy the $\mathsf{GoodKey}$ property, i.e. they are suitable for use as encryption keys for an IND-CCA scheme. The authentication properties state that on completion of a thread executing either role, it is guaranteed with overwhelming probability that the intended peer indeed received and sent the corresponding messages.

$$SEC_{sk}^{init} : [\mathbf{Init}]_A\ \mathsf{Hon}(\hat{A}, \hat{B}) \supset (\mathsf{GoodKeyAgainst}(X, sk_i) \vee \hat{X} \in \{\hat{A}, \hat{B}\}) \wedge$$
$$(\mathsf{GoodKeyAgainst}(X, sk_r) \vee \hat{X} \in \{\hat{A}, \hat{B}\})$$
$$SEC_{sk}^{resp} : [\mathbf{Resp}]_B\ \mathsf{Hon}(\hat{A}, \hat{B}) \supset (\mathsf{GoodKeyAgainst}(X, sk_i) \vee \hat{X} \in \{\hat{A}, \hat{B}\}) \wedge$$
$$(\mathsf{GoodKeyAgainst}(X, sk_r) \vee \hat{X} \in \{\hat{A}, \hat{B}\})$$
$$AUTH_{resp}^{init} : [\mathbf{Init}]_A\ \exists \eta.\ B = (\hat{B}, \eta) \wedge \mathsf{Receive}(B, \text{“}I\text{”}.info_{i1}.gx.n) <$$
$$\mathsf{Send}(B, \text{“}R\text{”}.info_{i2}.gy.m) < \mathsf{Receive}(B, enc_i) < \mathsf{Send}(B, enc_r)$$
$$AUTH_{init}^{resp} : [\mathbf{Resp}]_B\ \exists \eta.\ A = (\hat{A}, \eta) \wedge \mathsf{Send}(A, \text{“}I\text{”}.info_{i1}.gx.n) <$$
$$\mathsf{Receive}(A, \text{“}R\text{”}.info_{i2}.gy.m) < \mathsf{Send}(A, enc_i)$$

Table 2. IKEv2 Security Properties

Theorem 7 (IKEv2 Key Secrecy). *On execution of the* **Init** *role by a principal, the keys sk_i, sk_r are guaranteed to be good, in the sense of IND-CCA security, assuming that the Iniatiator and the Responder are both honest, the signature scheme is CMA-secure, the encryption scheme is IND-CCA and INT-CTXT secure, and the DDH assumption holds. Similar result holds for a principal executing the* **Resp** *role. Formally, IKEv2 $\vdash SEC_{sk}^{init}, SEC_{sk}^{resp}$.*

Theorem 8 (IKEv2 Authentication). *On execution of the* **Init** *role by a principal, it is guaranteed with asymptotically overwhelming probability that the intended Responder indeed received the intended messages and sent the expected responses assuming that both the Initiator and the Responder are honest, the signature scheme is CMA-secure, the encryption scheme is IND-CCA and INT-CTXT secure, and the DDH assumption holds. A similar result also holds for a principal executing the* **Resp** *role. Formally, IKEv2 $\vdash AUTH_{resp}^{init}, AUTH_{init}^{resp}$.*

6 Conclusion

We develop axioms and rules for proving authentication and secrecy properties of protocols that use Diffie-Hellman key exchange in combination with other mechanisms. The resulting reasoning method, which reflects intuitive informal direct arguments, is proved computationally sound by showing the existence of conventional cryptographic reductions.

We prove security of Kerberos with DHINIT, as defined in the RFC [43], in the *random oracle model,* and prove security in the standard model for a modification in which the KAS signs both the Diffie-Hellman exponentials. We also discover that the KAS is *not* authenticated to the client after the first stage and suggest a fix to the protocol to avoid this problem. While IKEv2 [34] provides for several cryptographic options, we focus on the mode in which Diffie-Hellman exponentials are never reused and signatures are used for authentication. We prove that IKEv2 satisfies computational authentication and secrecy guarantees in the standard model. Intuitively, we do not need the random oracle assumption

because honest IKEv2 parties authenticate both their own and their peer's Diffie-Hellman exponentials, which we believe is a prudent engineering practice.
Acknowledgement: We thank the referees and Iliano Cervesato for helpful comments and suggestions.

References

1. M. Abadi and P. Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.
2. P. Adão, G. Bana, and A. Scedrov. Computational and information-theoretic soundness and completeness of formal encryption. In *Proc. of the 18th IEEE Computer Security Foudnations Workshop*, pages 170–184, 2005.
3. W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold. Just Fast Keying: Key agreement in a hostile internet. *ACM Trans. Inf. Syst. Security*, 7(4):1–30, 2004.
4. M. Backes, I. Cervesato, A. D. Jaggard, A. Scedrov, and J.-K. Tsay. Cryptographically sound security proofs for basic and public-key Kerberos. In *Proceedings of 11th European Symposium on Research in Computer Security*, 2006. To appear.
5. M. Backes and B. Pfitzmann. Limits of the cryptographic realization of XOR. In *Proc. of the 10th European Symposium on Research in Computer Security*. Springer-Verlag, 2005.
6. M. Backes and B. Pfitzmann. Relating symbolic and cryptographic secrecy. In *Proc. IEEE Symposium on Security and Privacy*, pages 171–182. IEEE, 2005.
7. M. Backes, B. Pfitzmann, and M. Waidner. A universally composable cryptographic library. Cryptology ePrint Archive, Report 2003/015, 2003.
8. M. Backes, B. Pfitzmann, and M. Waidner. Limits of the reactive simulatability/UC of Dolev-Yao models with hashes. In *Proc. of the 11th European Symposium on Research in Computer Security*. Springer-Verlag, 2006.
9. G. Bella and L. C. Paulson. Kerberos version IV: Inductive analysis of the secrecy goals. In *Proceedings of the 5th European Symposium on Research in Computer Security*, pages 361–375, 1998.
10. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology - EUROCRYPT 2000, Proceedings*, pages 259–274, 2000.
11. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT*, pages 531–545, 2000.
12. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
13. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '93)*, pages 232–249. Springer-Verlag, 1994.
14. A. Boldyreva and V. Kumar. Provable-security analysis of authenticated encryption in Kerberos. In *Proc. IEEE Security and Privacy*, 2007.
15. E. Bresson, Y. Lakhnech, L. Mazare, and B. Warinschi. A generalization of DDH with applications to protocol analysis and computational soundness. In *Advances in Cryptology - CRYPTO 2007, Proceedings*, 2007.

16. F. Butler, I. Cervesato, A. D. Jaggard, and A. Scedrov. Verifying confidentiality and authentication in Kerberos 5. In *ISSS*, pages 1–24, 2003.
17. R. Canetti and M. Fischlin. Universally composable commitments. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Proceedings*, pages 19–40, 2001.
18. R. Canetti and J. Herzog. Universally composable symbolic analysis of mutual authentication and key-exchange protocols. In *Theory of Cryptography Conference - Proceedings of TCC 2006*, pages 380–403, 2006.
19. I. Cervesato, A. Jaggard, A. Scedrov, J.-K. Tsay, and C. Walstad. Breaking and fixing public-key Kerberos. In *Eleventh Annual Asian Computing Science Conference - ASIAN*, pages 164–178, 2006.
20. I. Cervesato, C. Meadows, and D. Pavlovic. An encapsulated authentication logic for reasoning about key distribution protocols. In *CSFW*, pages 48–61, 2005.
21. O. Chevassut, P.-A. Fouque, P. Gaudry, and D. Pointcheval. Key derivation and randomness extraction. Cryptology ePrint Archive, Report 2005/061, 2005. `http://eprint.iacr.org/`.
22. V. Cortier and B. Warinschi. Computationally sound, automated proofs for security protocols. In *Proceedings of 14th European Symposium on Programming (ESOP'05)*, pages 157–171, 2005.
23. A. Datta, A. Derek, J. Mitchell, A. Ramanathan, and A. Scedrov. Games and the impossibility of realizable ideal functionality. In *TCC*, pages 360–379, 2006.
24. A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system and compositional logic for security protocols. *Journal of Computer Security*, 13:423–482, 2005.
25. A. Datta, A. Derek, J. C. Mitchell, and A. Roy. Protocol Composition Logic (PCL). *Electronic Notes in Theoretical Computer Science*, 172:311–358, 2007.
26. A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov, and M. Turuani. Probabilistic polynomial-time semantics for a protocol security logic. In *ICALP*, pages 16–29, 2005.
27. A. Datta, A. Derek, J. C. Mitchell, and B. Warinschi. Computationally sound compositional logic for key exchange protocols. In *Proceedings of 19th IEEE Computer Security Foundations Workshop*, pages 321–334. IEEE, 2006.
28. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
29. C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell. A modular correctness proof of IEEE 802.11i and TLS. In *ACM Conference on Computer and Communications Security*, pages 2–15, 2005.
30. J. Herzog. The Diffie-Hellman key-agreement scheme in the strand-space model. In *Proceedings of 16th IEEE Computer Security Foundations Workshop*, pages 234–247. IEEE, 2003.
31. J. Herzog. *Computational Soundness for Standard Assumptions of Formal Cryptography*. PhD thesis, MIT, 2004.
32. R. Janvier, L. Mazare, and Y. Lakhnech. Completing the picture: Soundness of formal encryption in the presence of active adversaries. In *Proceedings of 14th European Symposium on Programming (ESOP'05)*, pages 172–185, 2005.
33. J. Katz and M. Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In *FSE*, pages 284–299, 2000.
34. C. Kaufman. Internet Key Exchange (IKEv2) Protocol, 2005. RFC.
35. J. Kohl and B. Neuman. The kerberos network authentication service, 1991. RFC.

36. Y. Lakhnech and L. Mazaré. Computationally sound verifiation of security protocols using Diffie-Hellman exponentiation. Cryptology ePrint Archive: Report 2005/097, 2005.
37. D. Micciancio and B. Warinschi. Completeness theorems for the Abadi-Rogaway logic of encrypted expressions. *Journal of Computer Security*, 12(1):99–129, 2004. Preliminary version in WITS 2002.
38. D. Micciancio and B. Warinschi. Soundness of formal encryption in the presence of active adversaries. In *Theory of Cryptography Conference - Proceedings of TCC 2004*, pages 133–151, 2004.
39. A. Roy, A. Datta, A. Derek, and J. C. Mitchell. Inductive proofs of computational secrecy. In *ESORICS*, pages 219–234, 2007. Full version at `http://www.stanford.edu/~arnab/rddm-InductiveProofs.pdf`.
40. A. Roy, A. Datta, A. Derek, and J. C. Mitchell. Inductive trace properties for computational security. *WITS*, 2007. Full version at `http://www.stanford.edu/~arnab/rddm-IndTraceProps.pdf`.
41. A. Roy, A. Datta, A. Derek, J. C. Mitchell, and J.-P. Seifert. Secrecy analysis in Protocol Composition Logic., 2006. to appear in Proceedings of 11th Annual Asian Computing Science Conference.
42. A. Roy, A. Datta, and J. C. Mitchell. Formal proofs of cryptographic security of Diffie-Hellman-based protocols. *Manuscript*, 2007. `http://www.stanford.edu/~arnab/rdm-DHProofs.pdf`.
43. L. Zhu and B. Tung. Public Key Cryptography for Initial Authentication in Kerberos (PKINIT). RFC 4556 (Proposed Standard), June 2006.