2014 IEEE 27th Computer Security Foundations Symposium

Decidability for Lightweight Diffie-Hellman Protocols

Daniel J. Dougherty Joshua D. Guttman

Worcester Polytechnic Institute
{dd, guttman}@wpi.edu

***Abstract*—Many protocols use Diffie-Hellman key agreement, combined with certified long-term values or digital signatures for authentication. These protocols aim at security goals such as key secrecy, forward secrecy, resistance to key compromise attacks, and various flavors of authentication. However, these protocols are challenging to analyze, both in computational and symbolic models. An obstacle in the symbolic model is the undecidability of unification in many theories in the signature of rings.**

In this paper, we develop an algebraic version of the symbolic approach, working directly within finite fields, the natural structures for the protocols. The adversary, in giving an attack on a protocol goal in a finite field, may rely on any identity in that field. He defeats the protocol if there are attacks in infinitely many finite fields. We prove that, even for this strong adversary, security goals for a wide class of protocols are decidable.

I. INTRODUCTION

Diffie-Hellman (DH) key agreement [15] is widely used. It produces a shared secret, and is often combined with additional techniques intended to authenticate the participants to each other. Despite vigorous research in symbolic analysis of security protocols, many questions remain about these fundamental techniques. While systems such as NPA-Maude [18], ProVerif [5], AVISPA [2], [3], CPSA [41], and Scyther [14] are extremely useful, indirect or ad hoc techniques are still needed to analyze protocols using DH, as in [29]. Computational techniques, also, for these protocols, have led to arduous proofs after which controversy remains [24], [26], [28], [35].

Much of the challenge for symbolic analysis derives from the fact that DH works in a finite cyclic group $\mathbb{C}_q$, using exponentiation to combine secret values. The exponents permit both addition and multiplication, forming a field $\mathbb{F}_q$. Equational theories extending the theory of rings tend to have undecidable unification problems [33]. Symbolic approaches typically simplify this signature, retaining the multiplicative but not the additive operator. Some theoretical work supports the restricted, multiplication-only framework [8], [25], [32].

However, important protocols, e.g. the implicitly authenticated DH protocols MQV and HMQV [4], [24], [26], [31], are out of reach of these techniques, even recent work such as Tamarin [43]. These protocols use the full field structure of the exponents, and call for a strong adversary model to match.

In this paper,[1] we make the following main contributions. *First*, we identify a class of DH protocols Π and a set of security goals G such that, whenever Π has an execution which is a counterexample to G, then there is a counterexample using no more than a bounded number b of runs of roles of Π. The bound b is determined from Π and G. We call these the *lightweight* Diffie-Hellman protocols, and they are reasonably inclusive. The Station-to-Station protocol [16] is not lightweight, but several variants of it are. Lightweight protocols also include implicitly authenticated DH protocols such as the Unified Model [1], as well as MQV and HMQV.

Second, we introduce a strong adversary model, motivated by the algebraic mechanisms underlying DH protocols. In this model, the adversary acts against a protocol Π in particular finite fields $\mathbb{F}_q$. In staging an attack, he can rely on any valid identity of $\mathbb{F}_q$, without restricting the adversary's algebraic operations (or those in the protocol). The adversary defeats a security goal G of Π if, for infinitely many q, he has counterexamples to G when Π executes in $\mathbb{F}_q$ (see Section III-E).

Our security goals are geometric sequents, which are formulas $\forall \overline{x} \,.\, \Phi \supset \Psi$, where Φ, Ψ are positive-existential, and may contain $\wedge$, $\vee$, and $\exists$, but not $\forall$, $\supset$, or $\neg$. An execution is a structure (model) in which a goal may be satisfied; if it is falsified, then this structure is an attack on the goal. Each structure involves a particular finite field $\mathbb{F}$, so an attack for $\mathbb{F}$ may not carry over to a different field $\mathbb{F}'$. Structures over a free algebra, although not themselves real executions, summarize common patterns in potential attacks over different fields.

We show that it is decidable whether a lightweight DH protocol enforces a security goal.

Thus, we combine symbolic and algebraic methods for reasoning about DH protocols. Unlike previous work that regards the messages as defined by a rewriting theory, including our recent work [17], we use the traditional algebraic structures of groups and fields. Hence, mathematical methods such as Gaussian elimination can replace rewriting methods, especially unification, in key places. This paper has seven main steps:

1. We define the lightweight protocols (Definition II.1).
2. We define a first-order logical language $\mathcal{GL}(\Pi)$ for each protocol Π. It expresses security goals for DH protocols as geometric sequents (Section IV). These goals include key secrecy, which we formalize as a reachability property, not by indistinguishability. They also include authentication of several flavors, as well as forward secrecy and resistance to impersonation attacks (Section IV-A).
3. Many questions about rings and fields are undecidable. Thus, $\mathcal{GL}(\Pi)$ is constructed to express our security goals, but to be insensitive to other characteristics of the underlying algebraic structures (Lemmas IV.3 and VI.16).
4. We prove a *small model* property for security goals and

---

[1] Funded by the National Science Foundation under Grant CNS-1116557.

DOI 10.1109/CSF.2014.23

lightweight protocols, i.e. a size bound such that, if any protocol run can falsify the goal, then some run smaller than the bound falsifies it (Thm. V.4).

5. In a very spare symbolic model we call FAlg, there are only finitely many non-isomorphic structures smaller than the bound (Lemma. III.3). Each one represents a (possibly empty) set of potential executions.
6. For each of these FAlg structures, we use a constraint solving method [10], [36], followed by Gaussian elimination, to determine for which q it yields a non-empty set of executions over $\mathbb{F}_q$. This produces either a solution that works uniformly in q, or else a solution for at most finitely many choices of q. Thus, when the adversary can defeat the protocol for infinitely many choices of q, a fixed solution wins for all q. This leads to decidability for lightweight DH protocols (Section VI).
7. We illustrate our method, showing the main part of checking that MQV achieves key secrecy. We also derive Kaliski's unknown key share attack against MQV [24].

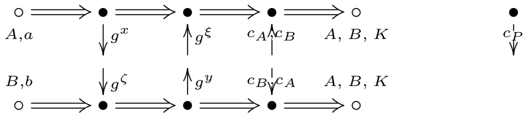
Step six is more challenging than the NP-completeness shown by Rusinowich and Turuani [42], since the adversary may rely on the identities true in each $\mathbb{F}_q$.

Thus we have obtained a stronger, cleaner analysis by working with a richer algebra—fields of rational functions—than rewriting methods allow.

Related Work. Two characteristics distinguish our work. First, we offer all the operations of fields, including addition and division. Addition occurs in many key computations. Division occurs in some protocols, but, more importantly, the adversary may use it, and some actual attacks depend on it (e.g. Kaliski's unknown key share attack on MQV [24]; cf. Section VII). Second, we do not assume the bounded session model, which asserts security goals only about small executions (see Chevalier et al. [8], and Millen and Shmatikov [37]). On the contrary, for lightweight protocols, we prove all attacks can be found among small executions. The lightweight conditions adapt an idea of Suresh and Ramanujam [40]. Our paper appears to be the first DH decidability result outside the bounded session model. Since several real protocols, often using the full field structure, are lightweight, this seems a major step forward.

We focus on defining the underlying semantics of messages, and their relations to groups and fields, accurately. Many papers have treated the operators and their properties purely syntactically, e.g. [11], [25]. Meadows and Pavlovic adopt an axiomatic approach, which leads to a flexible and suggestive method, but does not elucidate the message structures [38]. Many papers also use linear algebra as we do, such as Pereira and Quisquater's generic insecurity for AGDH [39] and Kremer and Mazaré on protocols using bilinear pairings [27].

Our use of extension fields in which the field extension elements are exponents chosen by the regular participants and then by the adversary, provides an algebraic framework which matches our strong symbolic adversary model. Bresson et al. [7] study a related adversary model, showing it sound relative to a passive computational adversary.



$$c_P = [\![\, \mathsf{cert}\ g^\pi,\ P\,]\!]_{\mathsf{sk(CA)}}$$

Initiator Parameters: $A, B, a, x, \xi, \beta, K$
Responder Parameters: $A, B, b, y, \zeta, \alpha, K$

Fig. 1. IADH Initiator, Responder, and CA Roles

II. SOME DH PROTOCOLS

Many Diffie-Hellman style protocols, including some implicitly authenticated DH protocols, have the form shown in Fig. 1. We use $t_0,\ t_1$ for the pair or concatenation of two messages t_0 and t_1, and we use $[\![\, t\,]\!]_K$ for a digital signature on t produced using signing key K.

Each party has a long-term secret exponent, which we write a for the participant named A and b for B, etc. Each party gets his or her secret certified. The signed message $[\![\, \mathsf{cert}\ g^a,\ A\,]\!]_{\mathsf{sk(CA)}}$ binds A's public long-term value g^a to A's name. We regard the CA as simply transmitting the relevant certificate (Fig. 1, top right). The CA may require procedures such as a proof of knowledge before issuing the certificate; their consequences may be expressed via axioms.

We represent each role of the protocol as a *strand*, namely a linear sequence of events (called *nodes*) each of which is a *transmission*, a *reception*, or a *neutral node*. Neutral nodes neither transmit nor receive, but only retrieve or store values into the principal's long term state [21]. We write $\circ$ for these local events, and $\bullet$ for transmissions and receptions.

Each party first retrieves its name and long term secret from storage without sending or receiving anything. They transmit the public exponentiated ephemeral values g^x, g^y matching their secrets x, y. They receive group elements g^ξ, g^ζ. These are the partners' ephemeral public values when the adversary is not active in a particular session (and the network cooperates). The roles differ only in the order of the transmission and reception events. A does not know that g^ξ is the same as the g^y that B sent, and conversely for B. All they know is that they have received group elements (which they test, ensuring $g^\xi, g^\zeta \neq g^0$). Although the participants do not explicitly know ξ and ζ, we regard them as parameters. We use Greek letters for parameters that the participants cannot identify.

Next, each party receives a certificate containing a name, possibly the actual name of his peer, and a value g^α, possibly equal to the actual public value g^a. The last event is a key computation. It deposits a record containing the new session key and the associated principal names into local storage. This key is never transmitted, though it may subsequently be used for encryption, or to derive keys.

Fig. 1 is compatible with many protocols, because the session key may be computed in many ways. Several of these are shown, as computed by the initiator, in Fig. 2. $\mathsf{Hash}(t)$ and $\mathsf{hash}(t)$ are hash functions that produce values in Mesg and E respectively. MQV uses an operator $[t]$, returning an exponent from any message. Since this "poor man's hash function"

UM	$\mathsf{Hash}((g^{\beta})^{a},\ (g^{\xi})^{x})$
Naxos	$\mathsf{Hash}((g^{\xi})^{a},\ (g^{\beta})^{x'},\ (g^{\xi})^{x'},\ A,\ B)$
CF	$(g^{\xi}g^{\beta})^{x+a}$
(H)MQV	$(g^{\xi}(g^{\beta})^{E})^{x+Da}$

MQV: $E=[g^{\xi}],\quad D=[g^{x}]$

HMQV: $E=\mathsf{hash}(g^{\xi},\ B),\quad D=\mathsf{hash}(g^{x},\ A)$

Naxos, etc: Use $x'=\mathsf{hash}(x,\ a)$ and $y'=\mathsf{hash}(y,\ b)$

Fig. 2. Some IADH key computations for initiator [1], [13], [26], [30], [31]

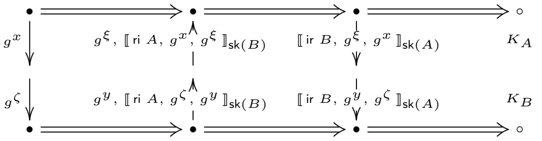

g^{x}
g^{ξ}, $[\![\mathsf{ri}\ A,\ g^{x},\ g^{\xi}]\!]_{\mathsf{sk}(B)}$
$[\![\mathsf{ir}\ B,\ g^{\xi},\ g^{x}]\!]_{\mathsf{sk}(A)}$
K_A
g^{ζ}
g^{y}, $[\![\mathsf{ri}\ A,\ g^{\zeta},\ g^{y}]\!]_{\mathsf{sk}(B)}$
$[\![\mathsf{ir}\ B,\ g^{y},\ g^{\zeta}]\!]_{\mathsf{sk}(A)}$
K_B

Fig. 3. Variant of the Station-to-Station protocol

satisfies Assumption III.7, we view it as a hash. Many other key computations have been proposed (see e.g. [4], [6]).

Naxos [30] suggests using the hashed values $\mathsf{hash}(x,a)$ and $\mathsf{hash}(y,b)$ in place of x and y respectively. This doubles the number of candidate protocols, and is an advantage in attack models where the long-term value a is protected by hardware, but the short term value x can be compromised more easily. We will not include the Naxos-style protocols as lightweight.

Protocols may also take other forms. Cremers-Feltz [13] place each ephemeral value g^x, g^y inside a digital signature. By contrast, the Station-to-Station protocol [16] signs the pair of both ephemeral values, and does not use certified long-term DH values. The session key is $K_A=(g^{\xi})^{x}$. The original STS uses encryptions to protect the signatures. The variant in Fig. 3 omits the encryptions. The tags ri, ir distinguish the directions of the signed units as responder-to-initiator or the reverse.

The encryptions and syntactically ambiguous signatures entail that STS itself is not a lightweight protocol. Another example of a protocol that we will not cover is MTI(C) [6], [34]. In this protocol, each participant chooses an ephemeral value x, y, but in fact transmits $(g^{\beta})^{x}$ or $(g^{\alpha})^{y}$. The message flow differs from Fig. 1, because the certificates must be received before these group elements are transmitted. The key in a successful session, in which the transmitted values are received unchanged, and in which $\alpha=a$ and $\beta=b$, is g^{xy}. A computes this from $g^{\alpha y}$ by exponentiating to the power x/a; B's computation is symmetric. In MTI(C), a group element is transmitted that depends partly on a value that was received, and partly on an exponent chosen by the principal. This type of protocol, like Naxos, calls for separate treatment in the future.

Definition II.1. The *lightweight Diffie-Hellman protocols* are those that satisfy the following four assumptions:

Well-typed messages are sent and received by the regular participants (Defn. III.6.1). In particular, every message sent or received is either *a group element g^v or other primitive value* such as a nonce or principal name; or *built recursively* by pairing and digital signature.

Exponents x assumed uncompromised are chosen on a role only when the group element g^x is transmitted. This concerns $x \in \mathsf{rl\_non}(\rho)$ (see Defn. III.6.2); it applies if the chosen exponent x is assumed uncompromised, *for every strand instantiating role ρ.*

Simple digital signatures are used. Specifically, there is a globally compatible order on the sequence in which different digitally signed messages may be used. This order has only finitely many different classes. See Def. V.2.

Linear use of received exponents α in a key computation. If group elements $g^{\alpha_0}, g^{\alpha_1}$ etc. are received in a role, then any monomial in the resulting key has, for these variables α_i, at most total degree 1. See Def. III.14.

This last assumption is really not a restriction on the protocols considered. For the compliant participants to compute a monomial of total degree >1 in the α_i, they would need to circumvent the Computational Diffie-Hellman assumption.

The assumption on digital signatures is crucial for the boundedness result, as helped by the uncompromised exponent principle. The first assumption entails that lightweight protocols do not send and receive encrypted messages; the purpose of these protocols being to agree on session keys, they therefore do not assume shared keys. The assumption that exponents appear only simply in the form g^x, in combination with the last assumption, excludes MTI(C) and Naxos.

III. MESSAGES, PROTOCOLS, AND THE ADVERSARY

We work with message algebras of two kinds. To model protocol executions we work with algebras including cyclic groups and fields as the interpretations of some of their sorts. These field-based algebras are not syntactic: they are not freely generated by taking a quotient under theory consisting of equations or conditional equations. This is unavoidable: fields cannot be axiomatized by conditional equations (as is shown by the fact that the product of fields is not a field). This essentially distinguishes protocol analysis in this new setting.

We also use syntactic algebras, generated from given values by free operators. They allow us to represent patterns that are independent of the choice of field, especially patterns for counterexamples to security goals. In fact we use two different syntactic algebras. One has a signature that includes the group and field operators; the other has a more restrictive signature. The latter allows only finitely many non-isomorphic structures with a bounded number of sessions (Section I, claim 5).

A. Strands and Protocols

We define strands, protocols, etc., uniformly for different algebras. Our algebras are order-sorted, with top sort D. We also distinguish two subsets Param and Basic, where $\mathsf{Param} \subseteq \mathsf{Basic} \subseteq D$. We assume that each algebra is equipped with a relation of *occurrence* and an *ingredient* relation $\sqsubseteq$; these features are defined when we instantiate them in Sections III-B, III-C. The term algebra $\mathcal{T}_{\Delta}(\mathsf{Param})$ over a signature Δ consists of terms freely generated from Param.

Homomorphisms from one algebra to another, or to itself (endomorphisms), are defined as usual; we usually write σ for a homomorphism, since they represent substitutions (among other purposes).

We use these algebras to build models of protocol behavior. We follow previous strand space practice (e.g. [22]) except that we allow the roles of a protocol to have *constraints*. A role constraint $\mathsf{rconstr}(\rho, i)$ gives an equation that must be true when the i^{th} step of the role ρ is instantiated. We use them to represent the key computations, relating a key parameter K to the other parameters, via a condition such as $K = (g^{\xi})^{x}$; the right hand side also can take any of the forms in Fig. 2. Constraints may be expressed in a signature richer than Δ.

A **strand** has a sequence of nodes, each of which is a *transmission*, a *reception*, or a *local* or neutral *event*. Each node n has a **message** $\mathsf{msg}(n)$. The messages are all chosen from an algebra Alg with some signature Δ. We write $s \downarrow i$ for the i^{th} node on s. We write $n_0 \Rightarrow n_1$ when $n_0 = s \downarrow i$ and $n_1 = s \downarrow i + 1$ for some s, i. We lift substitutions σ to strands pointwise: $\mathsf{msg}(\sigma(s) \downarrow i) = \sigma(\mathsf{msg}(s \downarrow i))$.

Message t **originates** at n if n is a transmission node, $t \sqsubseteq \mathsf{msg}(n)$, and for all n_0, $n_0 \Rightarrow^{+} n$ implies $t \not\sqsubseteq \mathsf{msg}(n_0)$.

A **protocol** Π is a set of strands over a signature Δ, called the roles of the protocol. Each role ρ may have some basic values that are assumed uniquely originating, $\mathsf{rl\_unique}(\rho) \subseteq \mathsf{Basic}$; this means that the corresponding value in every instance of the role will always be assumed to originate at most once. Similarly, $\mathsf{rl\_non}(\rho) \subseteq \mathsf{Basic}$ is a set of values assumed non-originating in every instance. Finally, $\mathsf{rconstr}(\rho, i)$ yields a set of equations $t = t'$ for each i less than the length of ρ. These are constraints that must be satisfied in a possible execution of Π. The constraints $\mathsf{rconstr}(\rho, i)$ may belong to an extended signature Δ'.

Each protocol Π contains special roles for specifying security goals. A **listener** role $\xrightarrow{t} \bullet$, has a single reception node documenting t's availability unprotected on the network. A **blab** role $\bullet \xrightarrow{t}$ has a transmission node documenting when a compromised value becomes available to the adversary.

The **parameters** of a role $\rho \in \Pi$ are all parameters $v \in \mathsf{Param}$ that occur in $\mathsf{msg}(\rho \downarrow i)$ for any i. The **regular strands** of Π over algebra Alg with signature expanding Δ are all substitution instances (in Alg) of all $\rho \in \Pi$. An **adversary model** is a set of strands, called *adversary strands*.

Skeletons are fragmentary executions of the regular participants, which factor out adversary behavior. A **skeleton** $\mathbb{A} = (\mathsf{nodes}, \preceq, \mathsf{non}, \mathsf{unique})$ consists of a finite set of regular nodes, a partial ordering on them, a set of values assumed non-originating, and a set of values assumed uniquely originating. These components are designed to code in the aspects of executions that we care about, namely the ordering, and what values are uncompromised ("non") or freshly chosen ("unique"). For a precise description, see [20].

Definition III.1. An *adversary recipe from messages S, using adversary choices C* is a directed, acyclic graph $\mathcal{G} = \langle \mathcal{N}, \rightarrow_G, \Rightarrow_G \rangle$ consisting only of adversary nodes, such that

1. $\mathsf{msg}(m) \in S$ if m is a reception node with no incoming communication arrow $n \rightarrow_G m$;
2. $\mathsf{msg}(m) \in C$ if $m = s \downarrow 1 \in \mathcal{N}$ is a transmission node, and the first node on s.

$\mathcal{G}$ *derives t avoiding N* iff $t = \mathsf{msg}(n)$ for some $n \in \mathcal{N}$, and $N \cap C = \emptyset$.

The *avoidance set* $\mathsf{VD}_{\mathbb{A}}$ *of* $\mathbb{A}$ defines what the adversary cannot choose when deriving values in $\mathbb{A}$; it is the set $\mathsf{non}_{\mathbb{A}} \cup \{a \in \mathsf{unique}_{\mathbb{A}} \colon a \text{ originates on some } n_0 \in \mathsf{nodes}(\mathbb{A})\}$.

A message t is *derivable at $n \in \mathsf{nodes}(\mathbb{A})$* iff it is derivable from earlier transmissions avoiding $\mathsf{VD}_{\mathbb{A}}$.

$\mathbb{A}$ is *realized* over Alg_1 iff (i) for every reception $n \in \mathsf{nodes}(\mathbb{A})$, $\mathsf{msg}(n)$ is derivable at n; and (ii) each $n \in \mathsf{nodes}(\mathbb{A})$ satisfies its constraints in Alg_1. I.e. $\sigma(t) = \sigma(t')$ in Alg_1 whenever $n = \sigma(\rho \downarrow i)$, and $t = t' \in \mathsf{rconstr}(\rho, i)$. ///

$$\begin{array}{ll} \text{Sorts:} & \mathsf{Name}, \mathsf{Data}, \mathsf{SigKey}, E, G \leq \mathsf{Basic} \leq \mathsf{Mesg} \\ \text{Functions:} & \mathsf{cat} \quad : \mathsf{TAG} \rightarrow \mathsf{Mesg} \rightarrow \mathsf{Mesg} \rightarrow \mathsf{Mesg} \\ & \mathsf{dsig} \quad : \mathsf{Mesg} \rightarrow \mathsf{SigKey} \rightarrow \mathsf{Mesg} \\ & \mathsf{sk} \quad : \mathsf{Name} \rightarrow \mathsf{SigKey} \qquad \mathsf{gexp} \colon E \rightarrow G \end{array}$$

Fig. 4. The formal DH signature Σ_0

The realized skeletons represent actual executions. The only essential addition in this section to earlier strand space treatments are the constraints $\mathsf{rconstr}(\rho, i)$.

B. The Formal Algebras

We now instantiate the definitions of the previous sections. Two algebras are syntactic structures, one without the group and field operations, and one in which they are included.

Definition III.2. Fix a set TAG. The sorts and functions of the order-sorted *formal DH signature* Σ_0 are in Fig. 4. The functions are tagged concatenation; digital signature; the signature keys of principals; and exponentiation with fixed base g. We write $\mathsf{tag}\ t_0,\ t_1$ for $\mathsf{cat}(\mathsf{tag}, t_0, t_1)$, and $[\![\, t \,]\!]_K$ for $\mathsf{dsig}(t, K)$. When the write $t_0,\ t_1$, we mean $\mathsf{nil}\ t_0,\ t_1$, for a distinguished "null" tag nil.

For each of the sorts Name, Data, SigKey, and E (but not G), we supply disjoint countable sets; Param is their union. The parameters of sort E are called the E-parameters EParam.

$\mathsf{FAlg} = \mathcal{T}_{\Sigma_0}(\mathsf{Param})$, the *formal message algebra*, is freely generated from Param by the functions of Σ_0, subject to the sort discipline. Its members are *messages* or *terms*. Basic includes the sorts Name, Data, SigKey, E, and G. a parameter $v \in \mathsf{Param}$ *occurs in* a message $t \in D$ if it is used in the inductive generation of t. The *ingredient* relation $\sqsubseteq$ is the smallest reflexive, transitive relation such that $t_0 \sqsubseteq \mathsf{tag}\ t_0,\ t_1$; $t_1 \sqsubseteq \mathsf{tag}\ t_0,\ t_1$; and $t_0 \sqsubseteq [\![\, t_0 \,]\!]_t$. ///

The key of a digital signature contributes no ingredients; only the plaintext does. For $t_1 \in \mathsf{Basic}$, "ingredient" is the identity, i.e. $t_0 \sqsubseteq t_1$ implies $t_0 = t_1$.

Every term of sort E is an E-parameter $x \in \mathsf{EParam}$. There are no G-parameters. Since $\mathsf{gexp}(e)$ will be interpreted as g^{e}, where g is a fixed generator of G, group elements are represented in the form $\mathsf{gexp}(e)$. Thus, FAlg, the terms of sort G are precisely the expressions $\mathsf{gexp}(x)$ for $x \in \mathsf{EParam}$. We also do not provide any parameters over the top sort *Mesg*; they are unnecessary for the lightweight DH protocols (see the *Well-typed message* clause of Assumption II.1). The usefulness of FAlg lies in the following lemma:

$$\begin{array}{rll} \cdot : G \times G \to G & id : \to G & inv : G \to G \\ +,\ -,\ * : E \times E \to E & 0 : \to E & exp : G \times E \to G \\ i : E \to E & 1 : \to E & g : \to G \end{array}$$

$$\mathsf{Hash} \quad : \mathsf{Mesg} \to \mathsf{Mesg} \qquad \mathsf{hash} \quad : \mathsf{Mesg} \to E$$

Fig. 5. Extended formal DH signature Σ_1, in addition to Σ_0

Lemma III.3. *To within isomorphism, there are only finitely many* Π*-skeletons over* FAlg *of bounded size.*

Definition III.4. The order-sorted *extended formal DH signature* Σ_1 consists of Σ_0 together with the group and field operators in Fig. 5, and two hash functions, into Mesg and E. We write $exp(t, e)$ as t^e.

eFAlg $= \mathcal{T}_{\Sigma_1}(\mathsf{Param})$, is freely generated from Param(FAlg) by Σ_1, subject to the sort discipline. ///

Henceforth, each Π will be over Σ_0 with constraints in Σ_1.

C. The Field-based Algebras

If q is a prime, let $\mathbb{C}_q$ and $\mathbb{F}_q$ be the cyclic group of order q and the field of characteristic q. Whenever $X \subseteq \mathsf{EParam}$, let $\mathbb{F}_q[X]$ be the ring of polynomials over X with coefficients in $\mathbb{F}_q$, and let $\mathbb{F}_q(X)$ be the field of fractions of these polynomials. Polynomials have the usual equalities, based on the laws of $\mathbb{F}_q$. These laws are not a set of equations or conditional equations, since the cancellation law has a negative premise: $e_1 \neq 0 \supset e_1 * i(e_1) = 1$. An E-parameter $x \in \mathsf{EParam}$ *occurs* in a polynomial if x has non-zero degree in it.

Definition III.5.

1. Let $\mathbb{F}$ be a field, and let the sort G contain the values $\mathsf{gexp}(v)$ for $v \in \mathbb{F}$. $\mathsf{Alg}_{\mathbb{F}}$ is freely generated from $G, \mathbb{F}$, and Param by the remaining operators, subject to the group laws and $\mathsf{gexp}(x) = g^x$. $\mathsf{Alg}_{\mathbb{F}}$ is called the *algebra of messages* over $\mathbb{F}$. When $\mathbb{F} = \mathbb{F}_q(\mathsf{EParam})$ for q a prime, we write the q-Alg for $\mathsf{Alg}_{\mathbb{F}}$.
2. For any field $\mathbb{F}$, there is a *canonical homomorphism* $h_{\mathbb{F}} \colon \mathsf{eFAlg} \rightarrowtail \mathsf{Alg}_{\mathbb{F}}$. It identifies terms of $\mathcal{T}_{\Sigma_1}$ using the identities true in $\mathbb{F}$. The restriction of $h_{\mathbb{F}}$ to FAlg is also a canonical homomorphism $h_{\mathbb{F}}^{-} : \mathsf{FAlg} \rightarrowtail \mathsf{Alg}_{\mathbb{F}}$. When $\mathbb{F} = \mathbb{F}_q(\mathsf{EParam})$, we write h_q for $h_{\mathbb{F}}$. ///

The homomorphism $h_q \colon \mathsf{eFAlg} \rightarrowtail q\text{-}\mathsf{Alg}$ maps parameters of FAlg to themselves, and reduces modulo q: $1+1+\ldots+1$, with q ones, equals 0 in q-Alg. This extends uniquely to $h_q \colon \mathsf{eFAlg} \to q\text{-}\mathsf{Alg}$. We lift $h_{\mathbb{F}}$ to skeletons by applying $h_{\mathbb{F}}$ to the message of each node. Two restrictions on lightweight protocols are:

Definition III.6. A message t is *well-typed* iff $t \in \mathsf{Param}$ is of type Name, Data, or SigKey; or $t \colon G$ is g^v for some $v \in \mathsf{EParam}$; or (recursively) t is built from well-typed messages by tagged concatenation and digital signature.

1. A protocol Π *handles well-typed messages* iff, for every role $\rho \in \Pi$, either ρ is a *blab* or *listener* role, or else every message sent or received on ρ is well-typed.
2. Π *sends role-nons* iff, for $x \in \mathsf{EParam}$, if $x \in \mathsf{rl\_non}(\rho)$, then x first occurs in a transmission $\rho \downarrow i$, and $g^x \sqsubseteq \mathsf{msg}(\rho \downarrow i)$. ///

$+v$, where $v \in$ Basic, but $v \notin G, E$

$-t \Rightarrow +\mathsf{hash}(t)$ and $-t \Rightarrow +\mathsf{Hash}(t)$

$-t \Rightarrow -K \Rightarrow +[\![t]\!]_K$

$-[\![t]\!]_K \Rightarrow +t$

$-t_1 \Rightarrow -t_2 \Rightarrow +\mathsf{tag}\ t_1,\ t_2$

$-(\mathsf{tag}\ t_1,\ t_2) \Rightarrow +t_1 \Rightarrow +t_2$

$+\mathsf{nop}$

$-a \Rightarrow +\mathsf{uop}(a)$

$-a \Rightarrow -b \Rightarrow +\mathsf{bop}(a, b)$

$+x$, where $x \in$ EParam

Fig. 6. Adversary strands. Transmission of t is $+t$; reception of t is $-t$. Nullary, unary, and binary operators written nop, uop, and bop, resp.

Hashes generate algebraically unpredictable values. But since each $\mathbb{F}_q$ is finite, we can engineer polynomials to define them in one $\mathbb{F}_q$. However, those polynomials fail as q varies.

Assumption III.7. Let $\mathsf{hash}(t_1), \ldots, \mathsf{hash}(t_k)$ be distinct terms; let $x_1, \ldots, x_j \in \mathsf{EParam}$; and let p be a polynomial in $x_1, \ldots, x_j, \mathsf{hash}(t_1), \ldots, \mathsf{hash}(t_k)$ that is non-trivial in the hashes. Then $p = 0$ is valid in $\mathbb{F}_q$ for at most finitely many q.

Lightweight protocols use hashes only in the key computation, as in Fig. 2, which constraints $\mathsf{rconstr}(\rho, i)$ formalize.

D. Adversary Model

We now specify the adversary model of interest to us. Although it consists of the same adversary strands across all of our algebras, it leads to different realized skeletons over the different q-Algs, since whether the right message is derivable depends on equality in the relevant algebra.

Definition III.8 (Adversary Strands)**.** Adversary strands of Adv fall into two groups. The *symbolic rules* [23] allow the adversary to originate values, to create signatures with access to plaintext and key, to access the message within a digital signature, to create hashes from the plaintext, and to concatenate or separate messages (Fig. 6, left).

Next, for the *algebraic rules*, if op is a nullary, unary, or binary operator of Def. III.4, and a, b are values in G, E, then the adversary can execute op on the given values. If $x \colon \mathsf{EParam}$ is a parameter of sort E, he can send x (Fig. 6, right). ///

The adversary obeys a "normal proof" property.

Lemma III.9 (Normal form [23])**.** *If t is penetrator derivable in* Alg *from S using C, then it may be derived using a* $\mathcal{G}$ *in which, if* n_0 *lies on a* Signature Access *or* Separation*, and* n_1 *lies on any strand of another kind,* $n_1 \not\preceq_{\mathcal{G}} n_0$.

Any value is derivable if it has previously originated.

Lemma III.10. *Suppose that* $n_0 \prec_{\mathbb{A}} n_1$*, where* n_0 *is a transmission node. If* $t \sqsubseteq \mathsf{msg}(n_0)$*, then t is derivable at* n_1.

Definition III.11. Fix $N \subseteq \mathsf{EParam}$; let $P = \{p_1, \ldots, p_n\}$; and let $S \subseteq q\text{-}\mathsf{Alg}$. An *$N$-avoiding (linear) combination* of P with S is a polynomial

$$e = f_0 + f_1 p_1 + \cdots + f_n p_n$$

where each $f_i \colon E$ is derivable from S avoiding N. ///

Lemma III.12. *Fix* q-Alg *and* $N \subseteq \mathsf{EParam}$*, and let* $S \subseteq q\text{-}\mathsf{Alg}$*. Let* $P = \{e : E \mid g^e \sqsubseteq S\}$.

Any G-value g^e *is derivable from S avoiding N iff e is an N-avoiding combination of* $P \cap N$ *with S.*

E. Adversary strategies

The adversary plays a game. He wants to exhibit attacks against a protocol Π. Thus, he will select an FAlg-skeleton $\mathbb{A}$ in which something occurs that would be contrary to the goals of Π. This could be disclosure of some intended secret t, as indicated by a listener node $\xrightarrow{t} \bullet$, or it could be a failure of authentication, as indicated by a run of one participant with no matching run of the peer (Section IV-A).

To win, the adversary must show how to realize $\mathbb{A}$ in different algebras. He supplies a recipe for each regular reception node n. This recipe generates $\mathsf{msg}(n)$ using transmissions on regular nodes $m \prec_{\mathbb{A}} n$ as desired, making any adversary choices compatible with the assumptions $\mathsf{unique}_{\mathbb{A}}$ and $\mathsf{non}_{\mathbb{A}}$.

An *adversary strategy* is a map f from reception nodes $n \in \mathsf{nodes}(\mathbb{A})$ to adversary recipes. An adversary strategy f *wins* for $\mathbb{A}$ at q if for each reception node $n \in \mathsf{nodes}(\mathbb{A})$, $f(n)$ derives $\mathsf{msg}(n)$ avoiding the avoidance set of $\mathbb{A}$, and satisfies all the role constraints (Defn. III.1). This depends on q-Alg, which defines the equalities between what $f(n)$ produces, and what n must receive. The adversary *wins* for $\mathbb{A}$ against Π if, for infinitely many q, some strategy f wins for $\mathbb{A}$ at q.

The adversary model generates any polynomial involving "unrestricted" values $v \in \mathsf{EParam}$ where $v \notin \mathsf{non}_{\mathbb{A}}$. For values $v \in \mathsf{non}_{\mathbb{A}}$ available only in the form g^v, the adversary can manipulate them only as described in Lemma III.12.

Whenever a value $v \in \mathsf{EParam}$ is chosen by a regular participant, and used to prepare a g^v which originates on this strand, we will regard v as being selected randomly and independently of all other parameters to the strands in this execution. If $v \notin \mathsf{non}_{\mathbb{A}}$, then the adversary may be able to gain access to the value chosen, e.g. via a hacked operating system. However, we will regard all of these values as indeterminates or field extension elements. This means that a successful adversary strategy f must always produce the right value, uniformly in all values that could be chosen for v. This genericity applies only to $v \in \mathsf{EParam}$; for other sorts, unique and non express the assumptions constraining the adversary.

Definition III.13. A parameter $x \in \mathsf{EParam}$ is *regular-chosen* (or *r.c.*) on $n = \rho \downarrow i$ iff n is a transmission node or local node, and n is the earliest node in which x occurs.

$\mathsf{R\_vals}(\mathbb{A}) = \{\sigma(x) \colon \exists n \in \mathsf{nodes}(\mathbb{A}) \,.\, n = \sigma(\rho \downarrow i)$ and x is r.c. on $\rho \downarrow i\}$.

$\mathsf{A\_params}(\mathbb{A}) = \{v \in \mathsf{EParam} \colon v$ occurs in $\mathbb{A}\} - \mathsf{R\_vals}(\mathbb{A})$.

If $\mathbb{A}$ is a FAlg skeleton, then $\mathbb{A}$ *makes distinct choices* iff any two distinct r.c. parameters take different values in $\mathbb{A}$. More precisely, for all $n, n' \in \mathsf{nodes}(\mathbb{A})$, letting $n = \sigma(\rho) \downarrow i$ and $n' = \sigma'(\rho') \downarrow j$, for all x r.c. on $\rho \downarrow i$ and x' r.c. on $\rho' \downarrow j$, $\sigma(x) = \sigma'(x')$ implies $\rho = \rho'$, $i = j$, and $x = x'$. ///

Our linearity assumption on lightweight protocols is formalized in terms of $\mathsf{A\_params}$.

Definition III.14. Π has *linear adversary contributions* if each polynomial p occurring in any $s = t$ in $\mathsf{rconstr}(\rho, i)$ has total degree 0 or 1 for all $\mathsf{A\_params}$.

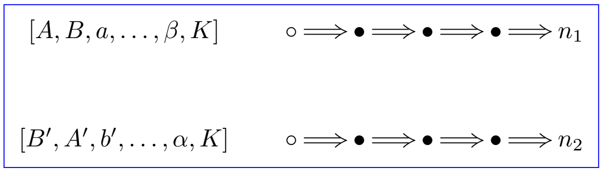

$[A, B, a, \ldots, \beta, K]$
$\circ \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow n_1$
$[B', A', b', \ldots, \alpha, K]$
$\circ \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow \bullet \Longrightarrow n_2$

Fig. 7. Strong implicit authentication premise. Assume $a, b', \mathsf{sk}(\mathsf{CA}) \in \mathsf{non}$. Conclusion is $A' = A$

IV. GOAL LANGUAGES

Security goals are formulas of the form:

$$\forall \overline{x} \,.\, (\Phi \supset \bigvee_i \exists \overline{y_i} \,.\, \Psi_i) \qquad (1)$$

where Φ and each Ψ_i are conjunctions of atomic formulas [19]. Any geometric sequent (i.e. implication between positive existential formulas) can be rewritten in this form. The vocabulary for the atomic formulas is identical to the goal language $\mathcal{GL}(\Pi)$ in our non-Diffie-Hellman protocol analysis [19].

Small model arguments apply to formulas of the form of Eqn. 1. Suppose that S is a set of small models $\mathbb{A}$ such that for every $\mathbb{B}$ model satisfying Φ, there is an $\mathbb{A} \in S$ and a homomorphism $h \colon \mathbb{A} \to \mathbb{B}$. Then, if one of the formulas Ψ_i is found to be true in each $\mathbb{A} \in S$, homomorphisms $h \colon \mathbb{A} \to \mathbb{B}$ will preserve that Ψ_i. Hence, checking $\bigvee_i \exists \overline{y_i} \,.\, \Psi_i$ in the models in S ensures that Eqn. 1 holds in general.

The signature of $\mathcal{GL}(\Pi)$ is sparser than Σ_1, or even Σ_0, but skeletons determine the models of $\mathcal{GL}(\Pi)$. The domains consist of the messages in a message algebra Alg, together with the regular nodes of the skeleton. The skeletons of Π are axiomatizable in richer languages, though not in the language $\mathcal{GL}(\Pi)$, which we will not linger over doing here.

A. Some Protocol Goals

We illustrate the language by examples. The first formalizes the implicit authentication goal in a strong form central to many of our protocols, that entails "preventing unknown key-share attacks" [4], [24], [31]. Fig. 7 illustrates the assumption Φ of our assertion. In it, n_1 is the last node of an initiator strand, and n_2 is the last node of a responder strand. The "self" parameter of n_1 is the variable A, meaning that principal A executes it. The "peer" parameter is the variable B, meaning that A believes this run to be an exchange with principal B; more concretely, B's name appears in the certificate c_B used in the next-to-last node. On n_2's strand, the self and peer are the variables B', A'. The "key" parameters defined by the two strands are the same variable K; i.e. the assumption is that these two strands have agreed on the key. The long term private values are represented by variables a, b, resp.

$$\begin{aligned}
&\texttt{InitLast}(n_1) \wedge \texttt{RespLast}(n_2) \wedge \\
&\quad \texttt{Self}(n_1, A) \wedge \texttt{Peer}(n_1, B) \wedge \texttt{Key}(n_1, K) \wedge \texttt{MyLT}(n_1, a) \wedge \\
&\quad \texttt{Self}(n_2, B') \wedge \texttt{Peer}(n_2, A') \wedge \texttt{Key}(n_2, K) \wedge \texttt{MyLT}(n_2, b') \wedge \\
&\qquad \texttt{Non}(a) \wedge \texttt{Non}(b') \wedge \texttt{Non}(\mathsf{sk}(\mathsf{CA})) \qquad (2) \\
&\qquad \supset A = A'
\end{aligned}$$

The desired conclusion is that $A = A'$. Since $A = A'$ depends on the CA not recertifying the long term value $g^{a'}$ of A' to a second principal, the mathematical core is that $\alpha = a$. The

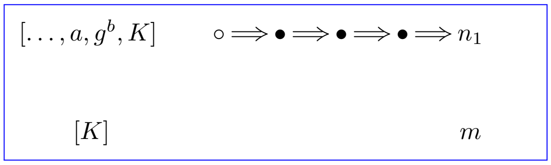


Fig. 8. A key secrecy goal. Assume $a, b \in \mathsf{non}$. Conclusion is `falsehood`.

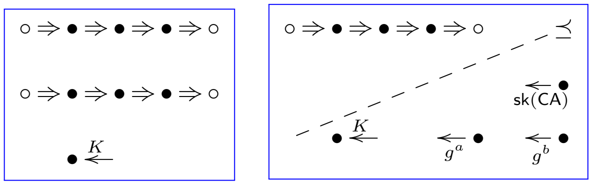


Fig. 9. Forward secrecy, with $x, y \in \mathsf{non}$: (l) weak; (r) strong, with $g^a, g^b, \mathsf{sk}(\mathsf{CA}) \in \mathsf{unique}$

authentication goal of the responder, who would like to infer that $B = B'$ is symmetric. Eqn. 2 is an instance of Eqn. 1 with only one disjunct, and a vacuous variable list $\overline{y_i}$.

The "unknown key share resistance" goal is somewhat weaker than Eqn. 2. It includes additional assumptions that the ephemeral secrets are also uncompromised. We will derive Kaliski's attack on it for MQV in Section VII.

Second, we formalize a key secrecy property, illustrated in Fig. 8. Here the desired goal is that this diagram cannot occur. We express the goal explicitly as the formula:

$$\begin{aligned}&\mathtt{InitLast}(n_1) \wedge \mathtt{Lsn}(m) \wedge \mathtt{Key}(n_1, K) \wedge \mathtt{Hear}(m, K) \wedge \\ &\quad \mathtt{MyLT}(n_1, a) \wedge \mathtt{Non}(a) \wedge \mathtt{YourLT}(n_1, b) \wedge \mathtt{Non}(b) \\ &\qquad\qquad \supset \mathtt{falsehood}\end{aligned} \tag{3}$$

Here, the conclusion is `falsehood`, namely the disjunction with zero disjuncts. $\mathtt{Lsn}(m)$ means that m is a listener node, i.e. one that simply receives a value to document its availability. $\mathtt{Hear}(m, K)$ means that K is the value that the listener node m hears.

Conclusions may also use existential quantifiers to assert the presence of new items, especially additional regular nodes, as in "explicit" authentication properties.

An important security goal of DH protocols is forward secrecy. Suppose that after a session of A with peer B computing a session key K, all of the long-term secrets of A and B, and of the CA, are compromised. Can the adversary compute K?

The *weak* version of forward secrecy assumes that both participants contributed an ephemeral value; no assumption is needed about the long-term values $g^a, g^b, \mathsf{sk}(\mathsf{CA})$. The left side of Fig. 9 (l) illustrates the assumption, with K received on a listener node. The goal—achieved in many DH protocols—is to ensure that this diagram cannot occur in a real execution.

The *strong* version assumes instead that the long term values *were* secret when the session occurred, and were subsequently exposed. We model this in Fig. 9 (r) by assuming that the long-term values $g^a, g^b, \mathsf{sk}(\mathsf{CA}) \in \mathsf{unique}$ are uniquely originating, and originate after the session. Again, the goal is to ensure that this cannot occur. If a protocol achieves explicit authentication, rather than merely implicit authentication, that implies occurrence of a second strand, representing the presence of the peer's session. Thus, the assumption of weak forward secrecy is also satisfied. Generally, this is the only way to achieve strong forward secrecy.

Functions:	$\mathtt{sk}(a)$	$\mathtt{inv}(k)$	
Relations:	$\mathtt{Preceq}(m, n)$	$\mathtt{Coll}(m, n)$	$=$
	$\mathtt{Unq}(v)$	$\mathtt{UnqAt}(n, v)$	$\mathtt{Non}(v)$

Fig. 10. Protocol-independent predicates of the languages $\mathcal{GL}(\Pi)$

In the cryptographic literature, e.g. [26], this property may be expressed as a real-or-random challenge for the adversary, and in somewhat different terms. However, the differences seem inessential (see also below).

Both varieties of forward secrecy are directly expressible in this language. Other expressible goals include resisting impersonation attacks; resisting unknown key share attacks; other variants of implicit authentication; and explicit authentication. A strength of this language is that it expresses the bulk of the properties that interest us in a framework that is unproblematically first order and linguistically very spare.

The most important gap concerns indistinguishability properties. They require a quite different language and semantics, since they are simply not properties of a single execution [9]. However, with the help of Cortier and Kremer, we have proved, for all lightweight DH protocols, that the most relevant indistinguishability property, namely the real-or-random property for the key, is equivalent to key disclosure [12].

B. *Defining the Languages*

For each protocol Π, $\mathcal{GL}(\Pi)$ is a single-sorted first order language. These languages match the languages for non-Diffie-Hellman protocols in [22], and their semantics, i.e. the definition of the satisfaction relation, is unchanged. $\mathcal{GL}(\Pi)$ says nothing about the structure of Π's messages. Instead, it classifies nodes by which action they are, on which role, and how they instantiate the role's parameters. This has two main advantages. First, it mean goals expressed in $\mathcal{GL}(\Pi)$ can be preserved when message format changes. Second, it makes the goal formulas simpler. A decidability result would be hopeless if there were existentially quantified equations in the full vocabulary of fields. $\mathcal{GL}(\Pi)$'s vocabulary, which we write in typewriter font, has two parts.

The protocol-independent part. This part, shown in Fig. 10, is shared by the languages for all Π. Its function symbol $\mathtt{sk}(a)$ returns a's private signature key. It also includes the predicate symbols shown in Table 10. $\mathtt{Preceq}(m, n)$ expresses that node m precedes node n. $\mathtt{Coll}(m, n)$ expresses that nodes m and n lie on the same strand. $\mathtt{Unq}(v)$ holds if the basic value v originates uniquely. $\mathtt{UnqAt}(n, v)$ holds if v originates uniquely, and originates at the node n. $\mathtt{Non}(v)$ holds if v is non-originating. As always, $=$ is equality. This vocabulary expresses structural properties of skeletons.

The protocol-specific part. The protocol-specific vocabulary consists of two kinds of predicates. *Role position predicates*

$R(n)$ assert that node n is a node lying at a particular position on a strand that is an instance of that regular role. For instance, `RespFirst`(n) asserts that n is an instance of the first node of a responder role. We have used `RespLast` and `InitLast` as role position predicates in Eqns. 2–3.

The second kind of predicate concerns the values of the parameters. A *parameter predicate* $P(n, v)$ asserts that node n is formed by instantiating a particular parameter of its role with the value v. The same parameter predicate may be used for different roles, so long as—whenever a node may be viewed as lying on instances of two different roles—it satisfies the same parameter predicates for both of those roles. We have used `Self`, `Peer`, `Key`, `YourLT`, `MyLT`, and `Hear` as parameter predicates in Eqns. 2–3.

The language $\mathcal{GL}(\Pi)$ thus expresses almost nothing about the structure of messages, and certainly none of the operations on E, but focuses only on the parameters.

The semantics of $\mathcal{GL}(\Pi)$ is unchanged from [22], and we omit it. The formulas satisfied for $\mathbb{A}, \eta$ depend only on the nodes within $\mathbb{A}$. What a strand would do "after" the part in $\mathbb{A}$ never changes the truth value of any atomic formula. Indeed:

Lemma IV.1 ([22])**.** *Let φ be a positive existential formula, and let $H \colon \mathbb{A} \to \mathbb{B}$ for* Alg*-skeletons $\mathbb{A}, \mathbb{B}$. If $\mathbb{A} \models_\eta \varphi$ then $\mathbb{B} \models_{(H \circ \eta)} \varphi$.*

C. *Relating* FAlg *and q-*Alg *skeletons*

If the only tests we have available are positive existential formulas of $\mathcal{GL}(\Pi)$, q-Alg skeletons and the corresponding FAlg skeletons are difficult to distinguish. Indeed, $\mathcal{GL}(\Pi)$ was designed to express no more than is needed for security goals. We use FAlg skeletons and q-Alg skeletons in complementary ways: FAlg skeletons enumerate the relevant ways to satisfy or falsify security goals, and q-Alg skeletons to assess whether those patterns can occur in realized executions.

Definition IV.2. Let $\mathbb{F}$ be a finite or countable field. Fix a bijection u from $\mathbb{F}(\mathsf{Param})$ to the E-atoms EParam. Extend u to map $\mathsf{Alg}_\mathbb{F}$ to FAlg injectively, acting as the identity on sorts other than E and introducing gexp by: $\mathsf{gexp}(x) = g^{(u^{-1}(x))}$.

Lift u to map $\mathsf{Alg}_\mathbb{F}$ skeletons to FAlg skeletons by preserving the nodes and precedence relations, and determining the messages using u. ///

Lemma IV.3. *Let $\Theta \in \mathcal{GL}(\Pi)$ be any formula and let $\mathbb{B}$ be any $\mathsf{Alg}_\mathbb{F}$-skeleton. Then $\mathbb{B} \models_\eta \Theta$ iff $u(\mathbb{B}) \models_{u \circ \eta} \Theta$.*

Proof: For atomic formulas, this holds essentially because $\mathcal{GL}(\Pi)$ expresses none of the field structure of $\mathbb{F}$. Preservation under propositional operators is routine, and preservation under quantifiers holds because u is a bijection preserving the open formula (by the induction hypothesis). ■

Of these skeletons, the ones that interest us are the FAlg skeletons that make distinct choices (Def. III.13), and the q-Alg skeletons that leave these R_vals as parameters.

Definition IV.4. Let Θ be a security goal as in Eqn. 1.

A q-Alg skeleton $\mathbb{B}$ is a *q-counterexample* to Θ iff (i) $\mathbb{B} \not\models \Theta$; (ii) $\mathsf{R\_vals}(\mathbb{B}) \subseteq \mathsf{EParam}$; and (iii) $u(\mathbb{B}) = \mathbb{A}$ makes distinct choices.

An FAlg skeleton $\mathbb{A}$ is a *counterexample* to Θ if there is an infinite family $\{\mathbb{B}_q \colon q \in I\}$ of q-counterexamples for Θ such that, for every $q \in I$, $\mathbb{A} = u(\mathbb{B}_q)$. ///

V. BOUNDING THE SIZE OF COUNTEREXAMPLES

In this section, given a lightweight Diffie-Hellman protocol Π and a geometric sequent $\forall \overline{x} \,.\, (\Phi \supset \bigvee \exists \overline{y} \,.\, \Psi_i)$, we will calculate a bound b depending only on Π and the antecedent Φ. For every q-Alg-skeleton $\mathbb{B}$ that satisfies Φ, we can select a subset of the regular strands of $\mathbb{B}$ of cardinality $\leq b$. These strands form a realized q-Alg-skeleton $\mathbb{A}$ themselves, where $\mathbb{A} \to \mathbb{B}$. Moreover, Φ is also satisfied in $\mathbb{A}$.

If $\mathbb{A}$ also satisfies some positive existential formula Ψ_i, then the inclusion ensures that $\mathbb{B}$ also satisfies it. Or, by contraposition, if $\mathbb{B}$ falsifies $\Phi \supset \bigvee \exists \overline{y} \,.\, \Psi_i$, then so does $\mathbb{A}$. Moreover, $\mathbb{A}$ is small, containing $\leq b$ regular strands.

A. *Criterion for Realized Skeletons*

We say that t *is a component of* t' if either $t = t'$, or else t' is a tagged concatenation $\mathtt{tag}\ t_0, \ \ldots, \ t_j$, and t is recursively a component of one of the t_i. Recall that $\mathsf{VD}_\mathbb{A}$ is the avoidance set of $\mathbb{A}$, the set of values that the adversary cannot choose, consistent with the assumptions of $\mathbb{A}$ (Defn. III.1).

Lemma V.1. *Let $\mathbb{A}$ be a q-*Alg *skeleton for a lightweight protocol Π. $\mathbb{A}$ is realized if and only if, for every reception node $n \in \mathsf{nodes}(\mathbb{A})$, and for every component c of $\mathsf{msg}(n)$ there is a set S of hashes $\mathsf{hash}(t), \mathsf{Hash}(t)$ such that:*

if $c = g^e$ is of sort G, then *e is an N-avoiding combination of $\{d \colon g^d$ originates on some $n_0 \prec_\mathbb{A} n\}$ with S.*

if c is a base value of non-G sort, then *either c originates on some $n_0 \prec_\mathbb{A} n$, or $c \notin \mathsf{VD}_\mathbb{A}$.*

if $c = [\![\, t \,]\!]_K$ is a digital signature, then *either c originates on some $n_0 \prec_\mathbb{A} n$; or else $K \notin \mathsf{non}_\mathbb{A}$, and every component of t recursively satisfies these conditions.*

if $c \in S$ is a hash $\mathsf{Hash}(t)$ or $\mathsf{hash}(t)$, then *either c originates on some $n_0 \prec_\mathbb{A} n$; or else every component of t recursively satisfies these conditions.*

Proof: By definition, if c is a component of $\mathsf{msg}(n)$ for $n \in \mathsf{nodes}(\mathbb{A})$, then $c \notin \mathsf{non}_\mathbb{A}$. Hence, if the conditions are fulfilled for every c, then (by Lemmas III.12 and III.10) each received message can in fact be derived from earlier transmissions, using adversary choices that avoid $\mathsf{VD}_\mathbb{A}$.

But otherwise, the adversary cannot derive c from earlier transmissions, so that $\mathbb{A}$ is not realized. ■

B. *Simple digital signatures*

We now define the *Simple digital signatures* property.

Definition V.2. Π has *simple digital signatures* if there exists a finite partition of the digitally signed messages of FAlg, and a strict ordering $\lhd$ on the partition classes such that, for all digitally signed messages d_1 and d_2:

1. If $d_1 = \sigma(d_2)$ for some σ, then $\mathtt{class}(d_1) = \mathtt{class}(d_2)$.
2. Let $\rho \in \Pi$ be a role such that $d_1 \sqsubseteq \rho \downarrow i$ and $d_2 \sqsubseteq \rho \downarrow j$.

If $i < j$, then $\texttt{class}(d_1) \lhd \texttt{class}(d_2)$. If $i = j$, then $\texttt{class}(d_1) = \texttt{class}(d_2)$.

If there are k partition classes, then Π has simple digital signatures *of index k*. ///

For instance, the IADH protocols of Fig. 1, we partition digital signatures into certificates $[\![\, \mathsf{cert}\ g^{\pi},\ P \,]\!]_{\mathsf{sk(CA)}}$ and all others. Only certificates appear in the protocol, and each role handles certificates on just one node; thus Defn. V.2 is satisfied. In the STS variant of Fig. 3, each role handles a digitally signed unit with tag ri before one with tag ir. Thus, we may partition digital signatures into these two classes, as well as all others, stipulating that $[\![\, \mathsf{ri}\ s \,]\!]_K \lhd [\![\, \mathsf{ir}\ t \,]\!]_K$. Thus, Fig. 3 satisfies Defn. V.2 also.

Lemma V.3. *Let Π have simple digital signatures of index k. Suppose that $\mathbb{A}$ is a Π skeleton, and $n_1, m_1, n_2, \ldots, n_j, m_j \in \mathsf{nodes}(\mathbb{A})$. Then $j \le k$ if:*

1. *Each n_i is a transmission node and m_i is a reception;*
2. *For each i, $m_i \Rightarrow^+ n_{i+1}$;*
3. *For each i, $n_i \preceq_{\mathbb{A}} m_i$, and there is a t and a $K \in \mathsf{non}_{\mathbb{A}}$ such that $[\![\, t \,]\!]_K \sqsubseteq \mathsf{msg}(n_i)$ and $[\![\, t \,]\!]_K \sqsubseteq \mathsf{msg}(m_i)$.*

Proof: $\texttt{class}(n_i) = \texttt{class}(m_i)$, and $\texttt{class}(m_i) \lhd \texttt{class}(n_{i+1})$, so each transmission/reception node pair reduces the available partition classes by one. ■

C. Bounding sizes

If Φ is a conjunction of atomic formulas in $\mathcal{GL}(\Pi)$, we will write $\mathsf{ndv}(\Phi)$ for the node variables of Φ, i.e. the set of variables x that appear in Φ as argument to a role predicate, the collinear predicate `Coll`, or the precedence predicate `Preceq`; or as first argument to a parameter predicate; or as second argument to the `UnqAt` predicate. These are precisely the variables x such that $\eta(x) \in \mathsf{nodes}(\mathbb{A})$ for every $\eta, \mathbb{A}$ such that $\mathbb{A} \models_\eta \Phi$. Thus, $\mathsf{ndv}(\Phi)$ is the set of nodes that Φ is "talking about."

We also write $\mathsf{nonv}(\Phi)$ for the set of variables x that appear in Φ as argument to the Non predicate. These are the non-originating values that Φ is "talking about."

We also use the notion of the *width* of a protocol Π; it is one greater than the largest number of components that are digital signatures in a single node along any role of Π. For instance, the protocol shown in Fig. 1 has width 3, because the next-to-last node on each strand receives two certificates c_A, c_B. Let $r = \max_{\rho \in \Pi} |\mathsf{rl\_non}(\rho)|$ be the maximum number of role-non values on any role $\rho \in \Pi$. We say that a skeleton $\mathbb{A}$ is *b-bounded* if the number of regular strands in $\mathbb{A}$ is $\le b$.

Theorem V.4 (Boundedness)**.** *Let Π be a lightweight protocol, of width w, with simple digital signatures of index k. Let Φ be a conjunction of atomic formulas. Let $\mathbb{B}$ be a realized q-Alg-skeleton for Π, and η an environment such that $\mathbb{B} \models_\eta \Phi$. There is a b-bounded realized subskeleton $\mathbb{A}$ of $\mathbb{B}$ such that $\mathbb{A} \models_\eta \Phi$, where*

$$b = |\mathsf{ndv}(\Phi)| \cdot (1 + |\mathsf{nonv}(\Phi)| + r|\mathsf{ndv}(\Phi)|) \cdot w^k.$$

Proof sketch: Define $\mathbb{A}_0$ to be the substructure of $\mathbb{B}$ containing a strand s in $\mathbb{B}$ only if $\eta(v)$ lies on s, for some $v \in \mathsf{ndv}(\Phi)$. $\mathbb{A}_0$ may not be realized. Lemma V.1 tells what to add from $\mathbb{B}$ to build a realized $\mathbb{A}$ which is a substructure of $\mathbb{B}$. These are nodes at which needed messages originate, either g^x for $x \in \mathsf{non}$, or digital signatures. We add origination nodes for g^x first, and digital signatures after. Since Π sends its role-nons (Defn. III.6.2), we will never add a digital signature that requires yet more g^x origination nodes. Finally, Lemma V.3 bounds the number of backward steps that adding digital signatures can require, and w is a branching factor. ■

Corollary V.5. *Let Π be lightweight; let Θ be a security goal. There exists a b such that, for all primes q, if there is any realized q-Alg skeleton falsifying Θ, then there is a b-bounded realized q-Alg skeleton falsifying Θ.*

Indeed, if for infinitely many q, there is a q-counterexample to Θ, then some b-bounded FAlg skeleton $\mathbb{A}$ is a counterexample to Θ.

The second claim holds because, there being only finitely many b-bounded FAlg skeletons, we must use the same one for infinitely many q. Its converse is immediate.

VI. FINDING REALIZED INSTANCES

We now use constraint-solving methods [10], [36] followed by Gaussian elimination, to determine whether a given FAlg skeleton A is a q-counterexample to a goal Θ.

A. Constraints

We will define two kinds of constraints: (i) derivability constraints $S \Vdash_N t$, and (ii) equality constraints $t_1 = t_2$. In the first kind of constraint S is a finite set of messages, t, t_1, t_2 are messages and $N \subseteq \mathsf{Basic}$. Such a constraint expresses the requirement that message t has to be explained based on the messages S sent earlier, avoiding the restricted values in N. The second kind of constraint expresses either a pattern-matching problem of the kind familiar from typical symbolic analysis or an algebraic equation whose variables range over rational functions with coefficients in various finite fields. As the formal development illustrates, different algebraic variables will range over different fields, containing different field extension elements, so a system of constraints also includes a map ARestr to control this choice.

We work relative to a fixed partition of EParam into regular-controlled parameters R_params and adversary-controlled parameters A_params. The A_params are like variables; the adversary wants to instantiate them to satisfy the constraints. Not all R_params can be used in the derivation: If $x \in \mathsf{non}$, then x cannot be used. If $x \in \mathsf{unique}$, then x can be used only if it was previously transmitted, or if it is transmitted nowhere in A. In the latter case, its sole point of origination can be an adversary node.

Transformations reduce derivability constraints to sets of equations. Hashes and digital signatures intertwine the process of reducing to equations and the process of solving the equations themselves. For example, consider the question of whether the message $g^{\mathsf{hash}(t_1)+\mathsf{hash}(t_2)}$ is derivable in some q-Alg from a set S of messages. One way this is possible is for each of $g^{\mathsf{hash}(t_1)}$ and $g^{\mathsf{hash}(t_2)}$ to be derivable. But another alternative is that t_1 and t_2 are equal in q-Alg and, say, $g^{\mathsf{hash}(t_1)}$ is derivable. This motivates the E-hashes transformation below.

In the standard constraints-based approaches to protocol analysis a solution to a constraint $S \Vdash_N t$ is a substitution θ over the $\mathsf{A\_params}$ such that the message $\theta(t)$ is derivable by the adversary from the messages in $\theta(S)$. One looks for a simultaneous solution to all of these constraints.

Our situation is more delicate, since we cannot work over a single message algebra. The notion of adversary derivability only makes sense in specific message algebras q-Alg, since the notion of equality varies with q. The potential solutions take values in eFAlg, so we compose such a solution θ with the canonical homomorphism $h_q : \mathsf{eFAlg} \to q\text{-}\mathsf{Alg}$; then ask whether $(h_q \circ \theta)(t)$ is derivable from $(h_q \circ \theta)(S)$ avoiding $\mathsf{Avoid}_{\mathbb{A}}(S)$ in q-Alg.

Thus, a substitution θ may be a solution at some q-Alg and not others; we focus on the θ that succeed at infinitely many q. Remarkably, we find that if there are θ_q in infinitely many q-Alg, there is a single θ that works for infinitely many q-Alg.

The adversary's choices at type E can be viewed as ranging over the field obtained by viewing EParam as a set of indeterminates adjoined to $\mathbb{F}_q$. Certain basic values will be assumed to be non-originating or uniquely originating, and this will be reflected in the fact that we will not allow adversary-chosen variables to range over all of this field. This is the role of the restriction sets $\mathsf{ARestr}_{\mathcal{T}}(\alpha)$ below (see Definition VI.1).

We write $\mathbb{F}_q(\mathsf{EParam})$ for the field of quotients of the polynomial ring $\mathbb{F}_q[\mathsf{EParam}]$, and if X is a subset of EParam then $\mathbb{F}_q(\mathsf{EParam} \setminus X)$ is the the field of quotients of the polynomial ring excluding X.

Definition VI.1 (Constraint System)**.** Let $\mathsf{A\_params}, \mathsf{R\_params}$ partition EParam. A *constraint system* $\mathcal{T}$ is given by the following data, where S is a finite set of messages, t, t_1, t_2 are messages, and v is an $\mathsf{R\_param}$:

- a set of *derivability constraints* $S \Vdash_N t$
- a set of an *equational constraints* $t_1 = t_2$, including a set of *hash abbreviations* $v = \mathsf{hash}(t)$
- a *restriction function* $\mathsf{ARestr}\colon P \to \mathcal{P}(\mathsf{R\_params})$

We write elements of $\mathsf{A\_params}$ as lowercase Greek letters.

Definition VI.2 (Solutions)**.** A function $\theta : \mathsf{A\_params} \to \mathsf{eFAlg}$ is a *q-solution* to $\mathcal{T}$ if

- If $S \Vdash_N t \in \mathcal{T}$, then $(h_q \circ \theta)(t)$ is derivable from $(h_q \circ \theta)(S)$ in q-Alg avoiding N, as in Definition III.1.
- If $t_1 = t_2 \in \mathcal{T}$, then in q-Alg, $(h_q \circ \theta)(t_1) = (h_q \circ \theta)(t_2)$.
- For each $\alpha \in \mathsf{A\_params}$, $(h_q \circ \theta)(\alpha) \neq 0$.
- $\theta(\alpha) \in \mathbb{F}_q(\mathsf{EParam} \setminus \mathsf{ARestr}(\alpha))$.

$\mathcal{T}$ is *infinitely-often solvable* if, for infinitely many q, $\mathcal{T}$ has a q-solution. ///

We require non-0 solutions for $\alpha : E$, because most protocols require checking that a received group element $g^{\alpha} \neq g^{0}$.

B. Construction of an Initial Constraint System

Given an FAlg skeleton $\mathbb{A}$ for a protocol Π we can construct a system expressing the constraints for $\mathbb{A}$ to be "fleshed out" to a realized skeleton. We need to reflect the origination assumptions on $\mathbb{A}$. Let $U_{\mathbb{A}} = \mathsf{unique} \cap \{a \in \mathsf{Basic}\colon a \text{ originates on some } n \in \mathsf{nodes}(\mathbb{A})\}$, and let $O(n) = \{a \in \mathsf{Basic}\colon a \text{ originates on some } m \preceq_{\mathbb{A}} n\}$. Define $\mathsf{Avoid}_{\mathbb{A}}(n)$ to be $\mathsf{non}_{\mathbb{A}} \cup (U_{\mathbb{A}} - O(n))$; these are the things that started restricted and have not yet been released. $\mathsf{Avoid}_{\mathbb{A}}$ is non-increasing through time; $n_0 \preceq_{\mathbb{A}} n_1$ implies $\mathsf{Avoid}_{\mathbb{A}}(n_1) \subseteq \mathsf{Avoid}_{\mathbb{A}}(n_0)$.

Definition VI.3. Let $\mathbb{A} = (\mathsf{nodes}, \preceq, \mathsf{non}, \mathsf{unique})$ be a Π-skeleton over FAlg. Let $\mathsf{A\_params}, \mathsf{R\_params}$ partition EParam into two infinite sets such that $\mathsf{A\_params}(\mathbb{A}) \subseteq \mathsf{A\_params}$ and $\mathsf{R\_params}(\mathbb{A}) \subseteq \mathsf{R\_params}$ (Defn. III.13). The *constraint system $\mathcal{T}$ for* $(\mathbb{A}, \Pi)$ is generated as follows.

- If $n \in \mathsf{nodes}(\mathbb{A})$ is a reception, $\mathcal{T}$ contains $S \Vdash_N \mathsf{msg}(n)$ where $S = \{n' \prec_{\mathbb{A}} n\colon n' \text{ is a transmission node }\}$ contains all earlier transmissions, and $N = \mathsf{Avoid}_{\mathbb{A}}(n)$.
- If $n \in \mathsf{nodes}(\mathbb{A})$ is $\sigma(\rho \downarrow i)$ and $t = t'$ is in $\mathsf{rconstr}(\rho, i)$, then $\sigma(t) = \sigma(t')$ is an equality constraint in $\mathcal{T}$.
- We replace every term of the form $\mathsf{gexp}(e)$ by g^{e}.
- For all α, $\mathsf{ARestr}(\alpha) = \emptyset$.

Although lightweight Π have very restricted message forms, defined over Σ_0, the role constraints are over Σ_1. Thus, the constraint systems we work with involve terms over Σ_1.

Lemma VI.4. *Let $\mathcal{T}$ be the constraint system for $(\mathbb{A}, \Pi)$. If θ is a q-solution to $\mathcal{T}$, and $(h_q \circ \theta)(\mathbb{A})$ satisfies the origination assumptions for Π, then $(h_q \circ \theta)(\mathbb{A})$ is a realized skeleton for Π. If $(h_q \circ \theta)(\mathbb{A})$ is a realized Π-skeleton over q-Alg then θ is a q-solution to $\mathcal{T}$.*

Proof: By the definitions of realized skeleton (Definition III.1) and q-solution to a system (Definition VI.2). ■

C. Transformations

The transformations reduce constraints systems, ultimately, to systems of linear algebraic equations. Pattern-matching on the "non-algebraic" structure of messages, such as pairing and signatures, leaves a residue of algebraic problems to be solved over the finite fields in each q-Alg.

Definition VI.5 (Transformations)**.** See Figure 11. ///

The rules *E-hashes* and *Span* act globally on a system $\mathcal{T}$. The other rules replace a single constraint by one or more derivability constraints or equations; we indicate only these local actions in the figure. The symbols $\top$ and $\bot$ represent trivial constraints, always true and always false, respectively. We use the comma for union and the semicolon for disjoint union, so that "$S,\ t$" means $S \cup \{t\}$, but "$S;\ t$" also asserts that t is not an element of S. It is understood that a transformation is applied only if it causes a change n the system (e.g. redundant *Pairs-left* transformations are not done).

The first group of rules—*Pairs-left*, *Pairs-right*, *Signatures-left*, *Signatures-right*, and *Basic*— simplify derivability constraints. The rules *Span*, *E-hashes* and *Select* generate equality constraints. *Decompose* simplifies equality constraints.

Pairs-left $S;\ (\mathsf{tag}\, m_1,\ m_2) \Vdash_N\ t \implies$

$S, (\mathsf{tag}\, m_1,\ m_2), m_1, m_2 \Vdash_N\ t$

Pairs-right $S \Vdash_N\ (\mathsf{tag}\, t_1,\ t_2) \implies S \Vdash_N\ t_1,\ S \Vdash_N\ t_2$

Signatures-left $S;\ \llbracket m \rrbracket_k \Vdash_N\ t \implies S,\ \llbracket m \rrbracket_k,\ m \Vdash_N\ t$

Signatures-right $S \Vdash_N\ \llbracket t \rrbracket_k \implies S \Vdash_N\ t \quad k \notin N$

Basic $S \Vdash_N\ t \implies \top \quad t \notin N$

$S \Vdash_N\ t \implies \bot \quad t \in N$

when $t \in \mathsf{Basic}$ and contains no A-params.

Select $S;\ m \Vdash_N\ t \implies m = t$

Decompose $g^{e_1} = g^{e_2} \implies e_1 = e_2$

$f(t_1, \ldots, t_n) = f(t'_1, \ldots, t'_n) \implies \ldots, t_i = t'_i,\ \ldots$

$f(t_1, \ldots, t_n) = g(t'_1, \ldots, t'_{n'}) \implies \bot$

when $f \neq g$ are among $\mathsf{sk}, \mathsf{Hash}$, $(\mathsf{tag}\ \cdot,\ \cdot)$, or $\llbracket \cdot \rrbracket$.

—-

E-hashes $\mathcal{T} \implies \mathcal{T}_v,\ \{v = \mathsf{hash}(t), t = t' \colon \mathsf{hash}(t), \mathsf{hash}(t') \in W\}$

when W is some subset of the set of E-hashes occurring in $\mathcal{T}$; where v is freshly-chosen E-param; and $\mathcal{T}_v$ is the result of replacing each occurrence of a term in W by v.

—-

Span is detailed in Definition VI.6

Fig. 11. Constraint-Solving Transformations

The *Signatures-right* rule reflects that if the signing key k is not in N, then one can derive $\llbracket t \rrbracket_k$ as soon as one can derive t. When the signing key may be in N, we may instead use *Select*. It guesses a member m of S and converts $S \Vdash \llbracket t \rrbracket_k$ into the equality constraint that $\llbracket t \rrbracket_k$ match m.

In the *E-hashes* rule, since the hashes are intended to be uniformly algebraically independent of every value (*c.f.* Assumption III.7) a solution θ must match hashes on the left with hashes on the right. However, the terms *inside* the hashes can be complex terms with algebraic relations among them. To allow for this possibility a transformation may replace any number of hashes with the same fresh parameter, adding constraints to record that these equalities must hold.

The *Span* transformation is the crux of our approach.

Definition VI.6 (*Span*)**.** Suppose

$$S_1;\ g^{d_1}, \ldots g^{d_n} \Vdash_N\ g^e$$

is saturated with respect to *Pairs-left*, *Signatures-left*, and *E-hashes* and satisfies $S_1 \cap G = \emptyset$. Furthermore, suppose $\mathcal{V}$ is the collection of $\mathsf{R\_params}$ occurring in $S_1;\ g^{d_1}, \ldots g^{d_n}$ that are associated with hash-abbreviations. Partition $\mathcal{V}$ as $\mathcal{V}_0 \cup \mathcal{V}_1$. Then *Span* transforms $\mathcal{T}$ as follows

1. Replace this constraint with the equation $e = \alpha_0 + \alpha_1 d_1 + \cdots + \alpha_n d_n$ where each α_i is a fresh $\mathsf{A\_param}$
2. For each hash abbreviation $v_i = \mathsf{hash}(t_i)$ associated with an $\mathsf{R\_param}$ in $\mathcal{V}_0$, we add the derivability constraint $S_1;\ g^{d_1}, \ldots g^{d_n} \Vdash_N\ t_i$.
3. For each j, $\mathsf{ARestr}_{\mathcal{T}}(\alpha_j) = N \cup \mathcal{V}_1$. ///

D. Analysis of the Transformations

Three useful invariants follow by induction. The "linear adversary contribution" clause (Defn. III.14) yields the last.

Lemma VI.7. *Let $\mathcal{T}$ be the constraint system for $(\mathbb{A}, \Pi)$, and suppose $\mathcal{T} \Rightarrow \cdots \Rightarrow \mathcal{T}'$. Then*

1. *If $S \Vdash_N\ t$ is in $\mathcal{T}'$ and $m \in S$, then $m \sqsubseteq \sigma(\mathsf{msg}(\rho \downarrow i))$ for some $\rho \in \Pi$ (possibly* blab*), i, and substitution σ.*
2. *For every constraint $S \Vdash_N\ t$ of $\mathcal{T}'$, $S \cap N = \emptyset$.*
3. *Each message in $\mathcal{T}'$ of sort E in linear in the* $\mathsf{A\_params}$*: no monomial has more than one A-parameter occurring.*

Theorem VI.8 (Irreducible Forms)**.** *If $\mathcal{T}$ is an irreducible system other than $\bot$ then each constraint of $\mathcal{T}$ is an equality constraint $e_1 = e_2$ where e_1 and e_2 are polynomials over* EParam.

Proof: By inspection we see that any derivability constraint with a non-G term as its subject can be reduced, and *Span* will eventually apply to the others. Any equation between terms of non-E sort will submit to *Decompose*. Any equation between E terms is of the form indicated; note that *E-hashes* ensures that terms are purely algebraic, that is, with no hash-terms. ■

Elementary techniques show that the system of transformations is terminating

Theorem VI.9 (Termination)**.** *Each sequence of transformations is finite.*

Theorem VI.10 (Soundness)**.** *Suppose $\mathcal{T} \implies \mathcal{T}'$. If θ is a solution to $\mathcal{T}'$ at q-*Alg*, then θ is a solution to $\mathcal{T}$ at q-*Alg.

Proof: We examine each transformation in turn, to check that the transformed system does not admit any new solutions. The two *Pairs* transformations are standard. The *Signatures-left* rule is sound since we assume that the adversary can access the payload of a digital signature. The *Signatures-right* rule reflects the fact that if the signing key is *not* secret then the adversary can construct $\llbracket t \rrbracket_k$ if he can construct t. The *Basic* rules simply reflect the role of the restricted values N. *Select* and *Decompose* are clearly sound.

For the *E-hashes* rule we first note that the new parameter v is not an adversary-controlled parameter, so $\theta(v) = v$. The hashes from W behave as indeterminates in any algebraic calculation or derivability construction: the only way they can interact is by cancellation. The fact for each of the new equations $t = t'$ associated with W we have $(h_q \circ)(t) = (h_q \circ)(t')$ means that θ was a solution to the original system.

As for *Span*, assuming that θ is a q-solution to $\mathcal{T}'$, we need only show that θ solves the eliminated derivability constraint, that is, that $\theta(S_1);\ g^{\theta d_1}, \ldots g^{\theta d_n}$ derives $g^{\theta e}$ avoiding N. But this is clear from $\theta e = \theta \alpha_o + \theta \alpha_1 d_1 \cdots + \theta \alpha_n d_n$ (cf. Lemma III.12). ■

Soundness says that transformations cannot introduce spurious solutions. Completeness says, intuitively, that they generate all solutions, but that is not true as naively stated: hashing can create "accidental" solutions at finitely many q. The *E-hashes* transformation spoil these (as it should!). The notion of completeness that our transformations enjoy is more subtle.

Theorem VI.11 (Progress). *Let $\Delta = \{\theta_q \colon q \in I\}$ be an infinite family of q-solutions for $\mathcal{T}$. If $\mathcal{T}$ is not irreducible, there exists a transformation $\mathcal{T} \Longrightarrow \mathcal{T}'$, an infinite subset $I' \subseteq I$, and a family $\Delta' = \{\theta'_q \colon q \in I'\}$ of q-solutions for $\mathcal{T}$' with $\theta'_q \restriction_{Vars(\mathcal{T})} = \theta_q \restriction_{Vars(\mathcal{T})}$ for each $q \in I'$.*

Proof: We first show that each transformation other than *E-hashes*, *Span*, *Signatures-right*, and *Select* enjoys the property that every solution of $\mathcal{T}$ is also a solution of the transformed result (so if any of them can be applied the result follows). This property is evident for the rules *Pairs-left*, *Pairs-right*, *Signatures-left*, and *Decompose*. The first *Basic* rule preserves solutions simply because it removes a constraint; for the $\perp$ rule we observe that by Lemma VI.7.2, if $t \in N$ then there are no solutions to $S \Vdash_N t$.

Next suppose $\mathcal{T}$ has some occurrences of E-hashes. Choose any term of the form $\mathsf{hash}(t)$ that occurs in $\mathcal{T}$. Say that $\theta_q \in \Delta$ "makes a collision" with $\mathsf{hash}(t)$ if there is at least one term $\mathsf{hash}(t')$ occurring in $\mathcal{T}$ such that $\theta(\mathsf{hash}(t)) = \theta(\mathsf{hash}(t'))$ but $\theta(t) \neq \theta(t')$. By Assumption III.7 and the fact that there are only finitely many possible $\mathsf{hash}(t')$, there can be only finitely many q at which this happens for t. So in fact we may assume without loss of generality that our family Δ of solutions never makes any collisions with this $\mathsf{hash}(t)$. Now for each q let $T_q = \{\mathsf{hash}(t_i) \colon 1 \leq i \leq n\}$ be the set of E-hashes in $\mathcal{T}$ such that $\theta_q(t_i) = \theta_q(t)$. There are only finitely many possible such T_q and so one of these sets T occurs infinitely often. In particular, there is an infinite family $\Delta' \subseteq \Delta$ of solutions that solve the result of applying *E-hashes* with this T.

So suppose that $\mathcal{T}$ is irreducible with respect to the rules other than *Select*, *Signatures-right*, and *Span*. Since no equation can be reduced, and there is a reducible $S \Vdash_N t$, and t must be either a digital signature or a G-term g^e.

In the former case, t is $[\![n]\!]_k$ and—depending on whether $\theta(k)$ is in N or not— either *Select* or *Signatures-left* will apply and preserve any given θ_q. The choice of transformation in each case above might not be uniform across the $\theta_q \in \Delta$. But because there are only two choices in each case, at least one of them will preserve infinitely many elements of Δ.

In the latter case, there is some derivability constraint

$$S_1;\ g^{d_1}, \ldots g^{d_n} \Vdash_N g^e$$

obeying the preconditions of the *Span* rule. Consider the collection $\mathcal{V}$ of $\mathsf{R\_params}$ occurring in $S_1;\ g^{d_1}, \ldots g^{d_n}$ that are associated with hash-abbreviations. Define $\mathcal{V}_0$ to be the set of those $v \in \mathcal{V}$ such that, letting $v = \mathsf{hash}(t)$ be the hash abbreviation associated with v, θt is derivable from $\theta(S_1);\ g^{\theta d_1}, \ldots g^{\theta d_n}$. Let $\mathcal{V}_1$ be the complement of $\mathcal{V}_0$ in $\mathcal{V}$. This determines the data for an application of *Span*.

We may apply Lemma III.12, because the remaining rules ensure that any $g^e \sqsubseteq t$ for $t \in S_1$ is one of the g^{d_i}. Thus, there are terms $f_0, f_1, \ldots, f_n$ such that $e = f_0 + f_1 d_1 + \cdots + f_n d_n$, where each f_i is derivable from the left-hand side avoiding $N \cup \mathcal{V}_1$. We may then take Δ' to be the family of substitutions obtained from Δ by adding to each $\delta \in \Delta$ the bindings mapping α_i to f_i. ■

Theorem VI.12. *Suppose that $\Delta = \{\theta_q \colon q \in I\}$ is an infinite family of q-solutions for $\mathcal{T}$. Then there exists a sequence of transformations out of $\mathcal{T}$ resulting in an irreducible $\mathcal{T}^*$, an infinite subset $I^* \subseteq I$, and a family $\Delta^* = \{\theta^*_q \colon q \in I^*\}$ of q-solutions for $\mathcal{T}^*$ with $\theta^*_q \restriction_{Vars(\mathcal{T})} = \theta_q \restriction_{Vars(\mathcal{T})}$ for each $q \in I^*$.*

Proof: By the Progress Theorem VI.11, the Termination Theorem VI.9, and König's Lemma (the transformation system is finitely-branching). ■

We have reduced the problem of satisfying systems of constraints—and hence, testing security goals—to solving systems where the derivability constraints have been eliminated, and each equality constraint is an equation between purely algebraic terms. We call such systems *purely algebraic.*

Definition VI.13. If $\delta \in \mathsf{A\_params}$, an equation of the form $\delta = \alpha_0 + \alpha_1 d_1 + \cdots + \alpha_n d_n$ is a *principal constraint* for δ.

Lemma VI.14. *Let $\mathcal{T}$ be the constraint system for $(\mathbb{A}, \Pi)$, and suppose $\mathcal{T} \Rightarrow \cdots \Rightarrow \mathcal{T}^*$ with $\mathcal{T}^*$ irreducible. If δ is an A-parameter of sort E in $\mathcal{T}$ then $\mathcal{T}^*$ contains a principal constraint for δ.*

Proof: Consider a $\preceq$-minimal constraint where δ appears in $\mathcal{T}$. This corresponds to a regular reception node n, which generates a derivability constraint $S \Vdash_N \mathsf{msg}(n)$ with δ occurring in $\mathsf{msg}(n)$. Since $\mathbb{A}$ is an FAlg skeleton, $\mathsf{msg}(n)$ is built from g^δ using pairing and digital signatures. An invariant of the sequence $\mathcal{T} \Rightarrow \cdots \Rightarrow \mathcal{T}^*$ is that there is always some derivability constraint whose right-hand side contains g^δ, and since $\mathcal{T}^*$ is irreducible we conclude that at some stage there was some derivability constraint $S \Vdash_N g^\delta$ whose right-hand side is precisely g^δ. Since $\mathcal{T}^*$ is irreducible with respect to *Span* the result follows. ■

Theorem VI.15. *Let $\mathcal{T}$ be the constraint system for $(\mathbb{A}, \Pi)$. If $\mathcal{T}$ is infinitely-often solvable then there exists a single substitution $\theta : \mathsf{A\_params} \to \mathsf{eFAlg}$ that is a q-solution for infinitely many q. Such an θ is effectively computable from $\mathcal{T}$.*

Proof: Apply the transformations systematically and exhaustively to $\mathcal{T}$; by Theorem VI.9 and the fact that the transformation system is finite-branching there is a finite set of irreducible systems reachable from $\mathcal{T}$. It suffices to test whether any of these are infinitely-often solvable. So consider a fixed such irreducible system $\mathcal{T}^*$. For rest of this proof we fix a q and describe how to determine solvability of $\mathcal{T}^*$ at q. But it will be clear that our reasoning is independent of q, except for the very last step, which will depend on q in a quite transparent way.

By Lemma VI.14, for each A-var δ there is at least one principal constraint for δ:

$$\delta = \alpha_0 + \alpha_1 d_1 + \ldots \alpha_n d_n$$

We can replace δ everywhere it occurs in $\mathcal{T}^*$ by $\alpha_0 + \alpha_1 d_1 + \ldots \alpha_n d_n$.

After normalizing each equation $e_1 = e_2$ to be of the form $e_1 - e_2 = 0$, we have transformed each constraint into an equation $p = 0$ where p is a term over $\mathbb{F}_q(\mathsf{EParam})$, and we want to solve for the α_i, These equations have the form of *linear* equations over the α_i with coefficients in the field $\mathbb{F}_q(\mathsf{EParam})$, the field of quotients of the ring $\mathbb{F}_q[\mathsf{EParam}]$

obtained by viewing EParam as a set of indeterminates adjoined to $\mathbb{F}_q$. The fact that the original equations are linear in the A_params was observed in Lemma VI.7.3, and we have replaced each $\delta \in$ A_params by a linear combination of the α_j.

A subtlety is that our notion of solution does not allow variables to range over all of $\mathbb{F}_q(\mathsf{EParam})$: any given variable α must avoid its restriction set $\mathsf{ARestr}(\alpha)$ (see Definition VI.1). So straightforward Gaussian Elimination is not suitable.

But any equation can always be organized as $\Sigma\ \alpha_i m_i = 0$ where α_i is a parameter to be solved for and m_i is a power product of elements of $\mathbb{F}_q(\mathsf{EParam})$. Now write such an m as $a_1 \ldots a_n b_1 \ldots b_m$ where the b_i are the elements of EParam in $\mathsf{ARestr}(\alpha)$. (If there is a non-1 element of $\mathbb{F}_q$ in the monomial, it will be one of the a_i.) Let us call $\alpha a_1 \ldots a_n$ the "flexible part" of the monomial and $b_1 \ldots b_m$ the "rigid part" of the monomial. Now, writing each equation in the system as a sum of such differentiated monomials, we can collect terms according to their rigid parts and set each of the combinations of the flexible parts to 0. For example if a is in the restriction set of both α and γ and β is in the restriction set of y then $\gamma a + \beta b - \alpha a = 0$ if and only if $(\gamma - \alpha)a + \beta b = 0$ if and only if $\alpha = \gamma$ and $\beta = 0$. The key to this construction is that each combination of flexible parts consists of a sum of monomials involving variables and R_params that are available to be in the range of an instantiation of those variables.

A given equation will generate a set of such linear equations that set each flexible part to 0. Each of these can be viewed as a system of linear equations, with coefficients guaranteed, by construction, to be in a field generated by adjoining to $\mathbb{F}_q$ some parameters that are available to each variable to be solved for. We then solve each of these systems, by Gaussian Elimination. Note that the process of Gaussian Elimination can be done uniformly over the various q-Alg: we can always reduce a system to row echelon form by computations that are generic across the different primes q. The fact that the coefficients of the equations lie in an extension field $\mathbb{F}_q(\mathsf{EParam})$ of the "concrete" finite field $\mathbb{F}_q$ is of course no obstacle to the computability of the operations required for Gaussian Elimination.

At the end of each Gaussian Elimination we arrive at the general row-echelon form of a system of equations. This may turn out to involve some equations of the form $a = 0$ where a is not identically the 0 term. We are working over finite fields, where such equations between formally non-0 terms are not necessarily failures. For instance, the equation $0 = 22$ is solvable in q-Alg as long as $q = 2$ or $q = 11$. Our system is solvable at q if and only if in each equation of the form $0 = a$, a is a constant divisible by q. It is precisely here that we can see that the "infinitely-often" notion of solution is the most natural one. The system is infinitely-often solvable if and only if there are no equations $0 = a$ with a not identically 0.

If we obtain a solution—in the linear algebra sense— to these equations then it is a solution satisfying the restriction requirement in Definition VI.1. We also need to establish that if Gaussian Elimination finds no solutions, then there are indeed no solutions to the original equation. That is, if there are no solutions to a system over the field determined by adjoining the R_params in the flexible part, then there are no solutions in any larger field (for example using adversary-derivable creative constructs involving the available parameters). This is a general fact about linear algebra: if a system of linear equations with coefficients in a field F has no solutions in F then it has no solutions in any extension F' of F.

■

If—and only if—there is an infinite set of counterexamples, then for some FAlg skeleton $\mathbb{A}$ the above process will construct an eFAlg skeleton $\theta(\mathbb{A})$ such that for infinitely many q, $h_q(\theta(\mathbb{A})) \not\models \Theta$. We have seen how to compute an exhaustive search for such candidate $\theta(\mathbb{A})$. The final step in the decidability procedure is to recognize whether an eFAlg skeleton we have constructed does indeed falsify the original security goal Θ at infinitely many q. That is, we want to check whether, when our eFAlg skeleton is interpreted in q-Alg via the various h_q maps, Θ holds. This would be impossible if our goal language were able to make interesting algebraic assertions. But for assertions in the goal language there is an easy test, as described in the following lemma.

Recall that $h_{\mathbb{Q}}$ is the canonical homomorphism mapping an eFAlg skeleton to a skeleton over the rational numbers.

Lemma VI.16. *Suppose $\mathbb{C}$ is a skeleton over* eFAlg *and Θ is a sentence of the goal language. Then $h_q(\mathbb{C})$ falsifies Θ for infinitely many q if and only if $h_{\mathbb{Q}}(\mathbb{C})$ falsifies Θ.*

Proof: The goal language does not speak of the structure of messages and the maps h_q and $h_{\mathbb{Q}}$ are the identity outside of E, so the only way the h_q maps affect the interpretation of goal-language formulas is through equality. So it suffices to make the following easy observation that for E terms e_1 and e_2: infinitely many h_q satisfy $h_q(e_1) = h_q(e_2)$ if and only if $h_{\mathbb{Q}}(e_1) = h_{\mathbb{Q}}(e_2)$. ■

We can now assert our main result, the decidability of security goals for lightweight Diffie-Hellman protocols.

Theorem VI.17. *Algorithm DH-Decide satisfies the following specification, for input π and goal Θ:* If $\mathcal{A}$ is empty then Π satisfies Θ. Otherwise, for each $\mathbb{A} \in \mathcal{A}$ and for infinitely many q, the image of $\mathbb{A}$ under h_q falsifies Θ.

```
Input: A lightweight protocol $\Pi$ and a security goal
       $\Theta \equiv \Phi \supset \Psi$
Output: A list $\mathcal{A}$ of realized skeletons for $\Pi$ over $\Sigma_0$
/* $\Pi$ satisfies $\Theta$ almost everywhere iff $\mathcal{A}$ is empty */
let $b$ be the bound for $\Phi \supset \Psi$ per Corollary V.5 ;
foreach FAlg $\Pi$-skeleton $\mathbb{A} \models \Phi$ with no more than $b$
  strands do
    let $\mathfrak{I}$ be the constraint system for $(\mathbb{A}, \Pi)$ ;
    foreach reduction of $\mathfrak{I}$ to an irreducible system $\mathfrak{I}^*$
    and for each solution $\theta$ to $\mathfrak{I}^*$ do
        if $\theta(\mathbb{A})$ satisfies the role-origination
        assumptions of $\Pi$ and falsifies $\Psi$ with $\vec{a}$ then
            add $\theta(\mathbb{A})$ to the output list
```

Algorithm 1: DH-Decide

VII. SOME EXAMPLES

Two Analyses Concerning MQV We present two analyses concerning MQV. As remarked in Section II it is sensible to treat the operator $[\cdot]$ as a hash function.

Key Secrecy for MQV

The MQV protocol was defined in Section II. To determine whether key secrecy is guaranteed, let us assume A completes a run and that the A receives the correct certificate value for B, in other words, that $\beta = b$. Let us further assume that the long-term and ephemeral private keys are uncompromised: non= $\{a, b, x, y\}$

Among the FAlg skeletons constructed by the process in Section V is one with (i) one initiator strand for A, with long-term private key a, sending g^x, receiving message g^δ, and receiving a certificate with component g^b, and computing the key g^κ, with a role constraint setting $g^\kappa = (g^\beta g^\delta)^{(a+x)}$ and (ii) a listener strand, receiving g^κ

The resulting constraint system has

- the derivability constraints $\{g^x, g^a, g^\beta,\} \Vdash g^\kappa$ and $\{g^x, g^a, g^\beta,\} \Vdash g^\delta$
- the role constraint capturing the key computation
  $$g^\kappa = g^{\delta+(b[g^\delta])(x+a[g^x])} = g^{\delta x+\delta a[g^x]+xb[g^\delta]+ab[g^\delta][g^x]}$$
- the equation $\beta = b$ capturing the assumption that A does receive the correct certificate for B.

This system happens to already be irreducible in the sense of the first part of Section VI. So we proceed to the algebraic process described in Theorem VI.15.

We are led to

$$\kappa = \kappa_0 + \kappa_1 x + \kappa_2 a + \kappa_3 b = \delta x + \delta a[g^x] + xb[g^\delta] + ab[g^\delta][g^x]$$
$$\delta = \delta_0 + \delta_1 x + \delta_2 a + \delta_3 b$$

Next we use the *E-hashes* rule to introduce new indeterminates to stand in for the $[\cdot]$ expressions. There are two cases: we introduce one R_param c to replace all such expressions, or we use two new R_params c_1 and c_2. In the former case we also add another constraint to the system, namely that $g^x = g^\delta$, which implies $x = \delta$.

Case 1: we have

$$\begin{aligned} \kappa &= \kappa_0 + \kappa_1 x + \kappa_2 a + \kappa_3 b \qquad &= \delta x + \delta ac + xbc + abc^2 \\ \delta &= \delta_0 + \delta_1 x + \delta_2 a + \delta_3 b \end{aligned}$$

This has no solutions: no values for the κ_i can cancel out the monomials with non-elements.

Case 2 is similar.

Unknown Key-Share for MQV

Here we show how the "unknown key-share" attack on MQV defined by Kaliski [24] would be discovered by our algorithm.

We want to ask whether the following goal is satisfied: if A and B compute the same key, is each the other's expected peer? Taking the perspective of B, we want to ask whether, if B executes a run, then is the principal with which B shares the key the same principal whose certificate B used in computing the key? Since key secrecy has been established, it is sensible to refer to "the" principal with whom the key is shared.

This goal was expressed formally in Equation 2 in Section IV. In the same manner as for key secrecy we derive, for each bounded FAlg skeleton representing the hypothesis of the goal, a set of equations; we test the goal by asking whether there is a solution to these equations which does *not* satisfy $\alpha = a$. The UKS attack will be available if one principal can arrange to have a key certified that incorporates information from the message sent by the other principal: this will be reflected in our model by a skeleton with the property that a message containing g^γ comes before that for g^α.

After making the obvious instantiations by the role constraints we arrive at the following equations.

$$\begin{aligned} \kappa &= \delta x + \delta a[g^x] + xb[g^\delta] + a\beta[g^\delta][g^x] \\ &= \gamma y + \gamma b[g^y] + y\alpha[g^\gamma] + \alpha b[g^\gamma][g^y] \end{aligned}$$

As before we introduce R_params to stand in for the $[\cdot]$ terms. One of the possible choices that will be explored is that in which $[g^\delta]$ and $[g^y]$ are set equal, with the consequence that δ and y are set equal: this one will lead to the attack. Using c to name these terms and d and e for the others we have

$$\kappa = yx + yae + xbc + abce$$
$$\kappa = \gamma y + \gamma bc + y\alpha d + \alpha bdc$$

which is equivalent to the equation $x + ae = \gamma + \alpha d$.

Introducing principal constraints $\alpha = \delta_0 + \delta_1 x + \delta_2 a + \delta_3 b$ and $\gamma = \kappa_0 + \kappa_1 x + \kappa_2 a + \kappa_3 b$ yields

$$\begin{aligned} &\kappa_0 + \kappa_1 x + \kappa_2 a + \kappa_3 b + \delta_0 d + \delta_1 xd \\ &\qquad + \delta_2 ad + \delta_3 bd - x - ae = 0 \end{aligned}$$

Since d, corresponding to the $[g^\gamma]$ is emitted before the reception of g^α, the value d is not in the restriction set of the δ_i. A solution is readily computed:

$\delta_0 = d^{-1} \quad \kappa_0 = -1 \quad \kappa_1 = 1 \quad \kappa_2 = e$ all other variables = 0.

This is precisely the Kaliski attack.

Conclusion and Acknowledgments. The decidability of a rich class of security goals for the lightweight DH protocols validates the power of our method, with its algebraic faithfulness and strong adversary. Unfortunately, this result does not yield a practical procedure, since it relies on an exhaustive enumeration of exponentially many possible counterexamples. An important avenue for future work is to use this model to justify an "enrich-by-need" analysis method [22].

We are grateful to the anonymous referees, and also to Véronique Cortier, Steve Kremer, Moses Liskov, Dusko Pavlovic, John Ramsdell, and Paul Rowe.

REFERENCES

[1] R. Ankney, D. Johnson, and M. Matyas. The Unified Model. contribution to ANSI X9F1. *Standards Projects (Financial Crypto Tools), ANSI X*, 42, 1995.

[2] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In Kousha Etessami and Sriram K. Rajamani, editors, *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285. Springer, 2005.

[3] David A. Basin, Sebastian Mödersheim, and Luca Viganò. OFMC: A symbolic model checker for security protocols. *Int. J. Inf. Sec.*, 4(3):181–208, 2005.

[4] Simon Blake-Wilson and Alfred Menezes. Authenticated Diffe-Hellman key agreement protocols. In *Selected Areas in Cryptography*, pages 630–630. Springer, 1999.

[5] Bruno Blanchet. An efficient protocol verifier based on Prolog rules. In *14th Computer Security Foundations Workshop*, pages 82–96. IEEE CS Press, June 2001.

[6] Colin Boyd and Anish Mathuria. *Protocols for authentication and key establishment*. Springer, 2003.

[7] Emmanuel Bresson, Yassine Lakhnech, Laurent Mazaré, and Bogdan Warinschi. Computational soundness: The case of Diffie-Hellman keys. In Veronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, Cryptology and Information Security Series. IOS Press, 2011.

[8] Yannick Chevalier, Ralf Küsters, Michaël Rusinowitch, and Mathieu Turuani. Complexity results for security protocols with Diffie-Hellman exponentiation and commuting public key encryption. *ACM Transactions on Computational Logic (TOCL)*, 9(4):24, 2008.

[9] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

[10] Hubert Comon-Lundh, Véronique Cortier, and Eugen Zălinescu. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Transactions on Computational Logic (TOCL)*, 11(2):9, 2010.

[11] Véronique Cortier, Stéphanie Delaune, and Pascal Lafourcade. A survey of algebraic properties used in cryptographic protocols. *Journal of Computer Security*, 14(1):1–43, 2006.

[12] Véronique Cortier, Daniel J. Dougherty, Joshua D. Guttman, and Steve Kremer. The real-or-random property for lightweight Diffie-Hellman protocols. Meetings, January 2014.

[13] Cas Cremers and Michele Feltz. One-round strongly secure key exchange with perfect forward secrecy and deniability. Cryptology ePrint Archive, Report 2011/300, 2011. http://eprint.iacr.org/2011/300.

[14] C.J.F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006.

[15] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[16] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.

[17] Daniel J. Dougherty and Joshua D. Guttman. An algebra for symbolic Diffie-Hellman protocol analysis. In C. Palamidessi and M. Ryan, editors, *Trustworthy Global Computing*, LNCS. Springer, 2012. Post-proceedings to appear.

[18] Santiago Escobar, Catherine Meadows, and José Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. *Foundations of Security Analysis and Design V*, pages 1–50, 2009.

[19] Joshua D. Guttman. Security goals and protocol transformations. In Sebastian Mödersheim and Catuscia Palamidessi, editors, *Tosca: Theory of Security and Applications*, LNCS. Springer, March 2011.

[20] Joshua D. Guttman. Shapes: Surveying crypto protocol runs. In Veronique Cortier and Steve Kremer, editors, *Formal Models and Techniques for Analyzing Security Protocols*, Cryptology and Information Security Series. IOS Press, 2011.

[21] Joshua D. Guttman. State and progress in strand spaces: Proving fair exchange. *Journal of Automated Reasoning*, 48(2):159–195, 2012.

[22] Joshua D. Guttman. Establishing and preserving protocol security goals. *Journal of Computer Security*, 2014.

[23] Joshua D. Guttman and F. Javier Thayer. Authentication tests and the structure of bundles. *Theoretical Computer Science*, 283(2):333–380, June 2002.

[24] Burton S. Kaliski. An unknown key-share attack on the MQV key agreement protocol. *ACM Transactions on Information and System Security*, 4(3):275–288, 2001.

[25] Deepak Kapur, Paliath Narendran, and Lida Wang. An E-unification algorithm for analyzing protocols that use modular exponentiation. *Rewriting Techniques and Applications*, 2003.

[26] H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In *Advances in Cryptology–CRYPTO 2005*, pages 546–566. Springer, 2005.

[27] Steve Kremer and Laurent Mazaré. Computationally sound analysis of protocols using bilinear pairings. *Journal of Computer Security*, 18(6):999–1033, 2010.

[28] Sebastian Kunz-Jacques and David Pointcheval. About the Security of MTI/C0 and MQV. *Security and Cryptography for Networks*, pages 156–172, 2006.

[29] Ralf Küsters and Tomasz Truderung. Using ProVerif to analyze protocols with Diffie-Hellman exponentiation. In *IEEE Computer Security Foundations Symposium*, pages 157–171. IEEE, 2009.

[30] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. *Provable Security*, 4784, 2007. DOI: dx.doi.org/10.1007/978-3-540-75670-5_1.

[31] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003.

[32] Moses D. Liskov and F. Javier Thayer. Formal modeling of Diffie-Hellman derivability for exploratory automated analysis. Technical report, MITRE, June 2013. TR 13-0411.

[33] Yuri V Matiyasevich. Enumerable sets are diophantine. *Doklady Akademii Nauk SSSR*, 191(2):279–282, 1970.

[34] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distribution systems. *Transactions of the IECE of Japan*, E69:99–106, 1986.

[35] Alfred Menezes. Another look at HMQV. *Journal of Mathematical Cryptology*, 1:47–64, 2007.

[36] Jonathan Millen and Vitaly Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 166–175. ACM, 2001.

[37] Jonathan K. Millen and Vitaly Shmatikov. Symbolic protocol analysis with an abelian group operator or diffie-hellman exponentiation. *Journal of Computer Security*, 13(3):515–564, 2005.

[38] D. Pavlovic and C. Meadows. Deriving secrecy in key establishment protocols. *Computer Security–ESORICS 2006*, pages 384–403, 2006.

[39] Olivier Pereira and Jean-Jacques Quisquater. On the impossibility of building secure cliques-type authenticated group key agreement protocols. *Journal of Computer Security*, 14(2):197–246, 2006.

[40] R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–166, 2005. Preliminary version appeared in WITS ’03, *Workshop on Issues in the Theory of Security*, Warsaw, April 2003.

[41] John D. Ramsdell and Joshua D. Guttman. CPSA: A cryptographic protocol shapes analyzer, 2009. http://hackage.haskell.org/package/cpsa.

[42] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Computer Security Foundations Workshop*, pages 174–, 2001.

[43] Benedikt Schmidt, Simon Meier, Cas J. F. Cremers, and David A. Basin. Automated analysis of Diffie-Hellman protocols and advanced security properties. In *IEEE Symposium on Computer Security Foundations*, pages 78–94, 2012.