Enrich-by-need Protocol Analysis for Diffie-Hellman

Moses D. Liskov, Joshua D. Guttman, John D. Ramsdell, Paul D. Rowe, and F. Javier Thayer[⋆]

The MITRE Corporation

Abstract. Enrich-by-need analysis characterizes all executions of a security protocol that extend a given scenario. It computes a strongest security goal the protocol achieves in that scenario. CPSA, a Cryptographic Protocol Shapes Analyzer, implements enrich-by-need analysis.
In this paper, we show how CPSA now analyzes protocols with Diffie-Hellman key agreement (DH) in the enrich-by-need style. While this required substantial changes both to the CPSA implementation and its theory, the new version retains CPSA's efficient and informative behavior. Moreover, the new functionality is justified by an algebraically natural model of the groups and fields which DH manipulates.
The model entails two lemmas that describe the conditions under which the adversary can deliver DH values to protocol participants. These lemmas determined how CPSA handles the new cases. The lemmas may also be of use in other approaches.
This paper is dedicated to Cathy Meadows, with warmth and gratitude.

1 Introduction

Diffie-Hellman key agreement (DH) [8], while widely used, has been challenging for mechanized security protocol analysis. Some techniques, e.g. [12,7,22,15], have produced informative results, but focus only on proving or disproving individual protocol security goals. A protocol and a specific protocol goal are given as inputs. If the tool terminates, it either proves that this goal is achieved, or else provides a counterexample. However, constructing the *right* security goals for a protocol requires a high level of expertise.

By contrast, the *enrich-by-need* approach starts from a protocol and some scenario of interest. For instance, if the initiator has had a local session, with a peer whose long-term secret is uncompromised, what other local sessions have occurred? What session parameters must they agree on? Must they have happened recently, or could they be stale?

Enrich-by-need protocol analysis identifies all essentially different smallest executions compatible with the scenario of interest. While there are infinitely many possible executions—since we put no bound on the number of local sessions—often surprisingly few of them are really different. The Cryptographic Protocol Shapes Analyzer (CPSA) [24], a symbolic protocol analysis tool based on strand

⋆ Email: `mliskov, guttman, ramsdell, prowe@mitre.org`

spaces [27,13,18], efficiently enumerates these minimal, essentially different executions. We call the minimal, essentially different executions the protocol's *shapes* for the given scenario.

Knowing the shapes tells us a *strongest* relevant security goal, i.e. a formula that expresses authentication and confidentiality trace properties, and is at least as strong as any one the protocol achieves in that scenario [14,23]. Using these shapes, one can resolve specific protocol goals. The hypothesis of a proposed security goal tells us what scenario to consider, after which we can simply check the conclusion in each resulting shape (see [25] for a precise treatment).

Moreover, enrich-by-need has key advantages. Because it can also compute strongest goal formulas directly for different protocols, it allows comparing the strength of different protocols [25], for instance during standardization. Moreover, the shapes provide the designer with *visualizations* of exactly what the protocol may do in the presence of an adversary. Thus, they make protocol analysis more widely accessible, being informative even for those whose expertise is not mechanized protocol analysis.

In this paper, we show how we strengthened CPSA's enrich-by-need analysis to handle DH. We will not focus on the underlying theory, which is presented at length in a report [17]. Instead, we will focus here on how CPSA uses that theory. It is efficient, and has a flexible adversary model with corruptions.

Foundational issues needed to be resolved. Finding solutions to equations in the natural underlying theories is undecidable in general, so mechanized techniques must be carefully circumscribed. Moreover, these theories, which include fields, are different from many others in security protocol analysis. The field axioms are not (conditional) equations, meaning that they do not have a simplest (or "initial") model for the analysis to work within. Much work on mechanized protocol analysis, even for DH relies on equational theories and their initial models, e.g. [12,7,22,15]. However, an analysis method should have an explicit theory justifying it from standard mathematical structures such as fields. CPSA now has a transparent foundation in the algebraic properties of the fields that DH manipulates. This extends our earlier work [11,16].

Contributions. In this paper, we describe how we extended CPSA to analyze DH protocols. CPSA is currently restricted to the large class of protocols that do not use addition in the exponents, which we call *multiplicative* protocols. CPSA is also restricted to protocols that disclose randomly chosen exponents one-by-one. This allows modeling a wide range of possible types of corruption; however, it excludes a few protocols in which products of exponents are disclosed. A protocol *separates disclosures* if it satisfies our condition.

These restrictions justify two principles—Lemmas 1 and 2—that are valid for all executions of protocols that separate disclosures. The lemmas characterize how the adversary can obtain an exponent value such as xy or an exponentiated DH value such as g^{xy}, respectively. They tell CPSA what information to add to enrich a partial analysis to describe the possible executions in which the adversary obtains these values. Lemmas 1 and 2 are a distinctive contribution that may also prove useful to other analysis tools, helping to narrow their search.

To formulate and prove these lemmas, we had to clarify the algebraic structures we work with. The messages in our protocol executions contain mathematical objects such as elements of fields and cyclic groups. We regard the the random choices of the compliant protocol participants as "transcendentals," i.e. primitive elements added to a base field that have no algebraic relationship to members of the base field.

CPSA yields results within a language of first order logic. The analysis delivers truths about all protocol executions; the executions are the *models* of the *theories* used in the analysis. This provides a familiar foundational setting. A long version [17] has a through discussion. Guttman [14] developed the underlying ideas, and Rowe et al. [25] applied them to "measure" the strength of protocols using the goal formulas the protocols achieve. That formal machinery is independent of whether the protocols use DH or not.

CPSA is very efficient. When executed on a rich set of variants of Internet Key Exchange (IKE) versions 1 and 2, our analysis required less than 30 seconds on a laptop. Section 7 tabulates results.

To Cathy, with gratitude. As so often, Cathy Meadows explored this area long before us. Maude-NPA has delivered informative results about Diffie-Hellman protocols for a long time, and her work jointly with Dusko Pavlovic a dozen years ago identified core protocol characteristics of Diffie-Hellman in a compact and informative way. We were certainly aware of an echo of their *guard* notion in our Lemmas 1–2. Perhaps the analysis we give in Section 4 should be regarded as an explanation of why the *guard* idea is properly applicable here.

2 An example protocol: Unified Model

In many Diffie-Hellman protocols [8], the participants A, B exchange both:

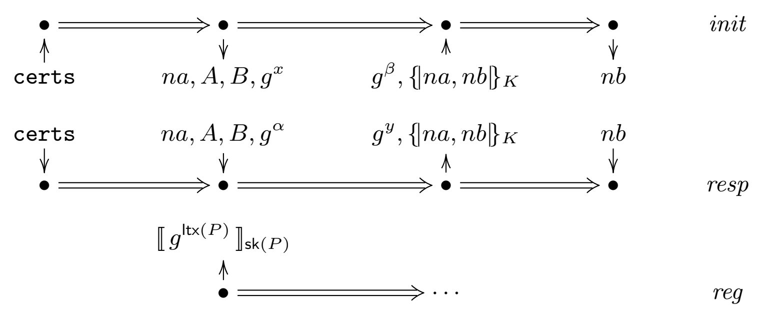
- certified, long-term values g^a and g^b, and also
- one-time, ephemeral values g^x and g^y.

The peers compute session keys using *key computation functions* $\mathsf{KCF}^{(\cdot)}$, where in successful sessions, $\mathsf{KCF}^{\mathsf{A}}(a, x, g^b, g^y) = \mathsf{KCF}^{\mathsf{B}}(b, y, g^a, g^x)$. The ephemeral values ensure the two parties agree on a different key in each session. The long-term values are intended for authentication: Any party that obtains the session key must be one of the intended peers. Different functions $\mathsf{KCF}^{(\cdot)}$ yield different security properties. Protocol analysis tools for DH key exchanges must be able to use algebraic properties to identify these security consequences.

We consider here a simple DH Challenge-Response protocol DHCR in which nonces from each party form a challenge and response protected by a shared, derived DH key (see Fig. 1).

A and B's long-term DH exponents are values $\mathsf{ltx}(A), \mathsf{ltx}(B)$, which we will mainly write as a lower case a, b resp. We assume the principals are in bijection with distinct long-term $\mathsf{ltx}(\cdot)$ values.

The *initiator* and *responder* each receive two self-signed certificates, certifying the long term public values $g^{\mathsf{ltx}(\cdot)}$ of his peer and himself.



init
certs
na, A, B, g^x
$g^\beta, \{\!|na, nb|\!\}_K$
nb
certs
na, A, B, g^α
$g^y, \{\!|na, nb|\!\}_K$
nb
resp
$[\![\, g^{\mathsf{ltx}(P)} \,]\!]_{\mathsf{sk}(P)}$
$\cdots$
reg

where `init` computes K as $\mathsf{KCF}^{\mathsf{A}}(a, x, g^b, g^\beta)$
`resp` computes K as $\mathsf{KCF}^{\mathsf{B}}(b, y, g^a, g^\alpha)$

Fig. 1. Protocol DHCR: Initiator, responder, and $\mathsf{ltx}()$ self-certifying roles. The `certs` are $[\![\, g^{\mathsf{ltx}(A)} \,]\!]_{\mathsf{sk}(P)}$ and $[\![\, g^{\mathsf{ltx}(B)} \,]\!]_{\mathsf{sk}(P)}$ for each role.

Each will send an ephemeral public DH value g^x, g^y in cleartext, and also a nonce. Each will receive a value which may be the peer's ephemeral, or may instead be some value selected by an active adversary. Neither participant can determine the exponent for the ephemeral value he receives, but since the value is a group element, it must be some value of the form g^α or g^β.

Each participant computes a session key using his $\mathsf{KCF}^{(\cdot)}$. The responder uses the key to encrypt the nonce received together with his own nonce. The initiator uses the key to decrypt this package, and to retransmit the responder's nonce in plaintext as a confirmation.

The *registration* role allows any principal P to emit its long-term public group value $g^{\mathsf{ltx}(P)}$ under its own digital signature. In full-scale protocols, a certifying authority's signature would be used, but in this paper we omit the CA so as not to distract from the core DH issues.

We will assume that each instance of a role chooses its values for certain parameters freshly. For instance, each instance of the registration role makes a fresh choice of $\mathsf{ltx}(P)$. Each instance of the initiator role chooses x and na freshly, and each responder instance chooses y and nb freshly.

DHCR is parameterized by the key computations. The Unified Model [1] offers three key computations, with a hash function $\#(\cdot)$ standing for key derivation. The shared keys—when each participant receives the ephemeral value $g^\alpha = g^x$ or $g^\beta = g^y$ that the peer sent—are:

Plain UM:	$\mathsf{KCF}^{\mathsf{A}}(a, x, g^b, g^\beta) = \#((g^b)^a, (g^\beta)^x)$
Criss-cross UMX:	$\#(g^{ay}, g^{bx})$
Three-component UM3:	$\#(g^{ay}, g^{bx}, g^{xy})$

We discuss three security properties:

Authentication: If either the initiator or responder completes the protocol, and *both* principals' private long-term exponents are secret, then the intended peer must have participated in a matching conversation.

The key computations UM, UMX, and UM3 all enforce the authentication goal.

Impersonation resistance: If either the initiator or responder completes the protocol, and the intended peer's private long-term exponent is secret, then the intended peer must have participated in a matching conversation.

Here we do not assume that *ones own* private long-term exponent is secret. Can the adversary impersonate the intended peer if ones own key is compromised?

DHCR with the plain UM KCF is susceptible to an impersonation attack: An attacker who knows Alice's own long-term exponent can impersonate any partner to Alice. The adversary can calculate g^{ab} from a and g^b, and can calculate the g^{xy} value from g^x and a y it chooses itself.

On the other hand, the UMX and UM3 KCFs resist the impersonation attack.

Forward secrecy: If the intended peers complete a protocol session and then the private, long-term exponent of each party is exposed *subsequently*, then the adversary still cannot derive the session key.

This is sometimes called *weak* forward secrecy.

To express forward secrecy, we allow the registration role to continue and subsequently disclose the long term secret $\mathsf{ltx}(P)$ as in Fig. 2. If we assume in an analysis that $\mathsf{ltx}(P)$ is uncompromised, that implies that this role does not complete. The dummy second node allows specifying that the $\mathsf{ltx}(P)$ release node occurs after some other event, generally the completion of a normal session.

The UM KCF guarantees forward secrecy, but UMX does not. In UMX, if an adversary records g^x and g^y during the protocol and learns a and b later, it can compute the key by exponentiating g^x to the power b and g^y to the power a. UM3 restores forward secrecy, meeting all three of these goals.

$$\begin{tikzcd}
\bullet \arrow[r, Rightarrow] \arrow[d] & \bullet \arrow[r, Rightarrow] & \bullet \arrow[d] \\
[\![g^{\mathsf{ltx}(P)}]\!]_{\mathsf{sk}(P)} & \texttt{dummy} \arrow[u] & \mathsf{ltx}(P)
\end{tikzcd}$$

Fig. 2. The full registration role

2.1 Strand terminology

A sequence of transmission and reception events as illustrated in Figs. 1–2 is a *strand.* Each send-event or receive-event on it is a *node.* We draw strands either horizontally as in Figs. 1–2 or vertically, as in diagrams generated by CPSA itself. We write $\mathsf{msg}(n)$ for the message sent or received by node n.

A *protocol* consists of a finite set of these strands, which we call the *roles* of the protocol. The roles contain variables, called the *parameters* of the roles, and by plugging in values for the parameters, we obtain a set of strands called the *instances* of the roles. We also call a strand a *regular strand* when it is an instance of a role, because it then complies with the rules. *Regular nodes* are the nodes that lie on regular strands. We speak of a *regular principal* associated with a secret if that secret is used only in accordance with the protocol, i.e. only in regular strands.

In an execution, events are (at least) partially ordered, and values sent on earlier transmissions are available to the adversary, who would like to provide the messages expected by the regular participants on later transmission nodes. The adversary can also generate primitive values on his own. We will make assumptions restricting which values the adversary does generate to express various scenarios and security goals.

3 How CPSA works: UMX initiator

Here we will illustrate the main steps that CPSA takes when analyzing DHCR. For this illustration, we will focus on the impersonation resistance of the UMX key computation, in the case where the initiator role runs, aiming to ensure that the responder has also taken at least the first three steps of a matching run. The last step of the responder is a reception, so the initiator can never infer that it has occurred. We choose this case because it is typical, yet quite compact.

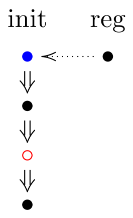


Fig. 3. Initial scenario, skeleton 0: $\mathsf{ltx}(B)$ non-compromised and $A \neq B$

Starting point. We start CPSA on the problem shown in Fig. 3, in which A, playing the initiator role, has made a full local run of the protocol, and received the long term public value of B from a genuine run of the registration role. These are shown as the vertical column on the left and the single transmission node at the top to its right. We will assume that B's private value $\mathsf{ltx}(B)$ is non-compromised and freshly generated, so that the public value $g^{\mathsf{ltx}(B)}$ originates only at this point. In particular, this run definitely does not progress to expose the secret as in the third node of Fig. 2. The fresh selection of $\mathsf{ltx}(B)$ must certainly precede the reception of $g^{\mathsf{ltx}(B)}$ at the beginning of the initiator's run. This is the meaning of the dashed arrow between them. We *do not* assume that A's long term secret $\mathsf{ltx}(A)$ is uncompromised, although we will assume that the ephemeral x value is freshly generated by the initiator run, and not available to the adversary. We assume $A \neq B$, which is the case of most interest.

Exploration tree. Fig. 4 shows the exploration tree that CPSA generates. Each item in the tree—we will call each item a *skeleton*—is a scenario describing some behavior of the regular protocol participants, as well as some assumptions. For instance, skeleton 0 contains the assumptions about $\mathsf{ltx}(B)$ and A's ephemeral value x mentioned before. The exploration tree contains one blue, bold face entry, skeleton 1 (shown in Fig. 5), as well as a subtree starting from 2 that is all red. The bold blue skeleton 1 is a *shape*, meaning it describes a simplest possible execution that satisfies the starting

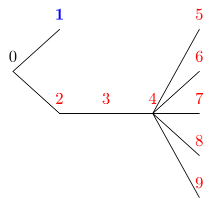


Fig. 4. CPSA exploration tree

skeleton 0. The red skeletons are *dead skeletons*, meaning possibilities that the search has excluded; no executions can occur that satisfy these skeletons. Thus, skeleton 1 is the only shape, and CPSA has concluded that all executions that satisfy skeleton 0 in fact also satisfy skeleton 1.

In other examples, there may be several shapes identified by the analysis, or in fact zero shapes. The latter means that the initial scenario cannot occur in any execution. This may be the desired outcome, for instance when the initial scenario exhibits some disclosure that the protocol designer would like to ensure is prevented.

First step. CPSA, starting with skeleton 0 in Fig. 3, identifies the third node of the initiator strand, which is shown in red, as unexplained. This is the initiator receiving the DH ephemeral public value g^y and the encryption $\{\!|na, nb|\!\}_K$, where K is the session key A computes using g^y and the other parameters. The node is red because the adversary cannot supply this message on his own, given the materials we already know that the regular, compliant principals have transmitted. Thus, CPSA is looking for additional information, including other transmissions of regular participants, that could explain it. Two possibilities are relevant here, and they lead to skeletons 1 and 2 (see Fig. 5–6).

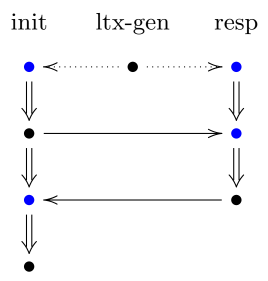


Fig. 5. Skeleton 1, the sole resulting shape.

In skeleton 1, a regular protocol participant executing the responder role transmits the message $g^y, \{\!|na, nb|\!\}_K$. Given the values in this message—including those used to compute K using the UMX function—all of the parameters in the responder role are determined.

This is an encouraging result: The initiator has interacted with a run of the responder role. Moreover, the solid arrows indicate that the responder has received exactly what the initiator has transmitted, and *vice versa*. CPSA's accompanying text output confirms that the two strands agree on all of their parameters. Thus, they agree about their identities, the names A, B; the long-term exponents $\mathsf{ltx}(A), \mathsf{ltx}(B)$; the ephemeral values x, y, and the nonces na, nb. Thus, the initiator has authenticated the responder with an exactly matching run [5,20].

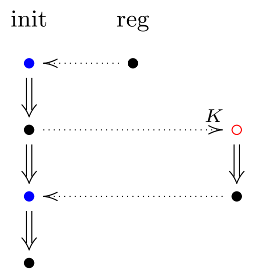


Fig. 6. Skeleton 2: Can the UMX encryption key K be exposed, on the rightmost strand?

Skeleton 2 considers whether the key $K = \#(g^{ay}, g^{bx})$, computed by the initiator, might be compromised. K is the value received on the rightmost strand. The reception node is called a *listener node*, because it witnesses for the availability of K to the adversary. The "heard" value K is then retransmitted so that CPSA can register that this event must occur before the initiator's third node. This listener node is red because CPSA cannot yet explain how K would become available. However, if additional information, such as more actions of the regular participants, would explain it, then the adversary could use K to encrypt na, nb and forge the value A receives. Thus, skeleton 2 identifies this listener node for further exploration.

Step 2. Proceeding from skeleton 2, CPSA performs a simplification on $K = \#(g^{ay}, g^{bx})$. The value y is available to the adversary, as is a, since we have not assumed them uncompromised. Thus, g^{ay} is available. The adversary will be able to compute K if he can obtain g^{bx}. Skeleton 3 (not shown) is similar to skeleton 2 but has a red node asking CPSA to explain how to obtain g^{bx}.

This requires a step which is distinctive to DH protocols. CPSA adds Skeleton 4 (Fig. 7), which has a new rightmost strand with a red node, receiving the pair $g^{bx/w}, w$. To resolve this, CPSA must meet two constraints. First, it must choose an exponent w that can be exposed and available to the adversary. Second, for this value of w, either $w = bx$ or else the "leftover" DH value $g^{bx/w}$ must be transmitted by a regular participant and extracted by the adversary.

One of our key lemmas, Lemma 2, justifies this step.

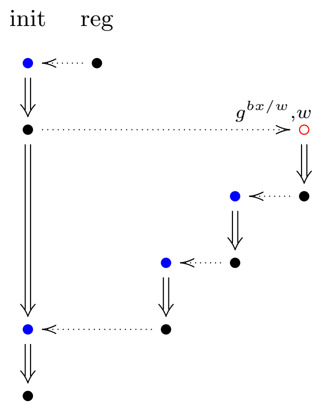


Fig. 7. Skeleton 4: Is $w = bx$? Or is there an exposed exponent w where $g^{bx/w}$ was sent by a regular participant?

Step 3, clean-up. From skeleton 4, CPSA considers the remaining possibilities in this branch of its analysis. First, it immediately eliminates the possibility $w = bx$, since the protocol offers no way for the adversary to obtain b and x.

In fact, because b and x are random values, independently chosen by different principals, the adversary cannot obtain their product without obtaining the values themselves.

CPSA then considers each protocol role in turn, namely the initiator, responder, and registration roles. Can any role transmit a DH value of the form $g^{bx/w}$, where the resulting inferred value for w would be available to the adversary?

In skeleton 5, it considers the case in which the initiator strand is the original starting strand, which transmits g^x. Thus, $x = bx/w$, which is to say $w = b$. However, since b is assumed uncompromised, the adversary cannot obtain it, and this branch is dead. Skeleton 6 explores the case where a different initiator strand sends g^z, so $z = bx/w$, i.e. $w = bx/z$. However, this is unobtainable, since it too is compounded from independent, uncompromised values b, x, z.

Skeleton 7 considers the responder case, and skeletons 8 and 9 consider a registration strand which is either identical with the initial one (skeleton 8) or not (skeleton 9). They are eliminated for corresponding reasons.

$$
\begin{array}{l}
\mathsf{init}(z_1, 4) \ \wedge\ \mathsf{init\_na}(z_1, na) \ \wedge\ \mathsf{init\_nb}(z_1, nb) \ \wedge\ \mathsf{init\_a}(z_1, a) \ \wedge\ \mathsf{init\_b}(z_1, b) \\
\quad \wedge\ \mathsf{init\_ltxa}(z_1, ltxa) \ \wedge\ \mathsf{init\_ltxb}(z_1, ltxb) \ \wedge\ \mathsf{init\_x}(z_1, x) \ \wedge\ \mathsf{init\_y}(z_1, y) \\
\quad \wedge\ \mathsf{reg}(z_2, 1) \ \wedge\ \mathsf{reg\_self}(z_2, b) \ \wedge\ \mathsf{reg\_l}(z_2, ltxb) \\
\quad \wedge\ (z_2, 1) \prec (z_1, 1) \ \wedge\ \mathsf{non}(ltxb) \\
\Longrightarrow \\
\quad \exists z_3. \\
\qquad \mathsf{resp}(z_3, 3) \ \wedge\ \mathsf{resp\_na}(z_3, na) \ \wedge\ \mathsf{resp\_nb}(z_3, nb) \ \wedge\ \mathsf{resp\_a}(z_3, a) \ \wedge\ \mathsf{resp\_b}(z_3, b) \\
\qquad \wedge\ \mathsf{resp\_ltxa}(z_3, ltxa) \ \wedge\ \mathsf{resp\_ltxb}(z_3, ltxb) \ \wedge\ \mathsf{resp\_x}(z_3, x) \ \wedge\ \mathsf{resp\_y}(z_3, y) \\
\qquad \wedge\ (z_2, 1) \prec (z_3, 1) \ \wedge\ (z_1, 2) \prec (z_2, 2) \ \wedge\ (z_2, 3) \prec (z_2, 3) \\
\qquad \wedge\ \mathsf{uniq\_at}(na, (z_1, 2)) \ \wedge\ \mathsf{uniq\_at}(nb, (z_2, 3))
\end{array}
$$

Fig. 8. Shape analysis sentence: UMX initiator impersonation

Thus, the whole subtree below skeleton 2 (Fig. 6) is dead, i.e. there is no possible execution to which skeleton 2 leads.

The entire analysis takes about 0.2 seconds.

Result of the analysis. Having eliminated everything below skeleton 2 (Fig. 6), the analysis has left only skeleton 1 (Fig. 5) as a shape. In that skeleton, the initiator has authenticated the responder with an exactly matching run. Thus, the analysis shows that an implication holds: For every execution, if it contains at least the structure shown in skeleton 0, then it has all of the structure shown in skeleton 1.

CPSA generates a *shape analysis sentence* that expresses this. Its antecedent describes the facts present in skeleton 0, and the conclusion describes the facts in skeleton 1. If there were multiple alternative shapes rather than just one, the conclusion would be a disjunction of the descriptions of the different possible outcomes. In the important special case in which the whole search tree is dead, so that the initial scenario—which may describe some undesired disclosure of values that should remain confidential—cannot occur, the conclusion is the disjunction of the empty set of formulas, i.e. the formula `false`.

Specifically, in our example analysis, the hypothesis is that there is an initiator strand and an registration strand, in which the registration strand generates $\mathsf{ltx}(B)$ and moreover $\mathsf{ltx}(B)$ is non-compromised. The conclusion is that there is also a responder strand with exactly matching parameters.

The formula generated by CPSA is shown, in a more humanly readable form, and without its leading universal quantifiers, in Fig. 8.

The variables z_1, z_2 range over strands; na, nb range over data; a, b range over names; and $ltxa, ltxb, x$ range over randomly chosen exponents. These are *transcendentals* of the exponent field. By contrast, y may be a field member that is not a simple random value, but e.g. a product of field values.

The strand z_1 is an instance of the *init* role, and it is of “full height,” meaning that all four steps have occurred. The next eight atomic formulas fix each of the parameters of strand z_1, using the predicates $\mathsf{init\_na}$, $\mathsf{init\_nb}$, etc., to make assertions about the parameters in question, and the variables na, nb, etc. to refer to their values.

The second strand z_2 is an instance of the *reg* role, with one node, the transmission node. It stipulates that the self parameter refers to the same intended peer b, with the same long term exponent $ltxb$. It states a precedence relation between the first node on the two strands, and assumes that $ltxb$ is non-compromised, i.e. never transmitted as part of a payload, or by the adversary.

The conclusion asserts the existence of a strand z_3 which is an instance of the *resp* role, and has the matching parameter values. It also records the precedence ordering, as determined by CPSA, and some freshness properties $\mathsf{uniq\_at}(na, (z_1, 2)) \ \wedge\ \mathsf{uniq\_at}(nb, (z_2, 3))$ for the nonces, which follow from the definitions of the roles.

4 Modeling DH values and executions

The analysis of Section 3 depends on some modeling decisions. For one, encryption and digital signature are assumed to satisfy the Dolev-Yao properties [10]:

digital signatures can be produced only using the signing key;
encryptions can be produced only from the plaintext using the encryption key;
plaintext can be recovered from an encryption only using the decryption key;
keys cannot be recovered from signatures or encryptions.

Step 1 assumes that, for the adversary to produce the encryption $\{\!|na, nb|\!\}_K$, he can obtain the key K. A regular responder may produce the encryption, as in skeleton 1, in a session with the right parameters, since $K = \mathsf{KCF}^{\mathsf{B}}(b, \beta, g^a, g^x)$.

There are also important modeling properties of the DH values. First, the adversary can certainly carry out the four basic algebraic operations on exponents α and β to obtain $\alpha + \beta$, $\alpha \cdot \beta$, etc. Given a DH value g^α and an exponent β, the adversary can exponentiate, obtaining $(g^\alpha)^\beta = g^{\alpha\beta}$. Given two DH values g^α and g^β, they may be combined to yield $g^\alpha g^\beta = g^{\alpha+\beta}$.

Thus, the exponents form a *field*, i.e. a structure with commutative addition and multiplication operations and identity elements $\mathbf{0}, \mathbf{1}$, related by a distributive law. Addition has an inverse, as does multiplication, for non-$\mathbf{0}$ divisors.

The DH values form a *cyclic group*, generated from a generator g by repeated multiplication. Since, e.g., $ggg = g^3$, we can regard the cyclic group as built from g by using field elements as exponents. Although we have been writing the group operation multiplicatively, and combining a group element and a field element by exponentiation, the algebra is isomorphic if the group operation is written additively as $+$. Then the field elements are combined with group elements by a scalar multiplication αP. We will continue to write the group operation with the same multiplicative convention.

Each field $\mathcal{F}$ determines a cyclic group, with domain $\{g^\alpha \colon \alpha \in \mathcal{F}\}$. Its group operation maps g^α and g^β to $g^{\alpha+\beta}$. Its inverse maps g^α to $g^{-\alpha}$ since $g^\alpha g^{-\alpha} = g^{\mathbf{0}}$.

But: what fields $\mathcal{F}$ are relevant? In particular, we must decide how to represent random choices of the regular principals (or the adversary) as field elements. The criterion for this depends on our view of the adversary.

Adversary model. A protocol designer designs a protocol with some scenarios of interest in mind, with authentication and confidentiality goals for each. The adversary's goal is to exhibit protocol executions that provide counterexamples to these goals of the designer. A counterexample is an execution, so the adversary must be able to supply every message received during the execution.

Executions involve cyclic group and field values, structures that are not freely generated, and our adversary works directly with polynomials, within the mathematical structures. Thus, our adversary performs group and field operations, which have the same effect regardless of how the structures are implemented as bitstrings. The adversary does not perform arbitrary efficient computations on bitstrings, unlike in the standard computational model.

This is similar to the generic group model [26,21]. This is an asymptotic computational model, in which the adversary is assumed to get negligible advantage from the bitstrings but can use group operations. Barthe et al. [3] show that an adversary which solves equations in a non-asymptotic model soundly approximates the generic group model. If the adversary's recipe for generating values for a recipient is a polynomial p_0, and the recipient expects the values produced by p_1, the adversary succeeds when the arguments satisfy $p_0 - p_1 = 0$. If the underlying fields are the prime order fields $\mathcal{F}_q$, then as q increases (or, better, as $\log q$ increases), the adversary wins with non-negligible probability only if p_0 and p_1 are identically equal (cf. also [19]). Otherwise, $p_0 - p_1$ has at most d zeros, where d is the (constant) maximum of their degrees.

Thus, the adversary uses polynomials as recipes, and wins when the polynomials evaluate to a result acceptable to the regular principals.

Fields and extension elements. The "variables" in these polynomials are *extension elements* that represent the random choices of the regular participants and the adversary. A set X of extension elements means a set of new values adjoined to a base field $\mathcal{F}$. They generate a new field $\mathcal{F}(X)$ in which the elements are polynomials in X, as well as quotients of polynomials. Extension elements come in two flavors. Some, called *algebraic extension elements*, like the square root of two, are introduced to supply a root for a polynomial, in this case $x^2 - 2$. Others, introduced without an associated polynomial, are called *transcendental extension elements*, because transcendentals such as π and e arise in this way. The random choices are effectively transcendental extension elements, because from the adversary's point of view, they are:

– disjoint from the underlying field: the adversary loses with overwhelming probability if he assumes a random choice equals a particular member.
– algebraically independent of each other: the adversary loses with overwhelming probability if he assumes the random choices will furnish a $\mathbf{0}$ for a polynomial p, other than the vacuous polynomial with all zero coefficients.

Fix an infinite set trsc as the transcendentals. We will work over a single base field $\mathbb{Q}$ of the rationals. $\mathbb{Q}$ is the right base field. If a set of polynomials has a solution in the finite field $\mathcal{F}_q$ for infinitely many choices of a prime q, then it also has solutions in $\mathbb{Q}$. Conversely, any solution in $\mathbb{Q}$ yields a solution in every $\mathcal{F}_q$.

Creation: $\mathbf{g}\uparrow \quad \mathbf{1}\uparrow \quad a\uparrow$ for $a\colon$ CREATE

Multiplicative: $\downarrow w_1 \Rightarrow \downarrow w_2 \Rightarrow w_1 \cdot w_2 \uparrow \qquad \downarrow w_1 \Rightarrow \downarrow w_2 \Rightarrow w_1/w_2 \uparrow$

$$\downarrow h \Rightarrow \downarrow w \Rightarrow \mathbf{exp}(h, w)\uparrow$$

Additive: $\downarrow w_1 \Rightarrow \downarrow w_2 \Rightarrow (w_1 + w_2)\uparrow \qquad \downarrow w_1 \Rightarrow \downarrow w_2 \Rightarrow (w_1 - w_2)\uparrow$

$$\downarrow \mathbf{exp}(h, w_1) \Rightarrow \downarrow \mathbf{exp}(h, w_2) \Rightarrow \mathbf{exp}(h, w_1 + w_2)\uparrow$$

Construction: $\downarrow m \Rightarrow \downarrow K \Rightarrow \{\!|m|\!\}_K \uparrow \qquad \downarrow m_1 \Rightarrow \downarrow m_2 \Rightarrow (m_1, m_2)\uparrow$

Destruction: $\downarrow (m_1, m_2) \Rightarrow m_1 \uparrow \qquad \downarrow (m_1, m_2) \Rightarrow m_2 \uparrow$

$$\downarrow \{\!|m|\!\}_K \Rightarrow \downarrow K^{-1} \Rightarrow m\uparrow$$

$\downarrow m$ and $m\uparrow$ mean reception and transmission of m, resp.
$+, -$ mean field addition and subtraction

Fig. 9. Adversary strands

Analysis in $\mathbb{Q}(\mathsf{trsc})$ is thus faithful for adversary strategies that work in $\mathcal{F}_q(\mathsf{trsc})$ for infinitely many q. Thus, for the remainder of this paper, we fix $\mathcal{F} = \mathbb{Q}(\mathsf{trsc})$ as the extension field. Fix the cyclic group $\mathcal{C}$ to be the cyclic group generated from $\mathbf{g}$ using as exponents the members of $\mathcal{F}$.

Viewing random choices as members of trsc justifies our reasoning in Section 3, Step 3, where we argued that the adversary could not obtain products of independent random choices of the regular principals. The distinct random choices cannot cancel out to leave a value the adversary can obtain.

Adversary strands. The adversary constructs new values from available values via the strands in Fig. 9. In the *creation* strands, the sort CREATE is the union of sorts for atomic symmetric keys; asymmetric keys; texts; principal names; and transcendentals trsc. Group elements are not created; instead, one obtains a field element and exponentiates. The *multiplicative* and *additive* operations apply the field and group operations. The remaining strands are standard Dolev-Yao operations. The adversary concatenates or separates tuples; and applies encryption, decryption, or digital signature given the necessary keys, plaintext, or cipher text. Messages can be non-atomic symmetric keys. The adversary "routes" messages from one adversary strand to another for compound operations.

Corruption. We do not model corruption as an adversary action. Instead, we treat corruption as an action of a regular participant, who may disclose any parameter. The *registration* role shown in Fig. 2, which transmits the long term value $\mathsf{ltx}(P)$ in its last step, is an example. We can select which executions to query by stipulating that particular values are non-compromised: Then, the strand that chooses that value does not progress to the disclosure in the relevant executions. Any value not governed by an assumption may be compromised.

Although DHCR allows compromise of long term exponents, but not ephemerals, that is specific to the example. Protocols can be instrumented to allow compromise of any parameter. CPSA queries can exclude specific compromises [25].

Messages. Messages include the field $\mathcal{F}$ and its generated group $\mathcal{C}$, with additional sorts of atomic symmetric keys, asymmetric keys, texts, and names. We close the messages under free operations of tupling and cryptographic operations, e.g. symmetric and asymmetric encryption, digital signature, and hashing.

We do not distinguish notationally among symmetric encryption, asymmetric encryption, and digital signature. We say that $K = K^{-1}$ whenever K is an atomic symmetric key or a compound key, and $K \neq K^{-1}$ iff K is an asymmetric key. Then the adversary powers summarized in Fig. 9 are reasonable. We regard a hash $\#(m)$ as a symmetric encryption $\{|\mathbf{0}|\}_m$, using the argument as key to encrypt a fixed, irrelevant value. We use the word *encryption* as shorthand for all of these cryptographic operations.

Our claims hold across a wide range of choices of operators for the tupling and cryptographic operators, as long as they freely generate their results.

Definition 1. *We say that m_0 is* visible in *iff $m_0 = m_1$, or recursively m_1 is a tuple (m_2, m_3) and m_0 is visible in either m_2 or m_3.*

We say that m_0 is carried in m_1 *iff $m_0 = m_1$, or recursively m_1 is either:*
a tuple *$m_1 = (m_2, m_3)$ and m_0 is carried in either m_2 or m_3; or*
an encryption *$m_1 = \{|m_2|\}_K$ and m_0 is carried in m_2.*

In defining *visible*, we do not look inside encryptions at all. In *carried*, we look inside the plaintext m_2, but not the key K.

Executions are bundles. In our model, the executions of a protocol Π are *bundles*. A bundle is a set of strands that are either adversary strands or else instances of the roles of Π, or initial segments of them (see Sec. 2.1), and in which every reception is explained by an earlier matching transmission. We formalize bundles via *nodes*, i.e. the transmission and reception events along the strands.

A binary relation $\rightarrow$ on nodes is a *communication relation* iff $n_1 \rightarrow n_2$ implies that n_1 is a transmission node, n_2 is a reception node, and $\mathsf{msg}(n_1) = \mathsf{msg}(n_2)$.

Definition 2. *Let $\mathcal{B} = (\mathcal{N}, \rightarrow)$ be a set of nodes together with a communication relation on $\mathcal{N}$. $\mathcal{B}$ is a* bundle *iff:*

1. *$n_2 \in \mathcal{N}$ and $n_1 \Rightarrow n_2$ implies $n_1 \in \mathcal{N}$;*
2. *$n_2 \in \mathcal{N}$ and n_2 is a reception node implies there exists a unique $n_1 \in \mathcal{N}$ such that $n_1 \rightarrow n_2$; and*
3. *Letting $\Rightarrow_{\mathcal{B}}$ be the restriction of $\Rightarrow$ to $\mathcal{N} \times \mathcal{N}$, the reflexive-transitive closure $(\Rightarrow_{\mathcal{B}} \cup \rightarrow)^*$ is a well-founded partial order.*

We write $\mathsf{nodes}(\mathcal{B})$ for $\mathcal{N}$, and $\preceq_{\mathcal{B}}$ for $(\Rightarrow_{\mathcal{B}} \cup \rightarrow)^*$.

Clause 3 is an *induction principle* for bundles. Any non-empty set S of nodes will have nodes that are $\preceq_{\mathcal{B}}$-minimal among nodes in S; proofs often take cases on $\preceq_{\mathcal{B}}$-minimal nodes.

Definition 3. *Let $\mathcal{B}$ be a bundle, m a message, and $n \in \mathsf{nodes}(\mathcal{B})$. Then m* originates at *n iff m is carried in $\mathsf{msg}(n)$, n is a transmission node, and for all earlier $n_0 \Rightarrow^+ n$ on the same strand, m is not carried in $\mathsf{msg}(n_0)$.*

The message m originates uniquely *in $\mathcal{B}$ iff there is exactly one $n \in \mathsf{nodes}(\mathcal{B})$ such that m originates at n. The message m is* non-originating *in $\mathcal{B}$ if there is no such $n \in \mathsf{nodes}(\mathcal{B})$.*

We can now define two special classes of protocol that CPSA analyzes.

One is that when an instance of a role transmits a field element in carried position, then that field element should simply be a transcendental $x \in \mathsf{trsc}$. This covers two important reasons why protocols disclose field values. First, disclosing a random choice models a corruption step, as in the registration role, Fig. 2. Second, disclosing a random choice may be part of a decommit step that allows a third party to validate a previously committed value. In these cases, an independent random choice, i.e. some $x \in \mathsf{trsc}$, is the value to disclose. There are, by contrast, also protocols such as signature protocols in which polynomials such as $r - xe$ may be disclosed. We do not model these as protocols, but can certainly model systems that use these types of signature as a primitive cryptographic operation. We say that a protocol *separates disclosures* if it satisfies this property.

Second, many protocols—though again, not all—involve field multiplication and division, but not field addition or subtraction. The four operations together lead in general to an undecidable class of unification problems. Thus, we adopt a limitation that is common to many mechanized protocol analysis systems [12,22,7,15], and focus on the protocols in which the regular participants do not add or subtract field values, or use the group operation (which is addition in exponents).

Recall that a protocol contains a set of roles, which are strands containing parameters (i.e. variables) and terms built from them. Some of these variables are of sort *transcendental*, and variables and compound terms may be of sort *field*. The sort *transcendental* is a subsort of field, and in our current implementation, all terms of the narrower sort *transcendental* are variables. Bundles, however, contain actual field members, i.e. polynomials in our extension field $\mathcal{F} = \mathbb{Q}(\mathsf{trsc})$.

Definition 4. *Let Π be a protocol. Π* separates disclosures *iff, for all transmission nodes $n \in \mathsf{nodes}(\Pi)$, and all v of sort* field *carried in $\mathsf{msg}(n)$, v is simply a parameter of sort* transcendental.

Π is multiplicative *iff, for all transmission nodes $n \in \mathsf{nodes}(\Pi)$, and all v of sort* field *occurring in $\mathsf{msg}(n)$, neither addition nor subtraction occurs in v.*

A bundle $\mathcal{B}$ is purely monomial *iff, for every node $n \in \mathsf{nodes}(\mathcal{B})$ and every $p \in \mathcal{F}$, if p occurs in $\mathsf{msg}(n)$, then p is a monomial.*

The methods of our earlier paper [16] show that, when Π is a multiplicative protocol, and a security goal G for Π has a counterexample, then there is a purely monomial bundle $\mathcal{B}$ that is a counterexample for G.

CPSA focuses on protocols that are multiplicative and separate disclosures. Thus, we need consider only purely monomial bundles. For more detail on this section, see the long version [17].

5 Two key lemmas

We now state our two main results about how the adversary obtains field and group elements. The first holds even for protocols using the additive structure.

When a field element is exposed to the adversary in a bundle, then every transcendental present in it has also been exposed:

Lemma 1. *Suppose Π separates disclosures, $\mathcal{B}$ is a Π-bundle, and $x \in \mathsf{trsc}$ has non-0 degree in $p \in \mathcal{F}$. Let $n_p \in \mathsf{nodes}(\mathcal{B})$ be a node where p is visible in $\mathsf{msg}(n_p)$. There is a node $n_x \in \mathsf{nodes}(\mathcal{B})$ such that $n_x \preceq_{\mathcal{B}} n_p$, and x is visible in $\mathsf{msg}(n_x)$.*

The proof in the appendix illustrates a standard technique, namely proof using the induction principle on bundles (Def. 2, Clause 3), after which we take cases on the minimal node in a set S.

The second lemma says how an adversary obtains a group element g^{μ}, in purely monomial bundles. It says that μ is a product of two monomials. The first, ν, consists of compromised transcendentals. The rest, ξ, yields a group element g^{ξ} that some regular participant has sent in carried position.

Lemma 2. *Suppose Π separates disclosures; $\mathcal{B}$ is a purely monomial Π-bundle; $n_{\mu} \in \mathsf{nodes}(\mathcal{B})$; and $\mathbf{g}^{\mu} \in \mathcal{C}$ is carried in $\mathsf{msg}(n_{\mu})$. Then there is a monomial $\nu \in \mathcal{F}$ s.t. ν is a product of transcendentals visible before n_{μ}, and either*

1. *$\nu = \mu$ or else*
2. *letting $\xi = \mu/\nu$, there is a* regular *transmission node $n_{\xi} \in \mathsf{nodes}(\mathcal{B})$ such that $n_{\xi} \preceq_{\mathcal{B}} n_{\mu}$ and $\mathbf{g}^{\xi}$ is carried in $\mathsf{msg}(n_{\xi})$.*
   Moreover, either $\nu = \mathbf{1}$ or $\mathbf{g}^{\xi}$ was previously visible.

The proof is similar in form; in the main case, g^{μ} is constructed by an adversary exponentiation, and the new exponent factor is combined into ν.

6 The CPSA algorithm

Overall algorithm. CPSA manipulates descriptions of executions that we call *skeletons.* The initial scenario is a skeleton from which CPSA starts. At any step, CPSA has a set $\mathcal{S}$ of skeletons available. If $\mathcal{S}$ is empty, the run is complete.

Otherwise, CPSA selects a skeleton $\mathbb{A}$ from $\mathcal{S}$. If $\mathbb{A}$ is *realized*, meaning that it gives a full description of some execution, then CPSA records it as a result. Otherwise, there is some reception node n within $\mathbb{A}$ that is not explained. This n is the *target node.* That means that CPSA cannot show how the message received by n could be available, given the actions the adversary can perform on his own, or using messages received from earlier transmissions.

CPSA replaces $\mathbb{A}$ with a "cohort." This is a set $\mathbb{C}_1, \ldots, \mathbb{C}_k$ of extensions of $\mathbb{A}$. CPSA must not "lose" executions: For every execution satisfying $\mathbb{A}$, there should be at least one of the $\mathbb{C}_i$ which this execution satisfies. When $k = 0$ and there are no cohort members, CPSA has recognized that $\mathbb{A}$ is *dead*, i.e. it describes no executions. CPSA then repeats this process starting with $\mathcal{S} \setminus \{\mathbb{A}\} \cup \{\mathbb{C}_1, \ldots, \mathbb{C}_k\}$.

Cohort selection. CPSA generates its cohorts by adding one or more facts, or new equalities, to $\mathbb{A}$, to generate each $\mathbb{C}_i$.

The facts to add is based on a taxonomy of the executions satisfying $\mathbb{A}$. In each one of them, the reception on the target node n must somehow be

Regular transmission: A regular principal has transmitted a message in this execution which is not described in $\mathbb{A}$. E.g. Skeleton 1 in step 1 of Section 3.
Encryption key available: An encrypted value is received. CPSA explores if the adversary can obtain the encryption key. E.g. Skeleton 2 in step 1.
Decryption key available: A value escaped from a previously transmitted encryption. CPSA explores if the adversary can obtain the decryption key.
Specialization: The execution satisfies additional equations, not included in $\mathbb{A}$. CPSA explores if in this special case the adversary can obtain the target node message.
DH value computed: The adversary obtains a DH value g^{α}, by exponentiation $g^{\alpha/w}$ to power w, which must also be available. (Lemma 2.) Skeleton 4 in step 2.
Exponent value computed: The adversary obtains an exponent xw. There are then two subcases (Lemma 1.):
1. CPSA explores how x and w are obtained; or
2. w is instantiated as some v/x, and x cancels out. Thus, x is absent from the instance of xw.

Fig. 10. Kinds of cohort members.

explained. There are only a limited number of types of explanation, which are summarized in Fig. 10. The first three kinds are identical to the forms they take without Diffie-Hellman; they are essentially about encryption and freshness. The *Specialization* clause is implemented by a unification algorithm implemented as a combination of theories. It treats transcendentals as primitive values, leading to faster solutions. The last two clauses, justified by Lemmas 2 and 1 (resp.), are new. The last clause, which is infrequently used, is applied in forward secrecy results in which the exponents take center stage.

7 Results and related work

The CPSA implementation is highly efficient (Figure 11), running on a mid-2015 MacBook Pro with a 4-core 2.2 GHz Intel Core i7 processor, running up to 8 parallel threads using the Haskell run-time system.

We analyzed DHCR with each key derivation option, confirming the claims of Section 2. Each CPSA run checked five scenarios, determining the initiator and responder's guarantees under two sets of assumptions, as well as a forward secrecy property. Each run examines 90 to 230 skeletons.

We ran CPSA on the Station-to-Station protocol [9], together with two weakenings of it. In one, we do not assume the peer necessarily chooses a fresh exponent. In the other, we omit the flip in the second signed unit, enabling a reflection attack. In each, we test authentication, key secrecy, and forward secrecy, finding attacks against the weakened versions, and examining 55–190 skeletons.

We also ran a rich set of variants of Internet Key Exchange (IKE) versions 1 and 2. Scyther analyzed this same set of variants circa 2010 [7], requiring more substantial runtimes, although recent timings are similar to ours (cf. also [4]). This suggests that CPSA is broadly efficient, with or without DH. Performance

DHCR, STS: Example & Time

dhcr-um	4.06s	dhcr-umx	0.72s	dhcr-um3	0.47s
sts	0.36s	sts-weak	0.08s	sts-unflip	0.17s

IKEv1: Example & Time

IKEv1-pk2-a	1.06s	IKEv1-pk2-a2	1.02s
IKEv1-pk2-m	0.49s	IKEv1-pk2-m2	0.58s
IKEv1-pk-a1	1.27s	IKEv1-pk-a12	1.09s
IKEv1-pk-a2	1.00s	IKEv1-pk-a22	1.10s
IKEv1-pk-m	0.51s	IKEv1-pk-m2	0.49s
IKEv1-psk-a	0.43s	IKEv1-psk-m	0.68s
IKEv1-psk-m-perlman	0.69s	IKEv1-quick	0.66s
IKEv1-psk-quick-noid	0.65s	IKEv1-quick-nopfs	0.09s
IKEv1-sig-a1	0.15s	IKEv1-sig-a2	0.16s
IKEv1-sig-a-perlman	0.17s	IKEv1-sig-a-perlman2	0.19s
IKEv1-sig-m	0.21s	IKEv1-sig-m-perlman	0.19s

IKEv2: Example & Time

IKEv2-eap	1.35s	IKEv2-eap2	1.36s
IKEv2-mac	0.76s	IKEv2-mac2	0.97s
IKEv2-mac-to-sig	0.83s	IKEv2-mac-to-sig2	0.82s
IKEv2-sig	0.56s	IKEv2-sig2	0.54s
IKEv2-sig-to-mac	0.70s	IKEv2-sig-to-mac2	0.69s

Fig. 11. CPSA runtime: DHCR-UM*, Station-to-Station, Intern. Key Exch. v. 1 and 2.

data for Tamarin and Maude-NPA is less available. We analyzed each IKE variant for about five properties, yielding conclusions similar to those drawn using Scyther. We found no novel attacks, but did sharpen the previous analysis, because CPSA reflects the algebraic properties of Diffie-Hellman natively, while Scyther emulated some properties of Diffie-Hellman.

Related work. Our primary novelty is enrich-by-need for DH. CPSA provides a visualization of all of the minimal, essentially different executions compatible with a starting scenario. CPSA also computes a strongest security goal for the scenario. Moreover, CPSA is founded in the fields and cyclic groups that DH manipulates, with a clear connection with the generic group model [26,21,3].

It is, however, hardly the first method for DH. Within AVISPA [2,29], CL-Atse treated DH within a bounded session model [28]. In the unbounded session model, Küsters and Truderung [15] allow using ProVerif for DH; essentially, they compute *a priori* a set of DH terms that will suffice for ProVerif, and equip those terms with the rewrites ProVerif needs. The method is efficient and clever. However, it lacks a direct connection with the underlying algebra, and is not intended to support enrich-by-need. Like ProVerif, Scyther was not designed for DH's algebra; special-purpose roles were added to coerce messages to different forms, thereby simulating the commutative principle $g^{xy} = g^{yx}$ [6].

By contrast, multiplicative DH protocols are native to Maude-NPA [12] and Tamarin [22]. Indeed, both of these systems allow general rewrite systems, even with associative-commutative operators, which is not a CPSA goal. However, neither supports enrich-by-need. CPSA appears to provide a higher level of automation and efficiency than Maude-NPA. CPSA's treatment of the algebra of DH is also preferable, as Maude-NPA lacks a multiplicative inverse. Tamarin has a faithful theory for the multiplicative fragment, which—since it lacks 0—has an equational axiomatization. Although it offers great flexibility, performance information on Tamarin is hard to find.

Neither Maude-NPA nor Tamarin appears to have a sharp distinction between the primitive random choices—our "transcendentals"—and other exponents. Their variables are not similar to transcendentals, since they range over all exponents. Thus, random choices may appear or disappear in unification, so the distinction between the narrower sort of transcendentals and the larger sort of exponents sharpens our treatment of unification [17]. It also helped us to formulate the Lemmas 1–2.

The distinction between underlying transcendentals and exponents in general, and the two lemmas to which it leads, are reusable ideas that could well provide other systems with a strategy for more focused search.

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. Gilles Barthe, Edvard Fagerholm, Dario Fiore, John C. Mitchell, Andre Scedrov, and Benedikt Schmidt. Automated analysis of cryptographic assumptions in generic group models. In *CRYPTO*, volume 8616 of *LNCS*, pages 95–112. Springer, 2014.
4. David A. Basin, Cas Cremers, and Simon Meier. Provably repairing the ISO/IEC 9798 standard for entity authentication. *Journal of Computer Security*, 21(6):817–846, 2013.
5. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – Crypto '93 Proceedings*, number 773 in LNCS. Springer-Verlag, 1993.
6. Cas Cremers. Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. In *Computer Security–ESORICS 2011*. Springer, 2011.
7. Cas Cremers and Sjouke Mauw. *Operational semantics and verification of security protocols*. Springer, 2012.
8. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.
9. Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.

10. Daniel Dolev and Andrew Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.
11. Daniel J. Dougherty and Joshua D. Guttman. Decidability for lightweight Diffie-Hellman protocols. In *IEEE Symposium on Computer Security Foundations*, 2014.
12. Santiago Escobar, Catherine Meadows, and José Meseguer. Maude-NPA: Cryptographic protocol analysis modulo equational properties. In *Foundations of Security Analysis and Design V, FOSAD 2007–2009 Tutorial Lectures*, volume 5705 of *Lecture Notes in Computer Science*, pages 1–50. Springer, 2009.
13. 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.
14. Joshua D. Guttman. Establishing and preserving protocol security goals. *Journal of Computer Security*, 22(2):201–267, 2014.
15. 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.
16. Moses Liskov and F. Javier Thayer. Modeling Diffie-Hellman derivability for automated analysis. In *IEEE Computer Security Foundations*, pages 232–243, 2014.
17. Moses D. Liskov, Joshua D. Guttman, John D. Ramsdell, Paul D. Rowe, and F. Javier Thayer. Enrich-by-need protocol analysis for Diffie-Hellman (extended version). `http://arxiv.org/abs/1804.05713`, April 2018.
18. Moses D. Liskov, Paul D. Rowe, and F. Javier Thayer. Completeness of CPSA. Technical Report MTR110479, The MITRE Corporation, March 2011. `http://www.mitre.org/publications/technical-papers/completeness-of-cpsa`.
19. 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.
20. Gavin Lowe. A hierarchy of authentication specifications. In *10th Computer Security Foundations Workshop Proceedings*, pages 31–43. IEEE CS Press, 1997.
21. Ueli M. Maurer. Abstract models of computation in cryptography. In *Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, 2005.
22. Simon Meier, Benedikt Schmidt, Cas Cremers, and David A. Basin. The tamarin prover for the symbolic analysis of security protocols. In *Computer Aided Verification (CAV)*, pages 696–701, 2013.
23. John D. Ramsdell. Deducing security goals from shape analysis sentences. The MITRE Corporation, April 2012. `http://arxiv.org/abs/1204.0480`.
24. John D. Ramsdell and Joshua D. Guttman. CPSA: A cryptographic protocol shapes analyzer, 2009. `http://hackage.haskell.org/package/cpsa`.
25. Paul D. Rowe, Joshua D. Guttman, and Moses D. Liskov. Measuring protocol strength with security goals. *International Journal of Information Security*, 15(6):575–596, November 2016. DOI 10.1007/s10207-016-0319-z, `http://web.cs.wpi.edu/~guttman/pubs/ijis_measuring-security.pdf`.
26. Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.
27. F. Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.
28. Mathieu Turuani. The CL-Atse protocol analyser. In *Rewriting Techniques and Applications*, pages 277–286. Springer, 2006.

29. Luca Viganò. Automated security protocol analysis with the AVISPA tool. *Electronic Notes in Theoretical Computer Science*, 155:61–86, 2006.

A Appendix: Proofs

Lemma 1. Suppose Π separates disclosures, $\mathcal{B}$ is a Π-bundle, and $x \in \mathsf{trsc}$ has non-0 degree in $p \in \mathcal{F}$. Let $n_p \in \mathsf{nodes}(\mathcal{B})$ be a node where p is visible in $\mathsf{msg}(n_p)$. There is a node $n_x \in \mathsf{nodes}(\mathcal{B})$ such that $n_x \preceq_{\mathcal{B}} n_p$, and x is visible in $\mathsf{msg}(n_x)$.

Proof. Choose $\mathcal{B}$, and if there are any x, p, n_p that furnish a counterexample let $n_p \in \mathsf{nodes}(\mathcal{B})$ be $\preceq_{\mathcal{B}}$-minimal among counterexamples for any x, p. Observe first that $x \neq p$, since if $x = p$ this is not a counterexample: let $n_x = n_p$.

Since p is carried in $\mathsf{msg}(n_p)$, there exists an $n_o \preceq_{\mathcal{B}} n_p$ such that p originates on n_o. By the definition of originates, n_o is a transmission node.

First, we show that n_o does not lie on an adversary strand, by taking cases on the adversary strands. The **creation** strands that emit values in FLD originate $\mathbf{1}$ and transcendentals $y\colon \mathsf{trsc}$. But x is not present in $\mathbf{1}$, and if x is present in y, then x and y are identical, which we have excluded.

If n_o lies on a **multiplicative** or **additive** strand, then it takes incoming field values p_1, p_2. Since x has non-zero degree in p only if it has non-zero degree in at least one of the p_i, this contradicts the $\preceq_{\mathcal{B}}$-minimality of the counterexample.

Node n_o does not lie on an **construction** or **destruction** strand, which never originate field values. Thus, n_o does not lie on an adversary strand.

Finally, n_o does not lie on a regular strand of Π: Since Π separates disclosures, if n_o originates the field value p, then p is a transcendental. Thus, if x is present, $p = x$, which was excluded above. ☐

Lemma 2. Suppose Π separates disclosures; $\mathcal{B}$ is a purely monomial Π-bundle; $n_\mu \in \mathsf{nodes}(\mathcal{B})$; and $\mathbf{g}^\mu \in \mathcal{C}$ is carried in $\mathsf{msg}(n_\mu)$. Then there is a monomial $\nu \in \mathcal{F}$ s.t. ν is a product of transcendentals visible before n_μ, and either

1. $\nu = \mu$ or else
2. letting $\xi = \mu/\nu$, there is a *regular* transmission node $n_\xi \in \mathsf{nodes}(\mathcal{B})$ such that $n_\xi \preceq_{\mathcal{B}} n_\mu$ and $\mathbf{g}^\xi$ is carried in $\mathsf{msg}(n_\xi)$.
   Moreover, either $\nu = \mathbf{1}$ or $\mathbf{g}^\xi$ was previously visible.

Proof. Let $\mathcal{B}$ be a bundle, let $n_\mu \in \mathsf{nodes}(\mathcal{B})$, and assume inductively that the claim holds for all nodes $n \prec n_\mu$. If n_μ is a reception node, then the (earlier) paired transmission node satisfies the property by the IH. However, the same ν and n_ξ also satisfy the property for n_μ. If n_μ is a regular transmission, then the conclusion holds with $\nu = \mathbf{1}$, the empty product of transcendentals.

So suppose n_μ lies on an adversary strand. If n_μ transmits the group element $\mathbf{g}$, then let $\nu = \mu = \mathbf{1}$. The constructive strands for tupling or encryption provide no new group elements in carried position. Nor do the destructive strands for untupling or decryption.

Thus, the remaining possibility is that n_μ is the transmission on an exponentiation strand $-h \Rightarrow -w \Rightarrow +\mathbf{exp}(h, w)$ where $\mathbf{exp}(h, w) = g^\mu$. By the IH, for the node receiving $h \in \mathcal{C}$, the property is met. Thus, $h = \mathbf{g}^{\mu_0}$, where there exist ν_0, ξ_0 satisfying the conditions.

Hence, we may take $\nu = \nu_0 w$ and $\xi = \xi_0$. By Lemma 1, w is a product of previously visible transcendentals, so the requirements are met. $\square$