Logical Protocol Analysis for Authenticated Diffie-Hellman*

Daniel J. Dougherty and Joshua D. Guttman

ABSTRACT

Diffie-Hellman protocols for authenticated key agreement construct a shared secret with a peer using a minimum of communication and using limited cryptographic operations. However, their analysis has been challenging in computational models and especially in symbolic models.

In this paper, we develop a logical framework for protocol analysis based on strand space ideas. We show that it identifies exact assumptions on the behavior of a certifying authority. These assumptions prevent attacks on two authenticated DH protocols, the Unified Model and Menezes-Qu-Vanstone (MQV).

Verification within our framework implies that the adversary has no strategy that works uniformly, independent of the choice of the cyclic group in which the protocol operates. Computational soundness would assert that an adversary strategy successful in groups satisfying the Decisional Diffie-Hellman assumption would also furnish a uniform, group-independent strategy. Computational soundness awaits further investigation.

1 INTRODUCTION

The Diffie-Hellman key exchange [2] is a widely used cryptographic idea. Each principal A, B chooses a random value, x, y resp., raising a base g to this power modulo a suitable prime p:

$$A, x \qquad \bullet \xrightarrow{g^x} \qquad \xleftarrow{g^y} \bullet \qquad B, y \qquad (1)$$

They can then both compute the value $(g^y)^x = g^{xy} = (g^x)^y$. Since there is reason to believe that g^{xy} is indistinguishable from g^z for randomly chosen z, we can treat g^{xy} as a new shared secret for A, B. The protocol is thus secure against a passive adversary, who observes what the compliant principals do, but can neither create messages nor alter (or misdirect) messages of compliant principals.

It is however certainly not secure against an active adversary, which can choose its own x', y', sending $g^{y'}$ to A instead of g^y, and sending $g^{x'}$ to B instead of g^x. In this case, each of A, B actually shares a different key with the adversary, who can act as a man in the middle in any conversation between them. So as not to prejudice ourselves in evaluating the possible attacks, we will write R_B for the public value that A receives, purportedly from B, and R_A for the public value that B receives, purportedly from A. In the intended case, $R_A = g^x$ and $R_B = g^y$.

One approach to authenticating a Diffie-Hellman exchange, originating in the Station-to-Station protocol STS [3], is digitally to sign parts of the exchange. For instance, in a simplified STS, the exchange in Eqn. 1 is followed by the signed messages:

$$A \qquad \bullet \xrightarrow{[\![\, g^x \,\hat{}\, R_B \,]\!]_A} \qquad \xleftarrow{[\![\, g^y \,\hat{}\, R_A \,]\!]_B} \bullet \qquad B \qquad (2)$$

where the signatures exclude a man in the middle.[1] STS requires an additional message transmission and reception for each participant, in each session. Each participant must also prepare and verify a digital signature on each exchange. STS requires some public key infrastructure to certify the signature verification keys of A and B.

An alternative to digital signatures is *implicit verification* [1]. Here the goal is to ensure that any principal that can compute the same value as A *can only* be B. To implement this idea, we have each principal maintain a long-term secret, which we will write a for principal A, b for B; they publish the long-term public values g^a, g^b, which we will refer to as Y_A, Y_B, etc. The trick is to build the use of the private values a, b into the computation of the shared secret, so that only A, B can do it. For instance, in the "Unified Model" UM of Ankney, Johnson, and Matyas, the principals combine long term values with short term values by concatenating and hashing. They follow the exchange of Eqn. 1 with these computations, where $H(x)$ is a hash of x:

$$A:\ k = H({Y_B}^a \,\hat{}\, {R_B}^x) \quad B:\ k = H({Y_A}^b \,\hat{}\, {R_A}^y), \qquad (3)$$

obtaining a shared value when $R_A = g^x$ and $R_B = g^y$.

*Supported by the National Science Foundation under grant CNS-0952287.

[1] We use $t \,\hat{}\, t'$ for the result of concatenating t with t'. A digitally signed message $[\![\, t \,]\!]_A$ means the message t concatenated with the result of a digital signature algorithm applied to a hash of t. It should be keyed with a signing key associated with the principal A.

Here again some public key infrastructure is required so that each principal knows to associate the intended peer P with the right public value to Y_P. However, no digital signature needs to be generated or checked specific to this run. If A frequently has sessions with B, A can amortize the cost of verifying B's certificate once, by keeping Y_B in secure storage.

One might prefer a protocol in which the operations are only algebraic, as distinguished from UM's combination of algebraic operations such as exponentiation with the rather different operations of concatenation and hashing. Indeed MQV [11] does exactly this, computing the key via the rules:

$$A:\ k = (R_B \cdot {Y_B}^{\overline{R_B}})^{s_A} \qquad B:\ k = (R_A \cdot {Y_A}^{\overline{R_A}})^{s_B} \tag{4}$$

where $s_A = x + a\overline{R_A}$ and $s_B = y + b\overline{R_B}$. The "bar" operator coerces numbers mod p to a convenient form in which they can be used as exponents; we will discuss it more below. Now, in a successful run, A obtains the value

$$(g^y \cdot (g^b)^{\overline{g^y}})^{s_A} = (g^{(y+b\overline{g^y})})^{(x+a\overline{g^x})} = g^{(s_B \cdot s_A)} \tag{5}$$

and B obtains $g^{s_A \cdot s_B}$, which is the same value. This protocol is challenging, from an algebraic point of view, to model and to analyze. There is indeed some controversy about MQV [7, 9, 12], and it is hard to analyze these protocols in computational models. Until now, they have been out of reach in symbolic models [4, 8, 10]. The problem is that these symbolic approaches to protocol analysis have relied on unification as a central part of their reasoning. Unifiability, in the presence of the ring structure used in Eqn. 5, is undecidable, essentially by the unsolvability of Hilbert's tenth problem.

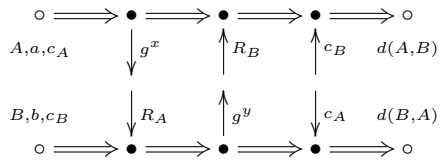
Nevertheless, interesting and relevant problems could be clarified by a more suitable symbolic approach. Exactly what constraints do the protocols assume about the authority that certifies public values Y_P? When are the protocols sound without key confirmation, or without including the principals' identities in key derivation? How can we define "implicit authentication"?

Goals of this paper. We develop a method for reasoning about implicit authentication for DH-style protocols within a symbolic model, namely a logical version of the strand space model. We infer exact conditions on behavior of the certification authorities for the long-term public values in protocols such as UM and MQV.

We start (Section 2) by analyzing the UM. We focus on clarifying the message structure, the limitations on what message values the adversary can generate, and the assumptions about the certificate authority CA. The protocol ensures authentication if the CA never re-certifies the same long-term public value, associating it with different identifies. There are various ways that the CA could achieve this, and the choice among them is essentially irrelevant from the current point of view.

In Section 3, we formalize the basic strand notions in a way that is uniform with regard to the choice of message algebra. Section 4 then fills in the specifics of the Diffie-Hellman message algebra needed for MQV. This preserves the simpler algebraic structure on which UM depends. In Section 5, we prove the lemma used informally in Section 2, that states that the adversary can cause a new value x to enter an exponentiation only if he possesses x. Finally, in Section 6 we prove that MQV achieves its authentication goal assuming that the CA satisfies two properties:

- The CA never certifies any one long-term value Y with two different identities;
- A certified long term value Y is never used in any session which began before Y was certified.



$$k_A = H({Y_B}^a \hat{} {R_B}^x) \qquad k_B = H({Y_A}^b \hat{} {R_A}^y)$$

$$c_P = [\![\, \mathsf{cert}\ Y_P \hat{} P \,]\!]_{CA} \qquad d(P, P') = \mathsf{keyrec}\ P \hat{} P' \hat{} k_P$$

Figure 1: The UM Protocol

The latter property can be achieved in several ways. For instance, the CA may wait for a period t_{CA} before responding to a certification request, while a client session always times out before t_{CA} has elapsed. Alternatively, clients could "age" certificates before use, with a similar timeout.

2 THE UNIFIED MODEL

We first illustrate our style of protocol specification with the Unified Model UM. We regard UM as involving several events that were left implicit in the introduction. We summarize this in Fig. 1. We write bullets • for transmission and reception events and circles ○ for "neutral" events, which consult or update local state. The top and bottom rows shows the initiator's actions and the responder's actions, respectively. In the initiator we have successively:

1. A neutral node consulting its principal's local state to obtain its own name, long term secret DH value, and certificate;
2. A transmission node sending R_A, where R_A is g^x for a freshly chosen value x;
3. A reception node receiving the ephemeral value R_B;
4. A reception node for the peer's certificate c_B associating Y_B with $B's$ identity; and
5. A neutral node that deposits the principals' names and the resulting session key as a new *key record* into the principal's local state database.

The responder's actions are identical except that in place of nodes 2 and 3, we have their duals; i.e. a reception node receiving some ephemeral value R_A, followed by a node transmitting R_B, where R_B is g^y for a freshly chosen value y.

Some assumptions about UM runs are not explicit in Fig. 1:

Origination. Compliant participants attempt to choose their long-term values a, b and ephemeral values x, y at random from a large set. The probability of finding a collision among their choices is negligible.

Since long term and ephemeral values a, x are never transmitted as ingredients of any message, they are *non-originating*.

Moreover, the values g^a, g^x *originate uniquely*. Although a value g^x may be retransmitted by the adversary, and the long-term value g^a will be retransmitted to make it available to new

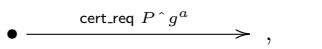
communication partners, all occurrences trace back ultimately to a single original point at which the exponent was chosen, and the exponentiated value first transmitted. Whenever any party later transmits this same value, he has first received it in some incoming message.

The adversary. An adversary who has observed g^x, g^y but not x, y is negligibly likely to be able to produce g^{xy}, or indeed to do better than chance in distinguishing it from a randomly chosen g^z. These are called the *Computational* and (resp.) *Decisional* Diffie-Hellman Assumptions.

We identify a protocol role for certificate authorities also. It receives a request containing the principal name P and the public value Y_P, and may then emit a public key certificate c_P.

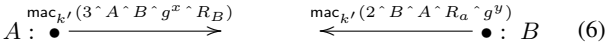
It does so after some procedure we will not represent, which is intended to ensure that the principal P possesses an a such that $g^a = Y$. The same Y should never be certified with two different principals P, P', lest a participant A not know whether a particular session key was shared with P or P'. For instance, if a priest had the same public value Y as a district attorney, a confession meant for one might be received by the other. However, if each principal chooses his long term secret at random from a set far larger than the set of principals, then this event is unlikely.

To request certification, a compliant principal transmits:



having made sure to choose a freshly generated long term secret a. Thus, whenever we have a certification request from a compliant principal, we will assume that a is non-originating, and that g^a originates uniquely at this transmission. The CA's (unrepresented) procedure is intended to ensure P has met these conditions, before the CA emits the certificate.

Even if a CA cannot check for all collisions, we can still avoid the bad consequences of colliding certificates by altering the protocol. We could include an additional step of key confirmation, using a different hash function H' to generate a confirmation key $k' = H'({Y_A}^b \hat{} {R_A}^y)$. By exchanging messages containing a MAC of the ephemeral values and principal identities, each principal can ensure that no confusion has occurred.



(6)

Instead of the key confirmation messages of Eqn. 6, one can also diversify the session key using the intended principal identities:

$$k = H(A \hat{} B \hat{} {Y_B}^a \hat{} {R_B}^x) = H(A \hat{} B \hat{} {Y_A}^b \hat{} {R_A}^y).$$

This key computation prevents the district attorney (if complying with the protocol) from receiving a message intended for the priest.

The CA can also ascertain whether Y is a genuine group element, i.e. whether there exists an a such that $Y = g^a$, and in particular that $Y \neq g^1$.

Certificate Authority. A certificate is useful only if certificates cannot be constructed by the adversary, i.e. the Certificate Authority's signature key is uncompromised. We model this as meaning that the key is non-originating.

Moreover, a certificate $c_P = [\![\mathsf{cert}\ Y \hat{} P]\!]_{CA}$ tells the recipient something. We interpret it as providing the information that a principal can use in deciding whether to believe that P's long-term secret is uncompromised, i.e. that it will be used only in accordance with the protocol. A principal may decide to infer that, for all a, if $Y = g^a$, then a is non-originating.

We also assume that the CA ensures (1) $\exists a \,.\, Y = g^a$ and $a \neq 1$, and (2) if both $c_P = [\![\mathsf{cert}\ Y \hat{} P]\!]_{CA}$ and $c_{P'} = [\![\mathsf{cert}\ Y \hat{} P']\!]_{CA}$ have been generated for the same Y, then $P = P'$.

Principal Certification. When a principal P (successfully) requests a certificate with long term public value Y, we will assume that its request is the event at which Y uniquely originates. We will also assume that Y is of the form g^a, i.e. that P has chosen a secret value a independent of other values chosen elsewhere, and used it to form the public value Y.

In this section, we will consider the protocol without key confirmation or diversification, and we intend to sketch a proof that—assuming no CA ever re-certifies the same Y with two principals—the protocol works as advertised.

In particular, an adversary cannot obtain a key shared with a compliant A or B, who has engaged in a session apparently with a B or A, if the latter's long term secret is used only in accordance with the protocol. Moreover, if two compliant principals A, B obtain the same key k, then each believes it to be shared with the other. We regard the first of these statements as a confidentiality property, and the second as an authentication property.

2.1 Confidentiality for UM

The confidentiality property states that the adversary cannot obtain a session key in a session satisfying reasonable assumptions.

Suppose that B has engaged in a full run of UM as responder. Moreover, suppose that the other parameters of the run are:

$$A, R_A, y, Y_A, b, CA$$

Assume $b, y, \mathsf{sk}(CA)$ are non-originating and g^y is uniquely originating, and every a such that $Y_A = g^a$ is non-originating. Then there is no node on which $k_B = H({Y_A}^b \hat{} {R_A}^y)$ is transmitted.

Proof sketch. If there were such a node n, it would be either a regular node or an adversary node. No regular node, however, transmits a value of this form. Thus, n would be an adversary node. However, in that case, the adversary must have constructed $H({Y_A}^b \hat{} {R_A}^y)$ from an earlier transmission of ${Y_A}^b \hat{} {R_A}^y$, which in turn must have been constructed from earlier transmissions of ${Y_A}^b$ and ${R_A}^y$.

1. No regular node can transmit ${Y_A}^b$.

In particular, Y_A having been certified, we have from CA assumption (1) that $Y_A \neq g^1$, hence ${Y_A}^b \neq Y_B$. Moreover, since $Y_B = g^b$ was certified, $b \neq 1$, whence ${Y_A}^b \neq Y_A$. Thus ${Y_A}^b$ was not transmitted on the certification nodes for A, Y_A or B, Y_B.

Moreover, ${Y_A}^b$ was not transmitted on any other certification request by a compliant principal. A compliant principal selects a value g^c, but we know that Y_A is of the form g^a, so that ${Y_A}^b = g^{ab}$, hence distinct from any g^c at all.[2] Nor, for the same reason, does ${Y_A}^b$ originate as the ephemeral value of any compliant strand.

[2] We formalize this below, Section 4. However, a, b being choices made independently of each other and of c, betting on $ab = c$ is a strategy that loses for the adversary with overwhelming probability.

2. No adversary action can produce ${Y_A}^b$. Although the adversary has the values $Y_A = g^a$ and $Y_B = g^b$, in order to incorporate b into the exponent of g^a, or in order to incorporate a into the exponent of g^b, the adversary would have to have access to a (non-originating) value a or b. □

Step 2 of the argument uses a central principle, which we will formalize in a stronger form in Section 5. For now, we will codify it as:

Principle 1 *If the adversary originates g^e, then $e = e_1 e_2$ where both g^{e_1} and e_2 have been previously transmitted.*

Principle 1 also leads to a result when A has engaged in a full run of UM as initiator similar to this result about B as responder.

2.2 Authentication for UM

The authentication goal for UM states that, on reasonable assumptions, if two compliant principals agree on a key, then they agree on their identities.

Suppose that A has engaged in a full run of UM as initiator. Moreover, suppose that the other parameters of the run are:

$$B, x, R_B, a, Y_B, CA$$

Assume $a, x, \mathsf{sk}(CA)$ are non-originating and g^x is uniquely originating, and every b such that $Y_B = g^b$ is non-originating. If there is another neutral node n in which a key record $\mathsf{keyrec}\ P \hat{}\ P' \hat{}\ k$ is deposited, for $k = H({Y_B}^a \hat{}\ {R_B}^x)$, then $P = B$ and $P' = A$.

In this case, n may be either a responder event or else an initiator event. The protocol permits two principals, each of whom starts a session as initiator, to successfully complete a run with each other. However, if one executes a local session as responder, the other must have acted as initiator.

Why does our authentication property hold? Suppose that the parameters of a responder session terminating in node n are:

$$P, P', R, b, Y, z, CA$$

(The case of an initiator session is similar.) Observe that ${Y_B}^a = Y^b$, and $R_B^x = R^z$. !!!

3 FORMALIZING STRANDS AND PROTOCOLS

In this section, we formalize the strand space theory (Section 3.1), and provide a pattern for defining particular protocols within it (Section 3.2). In the next sections, we use this mechanism to represent Diffie-Hellman style protocols, and the algebra of messages that they manipulate.

3.1 A Theory of Strands and Bundles

The strand space theory [13] identifies local sessions of the regular (uncompromised) principals as *strands*, i.e. bounded linear sequences of events belonging to a single local session of the protocol. Each event is either a message transmission, a message reception, or a "neutral," purely local event. We regard a neutral event as an operation against a local database of facts, and we use neutral events to allow a strand to manipulate long term secrets, recently agreed session keys, and public key certificates [6]. The basic actions of the adversary are also represented as strands, including generating new values, and applying an operation such as encryption, decryption, or exponentiation to available values.

We write $n_1 \Rightarrow n_2$ when n_1 immediately precedes n_2 on the same strand, using $\Rightarrow^+$ and $\Rightarrow^*$ for the transitive and reflexive-transitive closures of $\Rightarrow$. In this formalization, we do not make strands first-class objects, but axiomatize the nodes and the relation $\Rightarrow$.

An execution is a *bundle*, i.e. a partially ordered collection of strands (or initial segments of strands), where the partial ordering $\preceq$ reflects causality. If one node n_1 lies another node n_2 on the same strand $n_1 \Rightarrow^* n_2$, then certainly $n_1 \preceq n_2$. The key bundle property is that, for every reception node n_2, the message received on n_2 was transmitted on some transmission node n_1 with $n_1 \prec n_2$. Moreover, in a bundle, the precedence ordering $\preceq$ is *well-founded*.

Where the formulas contain the free variables $\vec{x}$, we write

$$\varphi_1, \ldots, \varphi_i \ \vdash\ \psi_1, \ldots, \psi_j,$$

called a *sequent*, to mean

$$\forall \vec{x}.\ (\varphi_1 \wedge \ldots \wedge \varphi_i) \supset (\psi_1 \vee \ldots \vee \psi_j).$$

As in Gentzen's sequent calculus [5], the comma means conjunction on the left but disjunction on the right. Free variables are implicitly universally quantified with the whole sequent as scope. When the right-hand side is empty, i.e. $j = 0$, it represents the unsatisfiable empty disjunction $\bot$. Thus, $\varphi_1, \ldots, \varphi_i \ \vdash\ \bot$ means the universal closure $\forall \vec{x}.\ \neg(\varphi_1 \wedge \ldots \wedge \varphi_i)$.

Str 1. $n_0 \Rightarrow n_2,\ n_1 \Rightarrow n_2 \ \vdash\ n_0 = n_1$

Str 2. $n_0 \Rightarrow n_1,\ n_0 \Rightarrow n_2 \ \vdash\ n_1 = n_2$

Str 3. $n_0 \Rightarrow n_0 \ \vdash\ \bot$

Str 4. $n_0 \Rightarrow n_1 \ \vdash\ \mathsf{node}(n_0) \wedge \mathsf{node}(n_1)$

Str 5. $\mathsf{xmit}(n_0) \ \vdash\ \mathsf{node}(n_0)$

Str 6. $\mathsf{recv}(n_0) \ \vdash\ \mathsf{node}(n_0)$

Str 7. $\mathsf{node}(n_0) \ \vdash\ \mathsf{xmit}(n_0),\ \mathsf{recv}(n_0),\ \mathsf{neutral}(n_0)$

Str 8. $\mathsf{reg}(n_0),\ \mathsf{adv}(n_0) \ \vdash\ \bot$

Str 9. $\mathsf{reg}(n_0) \ \vdash\ \mathsf{node}(n_0)$

Str 10. $\mathsf{adv}(n_0) \ \vdash\ \mathsf{node}(n_0)$

Str 11. $\mathsf{node}(n_0) \ \vdash\ \mathsf{reg}(n_0),\ \mathsf{adv}(n_0)$

Str 12. $n_0 \Rightarrow n_1,\ \mathsf{adv}(n_0) \ \vdash\ \mathsf{adv}(n_1)$

Str 13. $n_0 \Rightarrow n_1,\ \mathsf{reg}(n_0) \ \vdash\ \mathsf{reg}(n_1)$

Str 14. $n_0 \Rightarrow^* n_1 \ \vdash\ n_0 \preceq n_1$

Str 15. $n_0 \preceq n_1,\ n_1 \preceq n_2 \ \vdash\ n_0 \preceq n_2$

Str 16. $n_0 \preceq n_1,\ n_1 \preceq n_0 \ \vdash\ n_0 = n_1$

The crucial property of a bundle, or possible execution, is that everything received was previously sent. We use $\mathsf{msg}(n)$ to refer to the message sent or received on node n. We list this as its own axiom group.

Bnd $\mathsf{recv}(n_1) \vdash \exists n_0 \,.\, n_0 \preceq n_1 \wedge \mathsf{xmit}(n_0) \wedge \mathsf{msg}(n_1) = \mathsf{msg}(n_0)$

We also express the well-foundedness principle as its own axiom group. In this case, we have an axiom schema, i.e. a set of axioms containing one for each formula φ, called the "predicate of induction" for that instance. Unlike the other axioms we use, the left-hand side of well-foundedness is not positive-existential, even for simple φ.

WF$_\varphi$ $\forall n_1 \,.\, (\forall n_0 \,.\, n_0 \preceq n_1 \supset \varphi(n_0)) \supset \varphi(n_1) \vdash \forall n_1 \,.\, \varphi(n_1)$

3.2 A Pattern for Defining Protocols

A protocol determines a finite number of (parametric) strands, which we call the *roles* of the protocol. Each of these roles may be instantiated by substituting values from the algebra of messages in place of the parameters, in a type-respecting way. The regular behaviors compliant with the protocol are precisely these substitution-instances.

Thus, we regard the regular nodes as determined by two kinds of predicates. Each predicate of the first kind is called a *role-position* predicate. If $R_{\rho,i}(n)$ is a role-position predicate, it states that n is a node lying at the i^{th} position on an instance of the role ρ. Each role-position predicate concerns a fixed position along a fixed role. For a protocol Π, with each role ρ of length $|\rho|$, there will be $\sum_{\rho\in\Pi} |\rho|$ role-position predicates.

The predicates of the second kind are called *parameter* predicates. Each parameter predicate $P(n, v)$ relates a node n to the parameter value chosen for some parameter that has already appeared in or prior to n. For instance, $\mathsf{self}(n, a)$ may say that a is the name of the principal executing the strand containing node n, assuming that a's name helps to determine some message transmitted or received on this strand up to node n. Likewise, $\mathsf{mynonce}(n, v)$ may say that v is the value chosen along this strand to serve as a nonce. A parameter predicate $P(n, v)$ acts like a partial function: There is at most one v such that $P(n, v)$ holds, for any n. There may be no v such that $P(n, v)$, if this parameter is irrelevant to the strand n lies on, or if this parameter is chosen only later on a some node n' such that $n \Rightarrow^+ n'$. However, once determined it remains the same for later nodes.

When giving the predicates $R_1(n), \ldots, R_j(n)$ as the role position predicates for a role ρ, we are stipulating that the following axioms hold for $1 \leq i < j$ and $1 \leq k \leq j$:

RolePos 1. $R_1(n_1),\ n_0 \Rightarrow n_1 \vdash \bot$

RolePos 2. $R_{i+1}(n_1) \vdash \exists n_0 \,.\, R_i(n_0) \wedge n_0 \Rightarrow n_1$

RolePos 3. $R_k(n_1) \vdash \mathsf{node}(n_1)$

If a predicate $P(n, v)$ is declared to be a parameter predicate, then we stipulate:

Param 1. $P(n, v) \vdash \mathsf{node}(n)$

Param 2. $P(n, v) \vdash \mathsf{isMsg}(v)$

Param 3. $P(n, v),\ P(n, v') \vdash v = v'$

Param 4. $P(n, v),\ n \Rightarrow n_1 \vdash P(n_1, v)$

We use this format to characterize both the regular nodes of a protocol and also the adversary strands. To say that a parameter is of a particular type, expressed by the type τ, means:

ParamType$_\tau$ 1. $P(n, v) \vdash \exists y\colon \tau \,.\, v = y$

We illustrate these ideas with the one role that we include as a regular role in every protocol. We call this role the *listener role*. It contains a single reception node, which receives a message x of any form. We use instances of the listener role to witness for the fact that the message x has become available to the adversary. We use the role position predicate $\mathsf{lsn1}(n)$ to say that n is the first node on an instance of the listener role, and the parameter predicate $\mathsf{hear}(n, x)$ to say that the value received on n is x. The role position axioms then assert:

LsnPos 1. $\mathsf{lsn1}(n_1) \vdash \mathsf{node}(n_1)$

LsnPos 2. $\mathsf{lsn1}(n_1),\ n_0 \Rightarrow n_1 \vdash \bot$

The role parameter axioms for hear become:

Hear 1. $\mathsf{hear}(n, v) \vdash \mathsf{node}(n)$

Hear 2. $\mathsf{hear}(n, v) \vdash \mathsf{isMsg}(v)$

Hear 3. $\mathsf{hear}(n, v),\ \mathsf{hear}(n, v') \vdash v = v'$

Hear 4. $\mathsf{hear}(n, v),\ n \Rightarrow n_1 \vdash \mathsf{hear}(n_1, v)$

Since the listener node is a regular reception node, and its parameter is the message received, we have:

Lsn 1. $\mathsf{lsn1}(n) \vdash \mathsf{recv}(n) \wedge \mathsf{reg}(n)$

Lsn 2. $\mathsf{lsn1}(n),\ \mathsf{hear}(n, x) \vdash \mathsf{msg}(n) = x$

Lsn 3. $\mathsf{lsn1}(n),\ \mathsf{msg}(n) = x \vdash \mathsf{hear}(n, x)$

3.2.1 An Example: Regular Nodes of the Variant Unified Model

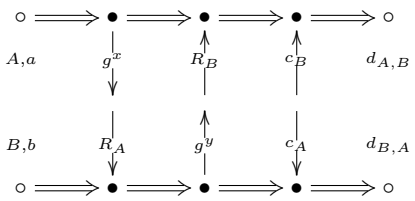
From this example it is clear that we can determine all of the axioms governing a role by giving a table containing the role position predicates, the parameter predicates, the directions of the successive nodes, and the message involved. We illustrate this style of protocol specification with the variant Unified Model VUM. We regard the VUM protocol as involving several events that were left implicit in the introduction. In particular, each role begins with a neutral node that consults the local state of its principal to obtain the principal's name and long term secret DH value. Then the ephemeral values R_A is constructed and transmitted; the ephemeral value R_B is received; and a certificate for the partner's long term public DH value is received. Finally, in a neutral node, the principals' names and the resulting key are deposited as a new record into the principal's local state database.

We summarize this as a strand space figure in Fig. 2. We write bullets • for transmission and reception nodes, with the direction of the arrow indicating which, and circles ∘ for neutral nodes. The top row shows the initiator's actions, and the bottom row the responder's actions.

3.2.2 An Example: Regular Nodes of Needham-Schroeder

As an example, for the familiar Needham-Schroeder protocol, we could use the role-position predicates

$$\mathsf{nsInit1}(n), \mathsf{nsInit2}(n), \mathsf{nsInit3}(n)$$



$c_P = [\![\mathsf{cert}\ Y_P \hat{}\ P]\!]_{CA} \qquad d_{P_1,P_2} = \mathsf{keyrec}\ P_1 \hat{}\ P_2 \hat{}\ K_{P_1}$

Figure 2: The VUM Protocol, with $K_A = {Y_B}^a \cdot {R_B}^x$ and $K_B = {Y_A}^b \cdot {R_A}^y$

Role pos.	Dir.	Params.	Msg.
nsInit1	xmit	$\mathsf{self}(n,a),\ \mathsf{peer}(n,b),\ \mathsf{mynonce}(n,v_1)$	$\{\!\lvert a \hat{}\ v_1 \rvert\!\}_{\mathsf{pk}(b)}$
nsInit2	recv	$\mathsf{yrnonce}(n,v_2)$	$\{\!\lvert v_1 \hat{}\ v_2 \rvert\!\}_{\mathsf{pk}(a)}$
nsInit3	xmit		$\{\!\lvert v_2 \rvert\!\}_{\mathsf{pk}(b)}$

Figure 3: Role Position and Parameter Predicates for NS Initiator

for the initiator role, and for the responder role

$$\mathsf{nsResp1}(n), \mathsf{nsResp2}(n), \mathsf{nsResp3}(n).$$

The parameter predicates $\mathsf{self}(n,a), \mathsf{peer}(n,a), \mathsf{mynonce}(n,v), \mathsf{yrnonce}(n,v)$ can be used to express the parameters. They respectively express the identity of the principal executing a strand; the identity of its intended partner; the nonce that principal chooses; and the nonce apparently chosen by its partner.

We can then express the regular behaviors permitted by the protocol in a table such as Figure 3. It presents, on its lines, the role position predicates, direction, parameter predicates, and message form for the successive nodes. The logical content of this table is expressed in a number of axioms. Instances of Axioms **RolePos** 1 and **RolePos** 2 are generated from the order of the lines. Axioms like **NS Init** 1 are generated from the direction indicator. The role parameters are determined by collecting all those up to a given line; each line shows only the newly applicable parameter predicates. These parameter predicates and the message entry determine the remaining axioms such as **NS Init** 2 and **NS Init** 3. The same process yields similar axioms for the remaining lines.

NS Init 1. $\mathsf{nsInit1}(n) \vdash \mathsf{xmit}(n) \wedge \mathsf{reg}(n)$

NS Init 2. $\mathsf{nsInit1}(n),\ \mathsf{self}(n,a),\ \mathsf{peer}(n,b),\ \mathsf{mynonce}(n,v_1) \vdash \mathsf{msg}(n) = \{\!\lvert a \hat{}\ v_1 \rvert\!\}_{\mathsf{pk}(b)}$

NS Init 3. $\mathsf{nsInit1}(n),\ \mathsf{msg}(n) = \{\!\lvert a \hat{}\ v_1 \rvert\!\}_{\mathsf{pk}(b)} \vdash \mathsf{self}(n,a),\ \mathsf{peer}(n,b),\ \mathsf{mynonce}(n,v_1)$

NS Init 4. $\mathsf{nsInit2}(n) \vdash \mathsf{recv}(n) \wedge \mathsf{reg}(n)$

NS Init 5. $\mathsf{nsInit2}(n),\ \mathsf{self}(n,a),\ \mathsf{mynonce}(n,v_1),\ \mathsf{yrnonce}(n,v_2) \vdash \mathsf{msg}(n) = \{\!\lvert v_1 \hat{}\ v_2 \rvert\!\}_{\mathsf{pk}(a)}$

NS Init 6. $\mathsf{nsInit2}(n),\ \mathsf{msg}(n) = \{\!\lvert v_1 \hat{}\ v_2 \rvert\!\}_{\mathsf{pk}(a)} \vdash \mathsf{self}(n,a),\ \mathsf{mynonce}(n,v_1),\ \mathsf{yrnonce}(n,v_2)$

NS Init 7. $\mathsf{nsInit3}(n) \vdash \mathsf{xmit}(n) \wedge \mathsf{reg}(n)$

Role pos.	Dir.	Params.	Msg.
		$\mathsf{self}(n,a),\ \mathsf{peer}(n,b),$	
nsResp1	recv	$\mathsf{yrnonce}(n,v_1)$	$\{\!\lvert a \hat{}\ v_1 \rvert\!\}_{\mathsf{pk}(b)}$
nsResp2	xmit	$\mathsf{mynonce}(n,v_2)$	$\{\!\lvert v_1 \hat{}\ v_2 \rvert\!\}_{\mathsf{pk}(a)}$
nsResp3	recv		$\{\!\lvert v_2 \rvert\!\}_{\mathsf{pk}(b)}$

Figure 4: Role Position and Parameter Predicates for NS Responder

NS Init 8. $\mathsf{nsInit3}(n),\ \mathsf{peer}(n,b),\ \mathsf{yrnonce}(n,v_2) \vdash \mathsf{msg}(n) = \{\!\lvert v_2 \rvert\!\}_{\mathsf{pk}(b)}$

NS Init 9. $\mathsf{nsInit3}(n),\ \mathsf{msg}(n) = \{\!\lvert v_2 \rvert\!\}_{\mathsf{pk}(b)} \vdash \mathsf{peer}(n,b),\ \mathsf{yrnonce}(n,v_2)$

The axioms for the responder role are symmetric; we interchange xmit with recv, self with peer, and mynonce with yrnonce. These axioms are generated in accordance with Fig. 4.

NS Resp 1. $\mathsf{nsResp1}(n) \vdash \mathsf{recv}(n) \wedge \mathsf{reg}(n)$

NS Resp 2. $\mathsf{nsResp1}(n),\ \mathsf{peer}(n,a),\ \mathsf{self}(n,b),\ \mathsf{yrnonce}(n,v_1) \vdash \mathsf{msg}(n) = \{\!\lvert a \hat{}\ v_1 \rvert\!\}_{\mathsf{pk}(b)}$

NS Resp 3. $\mathsf{nsResp1}(n),\ \mathsf{msg}(n) = \{\!\lvert a \hat{}\ v_1 \rvert\!\}_{\mathsf{pk}(b)} \vdash \mathsf{peer}(n,a),\ \mathsf{self}(n,b),\ \mathsf{yrnonce}(n,v_1)$

NS Resp 4. $\mathsf{nsResp2}(n) \vdash \mathsf{xmit}(n) \wedge \mathsf{reg}(n)$

NS Resp 5. $\mathsf{nsResp2}(n),\ \mathsf{peer}(n,a),\ \mathsf{yrnonce}(n,v_1),\ \mathsf{mynonce}(n,v_2) \vdash \mathsf{msg}(n) = \{\!\lvert v_1 \hat{}\ v_2 \rvert\!\}_{\mathsf{pk}(a)}$

NS Resp 6. $\mathsf{nsResp2}(n),\ \mathsf{msg}(n) = \{\!\lvert v_1 \hat{}\ v_2 \rvert\!\}_{\mathsf{pk}(a)} \vdash \mathsf{peer}(n,a),\ \mathsf{yrnonce}(n,v_1),\ \mathsf{mynonce}(n,v_2)$

NS Resp 7. $\mathsf{nsResp3}(n) \vdash \mathsf{recv}(n) \wedge \mathsf{reg}(n)$

NS Resp 8. $\mathsf{nsResp3}(n),\ \mathsf{self}(n,b),\ \mathsf{mynonce}(n,v_2) \vdash \mathsf{msg}(n) = \{\!\lvert v_2 \rvert\!\}_{\mathsf{pk}(b)}$

NS Resp 9. $\mathsf{nsResp3}(n),\ \mathsf{msg}(n) = \{\!\lvert v_2 \rvert\!\}_{\mathsf{pk}(b)} \vdash \mathsf{self}(n,b),\ \mathsf{mynonce}(n,v_2)$

We also state that these regular transmission and reception nodes are all those of NS.

NS All 1. $\mathsf{recv}(n),\ \mathsf{reg}(n) \vdash \mathsf{nsInit2}(n),\ \mathsf{nsResp1}(n),\ \mathsf{nsResp3}(n)$

NS All 2. $\mathsf{xmit}(n),\ \mathsf{reg}(n) \vdash \mathsf{nsInit1}(n),\ \mathsf{nsInit3}(n),\ \mathsf{nsResp2}(n)$

3.2.3 Adversary Nodes

4 A THEORY OF EXPONENTS

[@@ Here's a strange notational subtlety. Our base group G, since finite cyclic, is determined once we choose our q. So it is isomorphic to $\mathbf{Z}_q$. But then the space of exponents is $\mathbf{Z}_q^*$ (viewed as a field). But the latter is NOT taken as subset of G. So it would be confusing to refer to G as $\mathbf{Z}_q$. Since we want to talk about the various G uniformly I'll write G_q below... not clear that's the best notation...]

[@@ need a decent typographic convention for G terms vs E terms. Right now using uppercase for G and lowercase for E]

A little discussion about the formalization. The operations of the signature embody the computational power that we model for the adversary. So we don't have a log function, for example. Capturing adversary ability by writing terms is what the symbolic model is all about, the twist here is that terms are considered modulo some equations, complicating syntactic analysis.

A crucial point is that we want to model what the adversary can do *uniformly* over all values of q. For example, each particular choice of q leads to the set of exponents comprising a field (typically $\mathbf{Z}_q^*$, but this will not matter to our analysis) so we include equations defining fields in our axioms. It is also true that each choice q determines a field with characteristic q, so that the equation 1 + ... + 1 = 0 holds. But for an attack to be uniform it cannot rely on any one of these equations holding, so of course we exclude them from our axioms.

We work over a signature comprising two sorts, G and E, and operations

$$\begin{aligned} \odot &: G \times G \to G \\ id &: G \\ +, * &: E \times E \to E \\ -, (\cdot)^{-1} &: E \to E \\ 0, 1 &: E \\ exp &: G \times E \to G \end{aligned}$$

We use two sorts of variables, which we refer to as *G-variables* and *E-variables* respectively; in the natural way we then build *G-terms* and *E-terms.* As usual we write xy for $x * y$ and we write b^x for $exp(b, x)$.

Definition 1 *An equation $A = B$ between G-terms is* uniformly valid for finite fields *[ugly phrase; do better?] if it holds in every G_q*

That's what we care about.

The conditional equational theory DH below expresses the facts that $(G, \odot, id)$ is an abelian group $(E, +, -, *, /, 0, 1)$ is a field, and that exponentiation of G by E behaves as expected.

[Which equations to present, exactly? Mostly just a matter of taste. Discuss.]

$(G, \odot, i, id)$ is an abelian group

$$(a \odot b) \odot c = a \odot (b \odot c) \quad (7)$$
$$a \odot b = b \odot a \quad (8)$$
$$b \odot id = b \quad (9)$$
$$b \odot i(b) = id \quad (10)$$

$(E, +, -, *, (\cdot)^{-1}, 0, 1)$ is a field

$$(x + y) + z = x + (y + z) \quad (11)$$
$$x + y = y + x \quad (12)$$
$$x + 0 = x \quad (13)$$
$$x + (-x) = 0 \quad (14)$$
$$(x \cdot y) \cdot z = x \cdot (y \cdot z) \quad (15)$$
$$x \cdot y = y \cdot x \quad (16)$$
$$x * (y + z) = (x * y) + (x * z) \quad (17)$$
$$x * 1 = x \quad (18)$$
$$x \neq 0 \to \quad x * x^{-1} = 1 \quad (19)$$

exponentiation behaves properly

$$a^x \odot a^y = a^{(x+y)} \quad (20)$$
$$(a^x)^y = a^{x*y} \quad (21)$$
$$a^0 = id \quad (22)$$
$$id^x = id \quad (23)$$

these follow from above but

will need them for reduction

$$-(0) = 0 \quad (24)$$
$$-(x + y) = -x + -y \quad (25)$$
$$- - x = x \quad (26)$$
$$x * 0 = 0 \quad (27)$$
$$x * -(y) = -(x * y) \quad (28)$$
$$(29)$$

It is clear that each equation above is uniformly true for finite fields.

Let R be the set of rewrite rules obtained by orienting the equations above from left to right. Let AC be the set of equations expressing the associativity and commutativity of $\odot$, +, and *. The rewrite relation $\to_{R/AC}$ is rewriting with R modulo AC.

Say that a term t is *R-irreducible* if there does not exist a term t' with $t \to t'$. (Note: let's not speak of "normal forms" since that connotes uniqueness)

Lemma 1 *The rewrite system R is terminating; that is, every sequence of R-reductions is finite.*

Proof. (easy, todo) □

Lemma 2 *The G-terms irreducible with respect to R are of the form*

$$b_1^{e_1} \odot \cdots \odot b_n^{e_n}$$

where each b_i is a G-variable, the b_i are distinct, and each e_i is an irreducible E-term. (We consider id to be the empty product.)

The E-terms irreducible with respect to R are of the form $f_1 + \cdots + f_k$ (taking 0 to be the empty sum) where each f_i is of the form

$$x_1^{d_1} * \cdots * x_n^{d_n}$$

with the x_i distinct and each $d_i = \pm 1$

Proof. Here we slog through the rules... □

The following observation will be useful in what follows. For the notion of ultraproduct of structures see, for example, [**?**].

Lemma 3 *Let s and t be terms built from E-variables using $+, *, -, (\cdot)^{-1}, 0, 1$. If the equation $s = t$ holds in each $\mathbf{Z}_q^*$ then it holds in the field $\mathbf{Q}$ of rational numbers.*

Proof. Let D be a nonprincipal ultraproduct over the natural numbers, and define $\mathbf{F}_* = \prod_D \{\mathbf{Z}_q^* \mid q \text{ prime. }\}$ Then $\mathbf{F}_*$ is a field. If $s = t$ holds in each $\mathbf{Z}_q^*$ then it holds in $\mathbf{F}_*$. But it is easy to see that $\mathbf{F}_*$ has characteristic 0 and so the prime field of $\mathbf{F}_*$ is $\mathbf{Q}$. Since $\mathbf{Q}$ is a subfield of $\mathbf{F}_*$, $s = t$ holds there. □

In fact it follows from results of of Ax [**?**] that the set of first-order sentences true in all $\mathbf{Z}_q^*$ coincides with the first-order theory of $\mathbf{F}_*$. For this paper we require only the simple observation above.

Definition 2 *Let v be an E-variable. If e is an irreducible E-term, the* multiplicity $\mathsf{mult}(e, v)$ *of v in e is the number of occurrences of v in e.*

If $t = b_1^{e_1} \cdots b_n^{e_n}$ is an irreducible G-term, the b_i-multiplicity of v in t is the multiplicity of v in e_i.

Note that this is well-defined modulo the associativity and commutativity of $+$ and $*$.

We want to show that these multiplicities are an invariant of terms considered modulo uniform validity.

Note that we can't expect the converse, that the multiplicities characterize a term, at least not without a much more subtle notion of multiplicity. For example, we would have to distinguish between "positive" and "negative" multiplicities, in order to be able to distinguish $\frac{x}{y}$ from $\frac{y}{x}$. (and even then consider, eg $\frac{x}{y} + \frac{x}{z}$ vs $\frac{xx}{y} + \frac{1}{z}$)

Lemma 4 *If $A = a_1^{e'_1} \cdots a_n^{e'_n}$ and $B = b_1^{e_1} \cdots b_m^{e_m}$ are irreducible G-terms equal in each G_q then $m = n$ and the b_i and c_j are in one-to-one correspondence via permutation π on $[1..n]$ such that the each equation $e_i = e'_{\pi(i)}$ is true on every $\mathbf{Z}_q^*$.*

Proof. sketch: By negating exponents, reduce to showing that an equation

$$c_1^{e_1} \cdots c_n^{e_n} = 1$$

holds in each G_q if and only if each e_i is identically 0 on each $\mathbf{Z}_q^*$. By selectively setting c_i's to 1, that reduces to showing that if any e is not identically 0 then we can find a q and an instantiation such that $e \neq 0$ in $\mathbf{Z}_q^*$... Use Lemma 3 □

Lemma 5 *If the E-equation $s = t$ holds in $\mathbf{Z}_q^*$ for each prime q then for each v, s and t have the same v-multiplicity.*

Proof. By Lemma 3 it suffices to show that if s and t have different multiplicity for some variable v then there is an assignment of values in $\mathbf{Q}$ to the variables of s and t yielding different results. Let v be such a variable. Instantiate every other variable to the value 1. We are left with an equation $s' = t'$ in which each of s' and t' is a sum of positive and negative powers of v. Since the multiplicity of v differs in s' and t' it certainly true that s' and t' are not identical as rational functions; so it is clear that some value for v will distinguish them. □

[Note to Joshua: I observed above that "same multiplicities" is not sufficient to characterize equality of terms. But the above argument was so crude, left so much room for discrimination, that it suggests that there much more we can say about equal terms than just same multiplicities. Later...]

Theorem 6 *[this is clumsily stated, but I'm going to bed now:] Let t be an arbitrary G-term. Every R-irreducible form t_0 of t has the same multiplicity function. If we then define the multiplicity function for an arbitrary term to be the multiplicity function for any of its irreducible forms, then this function is invariant for terms that are equal in all G_q.*

Proof. corollary of all the previous, together with the fact that the equations supporting the rewriting are all true in each $\mathbf{G}_q$ □

Now for the defn of influences. [Or maybe "depends on"? Which do you like better?]

Definition 3 *Let v be a (G- or E-) variable and let t be a term. Then t* depends on *v if the multiplicity of v in t is non-zero.*

The following, essentially a corollary of Theorem 6, is justification for our use of the "influences" relation to constrain adversary behaviors.

Theorem 7 *Let T be a G-term. For any G and G-instantiation η of the variables of T in G, the value of T under η depends only on the variables that influence T.*

Now we can articulate principles like

$(x$ non-orig $) \wedge (g^x$ unique-orig at $n) \wedge (x$ influences $msg(m))$
$\vdash (n \preceq m)$

5 A LIMITATION OF THE ADVERSARY

6 COMPLETING THE MQV ANALYSIS

REFERENCES

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

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

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

[4] S. Escobar, C. Meadows, and J. Meseguer. Maude-npa: Cryptographic protocol analysis modulo equational properties. *Foundations of Security Analysis and Design V*, pages 1–50, 2009.

[5] G. Gentzen. Investigations into logical deduction (1935). In *The Collected Works of Gerhard Gentzen*. North Holland, 1969.

[6] Joshua D. Guttman. Fair exchange in strand spaces. In M. Boreale and S. Kremer, editors, *SecCo: 7th International Workshop on Security Issues in Concurrency*, EPTCS. Electronic Proceedings in Theoretical Computer Science, Sep 2009.

[7] 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.

[8] Deepak Kapur, Paliath Narendran, and Lida Wang. An E-unification algorithm for analyzing protocols that use modular exponentiation. *RewritingTechniques and Applications*, pages 150–150, 2003.

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

[10] 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.

[11] 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.

[12] A. Menezes, University of Waterloo. Dept. of Combinatorics, Optimization, and University of Waterloo. Faculty of Mathematics. *Another look at HMQV*. Citeseer, 2005.

[13] 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.