The Faithfulness of Abstract Protocol Analysis: Message Authentication*

Joshua D. Guttman F. Javier Thayer Lenore D. Zuck

December 20, 2002

Abstract

Dolev and Yao initiated an approach to studying cryptographic protocols which abstracts from possible problems with the cryptography so as to focus on the structural aspects of the protocol. Recent work in this framework has developed easily applicable methods to determine many security properties of protocols. A separate line of work, initiated by Bellare and Rogaway, analyzes the way specific cryptographic primitives are used in protocols. It gives asymptotic bounds on the risk of failures of secrecy or authentication.

In this paper we show how the Dolev-Yao model may be used for protocol analysis, while a further analysis gives a quantitative bound on the extent to which real cryptographic primitives may diverge from the idealized model. We illustrate this method where the cryptographic primitives are based on Carter-Wegman universal classes of hash functions. This choice allows us to give specific quantitative bounds rather than simply asymptotic bounds.

1 Introduction

Cryptographic protocols are sequences of messages that use cryptography to achieve security goals such as authentication and establishing new shared secrets. Despite frequently being simple, they are often wrong, sometimes disastrously. Much work (including [7, 21, 22, 19, 29, 24, 31, 17]) has been done to develop methods to ensure their correctness, starting with Dolev and Yao [12], who represent encryption as a free operator on terms, and abstract

*This work supported by the National Security Agency. Author's affiliations: The MITRE Corporation, Bedford MA, USA, and also (for L. Zuck) New York University. Authors' email addresses: `guttman,jt@mitre.org`; `zuck@cs.nyu.edu`

from the properties of particular cryptographic primitives. If an attack succeeds against a protocol assuming this abstract cryptography, then the same attack will also succeed when the protocol is implemented with real cryptographic primitives. By contrast, a proof that there are no attacks, based on the assumption of abstract cryptography, will no longer be valid when concrete primitives are selected. Possibly an adversary can manipulate the details of the cryptography to create attacks that would not succeed against abstract encryption.

Goals of this paper One form of the Dolev-Yao approach, the strand space theory, has now developed convenient methods to find what authentication and confidentiality goals a protocol achieves [17]; to determine when protocols may safely be combined [16]; to determine when type information may safely be omitted from a protocol [18]; and to generate protocols manually [15] or automatically [25] to achieve given goals. However, the approach relies on Dolev-Yao abstract cryptography. In this paper, we begin to adapt the strand space theory to the realities of cryptographic operators.

First, we show how to quantify the divergence between concrete cryptographic operators and traditional abstract encryption in the Dolev-Yao style, as used in a protocol, introducing the notion of *ϵ-faithfulness*. A protocol security goal, proved using abstract encryption, is *ϵ-faithful to a cryptographic primitive* if the probability that execution of the protocol—implemented using that primitive—violates the goal is $\leq \epsilon$. Establishing ϵ-faithfulness requires some stochastic assumptions. The security goals we will consider in this paper are authentication goals [33, 20, 31].

Second, for a particular primitive, we give precise, quantitative bounds on this divergence. If an attack does not succeed against a protocol with the abstract cryptography of the Dolev-Yao approach, then the likelihood it succeeds against the same protocol when implemented using this cryptographic primitive is below the bound ϵ. The particular primitive we consider here is a type of message authentication code. A function is chosen (using a shared secret) from a universal class in the sense of Carter and Wegman [11, 32]; the protocol participants apply the chosen function to their messages to construct tags. The tag serves to authenticate that an adversary not privy to the shared secret has not originated the message, or altered it before delivery. We expect our methods to extend to some other primitives, so we have designed our exposition to avoid the specifics of message authentication codes and Carter-Wegman hashing until Section 5.

For Carter-Wegman tagging functions, we achieve specific bounds for a probability of failure such as $\epsilon = 2^{-32}$ (see Section 5.4). The bounds are based on parameters. One parameter summarizes the lengths of randomly chosen values, such as nonces and keys. Another parameter is the number of runs; it bounds how many guesses the adversary may make and how many random values the regular participants must choose. In effect, this parameter dictates a re-keying schedule. Keys must be changed often enough to limit the number of sessions before re-keying, counting all sessions by non-adversary participants.

Technical Approach Our approach is based on two ideas, both relying on the strand space model of protocol execution [31, 17], expressed in the notion of *bundle*. A bundle is a directed graph describing the behavior of the adversary as well as the regular (non-adversary) principals. The arrows represent either message transmission and reception (written as single arrows $\rightarrow$) or the transition of a single principal through successive actions of a single session (written as double arrows $\Rightarrow$).

Bundles represent protocol executions at both the abstract level where reasoning is independent of cryptographic primitive and the concrete level where primitives are specified. A bundle represents an execution using abstract encryption when the messages transmitted and received belong to a suitable free algebra. A bundle represents an execution with particular cryptographic primitives when the messages are bitstrings generated using those primitives. We call bundles whose messages belong to a free algebra *abstract bundles*, while we call bundles whose messages are bitstrings *concrete bundles*. Given an abstract bundle, one may construct a concrete bundle by specifying cryptographic primitives and possibly other operations such as concatenation; one obtains a specific bitstring $\pi(t)$ by applying these operations to the basic values in each abstract message t. Thus any abstract bundle $\mathcal{B}_a$ determines a unique concrete bundle $\pi(\mathcal{B}_a)$.

Our first idea addresses the inverse question, that is whether a concrete bundle $\mathcal{B}_c$ is of the form $\pi(\mathcal{B}_a)$. We regard the adversary as acting according to an *adversary strategy*. A strategy has two ingredients:

1. A partially ordered set $\mathcal{N}_a$ of abstract regular nodes, which we call an *abstract skeleton*,

2. For each negative regular node n in $\mathcal{N}_a$, a synthesis function g_n which returns a concrete term for a sequence of concrete terms as inputs.

The adversary's job is to produce a concrete bundle $\mathcal{B}_c$ in which the regular nodes are $\pi(\mathcal{N}_a)$; the work lies in computing the right value to feed into each negative node n. For this the adversary uses g_n. For some strategies, these concrete bundles $\mathcal{B}_c$ may be possible in the abstract Dolev-Yao model, while for others, the synthesis function g_n exploits some characteristic of the particular cryptographic primitives in use, which may or may not work for all bitstrings. In this case, the adversary builds concrete bundles to which no abstract bundle corresponds.

Since a strategy may not work for all values of the parameters, we use, as our second main idea, a completely classical stochastic model. It quantifies the probability that the adversary succeeds in creating the desired concrete bundle $\mathcal{B}_c$, when there is no Dolev-Yao attack. We make some stochastic assumptions about $\mathcal{B}_c$ (Section 5), that certain parameters of the resulting bundles are stochastically independent of each other. We also assume that certain parameters of the bundles are uniformly distributed. From these assumptions, it may follow in a specific case (in our example, see Corollary 5.2) that the probability that $\mathbf{B}$ is a correct bundle, when there is no Dolev-Yao attack, is less than a suitable ϵ.

These two ideas therefore bound the divergence between what may happen in concrete bundles $\mathcal{B}_c$ using the concrete cryptographic primitive, when all abstract bundles $\mathcal{B}_a$ satisfy some security goal.

An Example In this paper, we will focus in most detail on pure entity authentication protocols. They involve honest participants, whom we will call *regular* principals, and an adversary. A tagging function f is selected from a large class of possible functions for use by some set of ordered pairs of regular participants. We assume the adversary does not know which function has been chosen.

Consider the protocol MAP1 of Bellare and Rogaway [6], whose intended behavior is summarized on the left in Figure 1. In this protocol, the initiator (called A here) sends in the clear a nonce (random bit string) of the form N_a to start an exchange intended for a responder (called B here). The responder B generates a fresh nonce N_b, which we assume is distinct from N_a, and responds to A's message by sending a term of the form $[\![B \cdot A \cdot N_a \cdot N_b]\!]_f = (B \cdot A \cdot N_a \cdot N_b) \cdot f(B \cdot A \cdot N_a \cdot N_b)$. Since f is unknown to the adversary, the value $f(B \cdot A \cdot N_a \cdot N_b)$ is intended to serve as a message authentication code, guaranteeing the integrity of the message to the recipient. When A receives $[\![B \cdot A \cdot N_a \cdot N_b]\!]_f$, it responds with $[\![A \cdot N_b]\!]_f$, thereby assuring B that the value N_b has been received by A. Again, $[\![A \cdot N_b]\!]_f$ is really a concatenation $(A \cdot N_b) \cdot f(A \cdot N_b)$.

MAP1 achieves entity authentication: If A has had a run with intended respondent B, then B has undertaken at least the first two steps of a run with intended initiator A, and the runs agree on the nonces N_a, N_b. Conversely, if B has had a run with intended initiator A, then A has had a run with

$$\begin{tikzcd}
A \arrow[r, "N_a"] \arrow[d, Rightarrow] & B \arrow[d, Rightarrow] \\
\bullet \arrow[d, Rightarrow] & \bullet \arrow[l, "\llbracket B\cdot A\cdot N_a\cdot N_b \rrbracket_f"'] \arrow[d, Rightarrow] \\
\bullet \arrow[r, "\llbracket A\cdot N_b \rrbracket_f"] & \bullet
\end{tikzcd}$$

Figure 1: Intended Run of MAP1 Protocol

intended respondent B, and the runs agree on the nonces N_a, N_b. In both these results, we need the assumption that the principal has chosen fresh nonces, and that the choice of f was secret.

The protocols we emphasize here do not have the goal of causing the participants to agree on any new secret. In this sense they are pure entity authentication protocols. Of course, preserving the secrecy of the choice of f is necessary. However, if the secrecy of f fails, then the authentication goals will also fail. That is why we will not need to treat secrecy goals directly.

2 Strand Spaces

2.1 Strands: Basic Notions

We very briefly summarize the ideas behind the strand space model [31, 17]; see also Appendix A. Let A be a structure; we call the members of the domain of A *messages* or *terms*, because they represent values that can be sent between principals. Operations of A include encryption and concatenation, and a binary relation on messages called the *subterm* relation, written $t \sqsubset t'$. We will define these structures in more detail in Section 2.2.

Strands A *strand* is a sequence of message transmissions and receptions, where transmission of a term t is represented as $+t$ and reception of term t is represented as $-t$. A strand represents the local view of a participant in a run of a protocol. Each vertical column in Figure 1 shows a strand, assuming that particular values are chosen for the parameters A, B, N_a, and N_b. A strand element is called a *node*. A *strand space* Σ is a set of strands. (See Definition A.1.)

If s is a strand and i is between 1 and the length of s, then $\langle s, i\rangle$ is the i^{th} *node* on s. The relation $n \Rightarrow n'$ holds between nodes n and n' if $n = \langle s, i\rangle$ and $n' = \langle s, i+1\rangle$. The relation $n \rightarrow n'$ represents inter-strand communication; it means that $\operatorname{term}(n_1) = +t$ and node $\operatorname{term}(n_2) = -t$. The two relations $\Rightarrow$ and $\rightarrow$ jointly impose a graph structure on the nodes of Σ. The vertices of this graph are the nodes, and the edges are the union of $\Rightarrow$ and $\rightarrow$.

Bundles A *bundle* is a causally well-founded collection of nodes and arrows of both kinds. In a bundle, when a strand receives a message m, there is a unique node transmitting m from which the message was received. By contrast, when a strand transmits m, many strands (or none) may receive m. (See Definition A.3.)

Origination A term t *originates* at a node $n = \langle s, i \rangle$ if the sign of n is positive; $t \sqsubset \text{term}(n)$; and $t \not\sqsubset \text{term}(\langle s, i' \rangle)$ for every $i' < i$. Thus, n represents a message transmission that includes t, and it is the first node in s including t. If a value originates on only one node in some set S of nodes, such as those contained in a particular bundle, then we call it *uniquely originating* in S; uniquely originating values are desirable as nonces. (See Definition A.2.)

Regular Strands For a legitimate participant, a strand represents the messages that participant would send or receive as part of one particular run of his side of the protocol. We call a strand representing a legitimate participant a *regular* strand. Typically, the regular strands of Σ are the instances of a finite number of parameterized strands (See Section 2.3.)

The Standard Adversary For the adversary, each strand represents an atomic deduction. More complex actions can be formed by connecting several adversary strands. While regular principals are represented only by the messages they send to others, the behavior of the adversary is represented more explicitly, because the values it deduces are treated as if they had been transmitted publicly, or as if they were sent by the adversary to itself. We partition adversary strands according to the operations they exemplify. C-strands and S-strands concatenate and separate terms, respectively; K-strands emit keys from a set of known keys; and M-strands emit known atomic texts or guesses. In protocols which use a genuine encryption operator, E-strands encrypt when given a key and a plaintext; D-strands decrypt when given a decryption key and matching ciphertext. (See Definition A.6.)

We will also consider non-standard adversaries below, which may be able to apply additional useful functions to messages.

The Form of Authentication Goals As an example of an authentication goal, consider the responder's guarantee in MAP1. Suppose that the responder B has a run apparently with A, using the nonces N_a and N_b. B may assume that the nonce N_b is uniquely originating, because he generates it himself using highly random methods. B's authentication guarantee is the implication:

$\mathfrak{A}_B$: **if** B has a run of the protocol as responder, apparently with A and involving nonces N_a and N_b, and N_b is uniquely originating, **then** A has had a matching run as initiator, apparently with B, using the nonces N_a and N_b.

Naturally, there are two ways that A may turn out not to have a matching run. One is that some attack on the protocol or underlying cryptography occurs. The other is that B has (by bad luck) chosen a nonce N_b that has been used before, so that the adversary can re-use some part of a message generated in a non-matching run. We will treat these two types of failure separately, focusing on the first type for the bulk of the paper, and returning to the problem collisions of randomly chosen values in Section 5.4.

2.2 Cryptoalgebras

A cryptoalgebra is a structure A where the domain is partitioned into *atoms*, *concatenations*, and *encryptions*. We also assume that atoms include disjoint sets of *nonces* and *keys*; there may be other atoms (e.g. serving as names of principals). We write $\mathsf{A}_{\mathrm{atom}}$, $\mathsf{A}_{\mathrm{nonce}}$, and $\mathsf{A}_{\mathrm{key}}$ for the atoms, nonces, and keys. We refer to the union of the keys and nonces as *cryptovalues*, and write $\mathsf{A}_{\mathrm{crypto}} = \mathsf{A}_{\mathrm{key}} \cup \mathsf{A}_{\mathrm{nonce}}$. By this we mean that they are intended to be chosen unpredictably, so that it is unlikely that choices can be guessed, or that independent choices will coincide.

A is equipped with two (possibly partial) binary operators, *concatenation* and *encryption*. Concatenation maps two messages to a concatenation, and encryption maps a message and a key to an encryption. We write $t_1 \cdot t_2$ for the concatenation of t_1 and t_2 and $\{|t|\}_K$ for the encryption of t by K.

We assume that t is a concatenation just in case there is a unique pair of values t_1, t_2 such that $t = t_1 \cdot t_2$. We also assume that terms may be equipped with a natural number-valued rank function such that the terms of rank 0 are precisely the atoms and encryptions, and if $t = t_1 \cdot t_2$, then the rank of t exceeds the ranks of t_1 and t_2.

We are interested in the subterm relation $\sqsubset$ primarily in the case where the cryptoalgebra is freely generated from atoms by concatenation and encryption, as we will discuss shortly.

Cryptoalgebras of Bitstrings Suppose B is a cryptoalgebra whose domain is a finite set of bitstrings. We call such a cryptoalgebra a *concrete cryptoalgebra*.

We assume that the various kinds of atoms—keys, nonces, and non-cryptovalues such as names—are distinguished by tags or prefixes of some kind. The keys may be further subdivided by tags that indicate an algorithm or cryptographic transform, together with a representation of the actual key.

The concatenation function will be partial, being undefined when for instance the result would be too long.

We allow the set of "encryptions" in B to contain the results of various kinds of cryptographic transform, whether or not the transforms are all genuine encryptions. For instance, we will refer to the result of a keyed hash as an encryption, despite the fact that no one knows how to decrypt it. A cryptographic transformation may also be undefined for some arguments, for instance if the key is unsuitable for the algorithm by its length or number-theoretic properties.

We assume that some form of encoding such as ASN.1 is used to represent atoms, concatenations, and encryptions unambiguously.

Free Cryptoalgebras If B is a concrete cryptoalgebra, define A_B to be the free algebra generated from $\mathsf{B}_{\mathrm{atom}}$ by two free total operations of concatenation and encryption. Thus, there is a canonical partial map $\pi : \mathsf{A}_\mathsf{B} \to \mathsf{B}$ which maps a term $t \in \mathsf{A}$ to a bitstring in the domain of B by replacing the abstract operations of A with the corresponding concrete ones; it is undefined when the concrete operations in B are undefined. We omit the subscript B when no confusion can result.

If $\overline{X} = \{X_i\}_{i \in I}$ is a set of parameters, we also consider the free cryptoalgebra $\mathsf{A}_\mathsf{B}(\overline{X})$ generated by $\mathsf{B}_{\mathrm{atom}}$ and parameters X_i, also used like atoms. In our intended interpretation, each parameter has a value determined by a regular participant.

We equip $\mathsf{A}_\mathsf{B}(\overline{X})$ with the subterm relation defined to be the smallest reflexive, transitive relation $\sqsubset$ such that $g \sqsubset g{\cdot}h$; $h \sqsubset g{\cdot}h$; and $g \sqsubset \{\!|g|\!\}_K$. In particular, $K \not\sqsubset \{\!|g|\!\}_K$ unless $K \sqsubset g$.

By contrast, we say that a parameter X *occurs* in a term t if either $X \sqsubset t$ or $\{\!|t'|\!\}_X \sqsubset t$. That is, a parameter occurs in a term also if it is in key position.

The possible choices of values for parameters are given by *assignment functions* associating an atom in $\mathsf{B}_{\mathrm{atom}}$ with each parameter. We allow each parameter to carry a type indicator such as nonce or name, in which case we consider only assignments in which these parameters take values in the appropriate subset of $\mathsf{B}_{\mathrm{atom}}$. In particular, if a parameter ranges over cryptovalues (i.e. nonces or keys), then we call that parameter a *cryptoparameter*. If $t \in \mathsf{A}(\overline{X})$ and α is an assignment, then π_α is defined as the canonical partial map $\mathsf{A}(\overline{X}) \to \mathsf{B}$ that extends π and such that $\pi_\alpha(X) = \alpha(X)$ for each parameter $X \in \overline{X}$. A *substitution* is a function σ on parameters such that $\sigma(X)$ is always either a parameter of the same type as X or else a value of suitable type in B_{atom}.

A *parametric strand* over $\overline{X}$ is a finite sequence s of signed terms in $\mathsf{A}(\overline{X})$; we carry over the same terminology of nodes, etc. from parameter-free strands (Definition A.2).

A parameter X has a *primary occurrence* in a node n on s if (1) n is positive; (2) X occurs in term(n); and (3) if $n' \Rightarrow^+ n$, then X does not occur in term(n'). This definition differs from the definition of origination because it uses *occurs* rather than *subterm. Primary occurrence* is to *point of origination* as *occurs* is to *subterm.* A primary occurrence of X is thus either a point of origination for X, if X is actually a subterm of the message transmitted, or else X is being used for the first time as the key to construct some encrypted ingredient of the message.

2.3 Representing Protocols in Strand Spaces

A protocol requires regular participants to play a number of different roles, such as initiator, responder, or key server. The protocol itself consists of a number of parametric strands, one for each role played by the regular principals. These parametric strands may be determined by programs executed by the principals against their local state; our concern is exclusively with the resulting behaviors.

For instance, the parametric strand $\text{MAP1Init}_f[A, B, N_a, N_b]$ that has parameters A, B, N_a, N_b and signed terms

$$\langle +N_a,\ -[\![\, B{\cdot}A{\cdot}N_a{\cdot}N_b \,]\!]_f,\ +[\![\, A{\cdot}N_b \,]\!]_f \rangle$$

defines the MAP1 initiator's behavior. The complementary parametric strand with behavior

$$\langle -N_a,\ +[\![\, B{\cdot}A{\cdot}N_a{\cdot}N_b \,]\!]_f,\ -[\![\, A{\cdot}N_b \,]\!]_f \rangle.$$

defines the responder's behavior $\text{MAP1Resp}_f[A, B, N_a, N_b]$. The parameters A, B range over names, while the parameters N_a, N_b range over nonces. No parameter ranges over concatenated terms such as $A{\cdot}N_a$.

A protocol may also have parameters. In MAP1, the shared secret f is a parameter of the protocol itself; given a value for f, a set of regular participants agree on that value. That is why f is not listed as a parameter of the parametric strands. The parameter f is private value; adversaries presumably cannot guess it, to emit the same value themselves. In other protocols, parameters to the protocol may be public, such as public keys for the participants, which are known to adversaries and also regular participants.

We regard a *protocol* as a structure $\langle \mathcal{S}, \mathsf{secret}, \mathsf{public} \rangle$, where $\mathcal{S}$ is a set of parametric strands, secret is a set of parameters containing the secret protocol-wide parameters occurring in members of $\mathcal{S}$, and public is a set of parameters containing the secret protocol-wide parameters occurring in members of $\mathcal{S}$.

Thus, MAP1, acting with shared secret f, is the structure $\text{PROT} = \langle \mathcal{S}_f, \{f\}, \emptyset \rangle$, where $\mathcal{S}_f = \{\text{MAP1Init}_f[A, B, N_a, N_b], \text{MAP1Resp}_f[A, B, N_a, N_b]\}$.

Given a message algebra A, a protocol $\text{PROT} = \langle \mathcal{S}, \mathsf{secret}, \mathsf{public} \rangle$ determines a set of strand spaces, which we call *parameter-free strand spaces generated by* PROT *over* A. Each such strand space Σ is defined in four steps:

1. Assign a value of compatible type to each parameter in $\mathsf{secret} \cup \mathsf{public}$, and let $\mathcal{S}'$ be $\mathcal{S}$ with these parameters replaced by the associated values.

2. The set of regular strands Σ_R contains all parameter-free strands of the form $s[\alpha]$ for $s \in \mathcal{S}'$ and α an assignment.

3. The set of adversary strands Σ_P contains all the strands of Definition A.6 over the algebra A.

4. $\Sigma = \Sigma_R \cup \Sigma_P$.

Similarly, we can define the *strand space with parameters generated by* PROT *over* $\mathsf{A}(\overline{X})$. Here, the protocol-wide parameters need no instantiation, but remain unchanged as free parameters. Hence, there is only one such space $\Sigma(\overline{X})$, defined by:

1. The set of regular strands Σ_R contains all parametric strands of the form $s[\alpha]$ for $s \in \mathcal{S}$ and σ a substitution where $\sigma(X) = X$ for each parameter X such that $X \in \mathsf{secret} \cup \mathsf{public}$.

2. The set of adversary strands Σ_P contains all the strands of Definition A.6 over the algebra A.

3. $\Sigma = \Sigma_R \cup \Sigma_P$.

Because there are no public protocol-wide parameters in the example we consider here, we omit public parameters in the remainder of our discussion.

In MAP1, the parameters range only over names and nonces, not over concatenated or tagged terms. This is the case for all (natural) pure authentication protocols, so we will assume it throughout the remainder of the paper. The assumption would not hold for other protocols, particularly shared-key protocols using a key server, such as Otway-Rees [23] or Carlsen [10]; see [17, Section 5.1.3].

3 Bundles and Skeletons

Fix a protocol PROT and let $\mathcal{B}$ be a bundle in $\Sigma(\overline{X})$, the strand space with parameters for PROT over $\mathsf{A}(\overline{X})$. The *$\mathcal{B}$-height* of a strand s is the number of nodes of s belonging to $\mathcal{B}$; the $\mathcal{B}$-height of s is less than the length of s if some nodes at the end of s are not included in $\mathcal{B}$. This represents their "not having happened yet" at the time the bundle represents; possibly they will never happen. We define $\preceq_{\mathcal{B}}$ to be the reflexive transitive closure of the arrows of both kinds in $\mathcal{B}$, so that $m \preceq_{\mathcal{B}} n$ means that there is a sequence of arrows leading from m to n in $\mathcal{B}$.

A set of regular strands S is *generic* for PROT if:

1. If a cryptoparameter X originates on a regular strand s in S, then X originates nowhere else in S; and

2. If a cryptoparameter X is a secret parameter to PROT, then X originates nowhere in S.

Our notion of subterm excludes the keys used in encryption, so a non-originating parameter is not useless. It may occur as a key used by one or more participants.

We do not expect the same cryptoparameter to originate on more than one strand, as this would suggest that independent choices are perfectly correlated, with no visible causal connection. A non-generic bundle would have a magic correlation between the values chosen for cryptoparameters at apparently unrelated points. There is a causally inexplicable link between values chosen independently on different strands, or between a value chosen on some strand and a value chosen as a parameter to the protocol. Hence, we restrict our attention in this paper to generic bundles.

Definition 3.1 ***(Skeleton)*** *An* unordered skeleton *is a pair* $(\mathcal{R}, h)$ *where* $\mathcal{R}$ *is a set of regular strands over* $\overline{X}$*, and* $h\colon \mathcal{R} \to \mathbb{N}$ *is a height function such that* $s \in \mathcal{R}$ *implies* $h(s) \leq \mathsf{length}(s)$*. A* node n *of the unordered skeleton is a pair* $n = (s, i)$ *where* $s \in \mathcal{R}$ *and* $1 \leq i \leq h(s)$*.*

An (ordered) skeleton $\mathbb{A}$ *is a generic unordered skeleton* $(\mathcal{R}, h)$ *together with a weak partial ordering* $\preceq_{\mathbb{A}}$ *on nodes of* $(\mathcal{R}, h)$*, where* $\preceq_{\mathbb{A}}$ *is compatible with the strand order. That is,* $(s, i) \preceq_{\mathbb{A}} (s, j)$ *when* $s \in \mathcal{R}$ *and* $1 \leq i \leq j \leq h(s)$*.*

If $\mathcal{B}$ *is a generic bundle, then the* (ordered) skeleton of $\mathcal{B}$ *is the triple* $(\mathcal{R}, h, \preceq_{\mathbb{A}})$*, where* $\mathcal{R}$ *is the set of regular strands with at least one node in* $\mathcal{B}$*, and* $h(s)$ *is the* $\mathcal{B}$*-height of* s*; we let* $\preceq_{\mathbb{A}}$ *be the restriction of* $\preceq_{\mathcal{B}}$ *to the nodes of* $\mathcal{R}$*.*

We typically omit the word "ordered." Observe that if $\mathcal{B}$ is a generic bundle, then the ordered skeleton of $\mathcal{B}$ is generic.

3.1 Security Properties

For the purposes of this paper, we define:

Definition 3.2 *A* security property *is a set* ϕ *of generic bundles such that if* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *have the same (ordered) skeleton, then* $\mathcal{B}_1 \in \phi$ *iff* $\mathcal{B}_2 \in \phi$*.*

Authentication properties such as those proved in previous papers [31, 17, 14] are security properties in this sense, including assertions of recency.

Secrecy properties are not exactly of the right form, because they typically state that the adversary does not extract a particular value v. However, they may be brought to a satisfactory form in more than one way. For instance, one could declare a value v to be *compromised* in a bundle $\mathcal{B}$ if there is a bundle $\mathcal{B}'$ differing from $\mathcal{B}$ only in having additional adversary strands, such that $\mathcal{B}'$ contains a node n where $\mathrm{term}(n) = v$. Then v being uncompromised is a security property in our sense. Alternatively, one can regard a principal, having used a secret value v, as offering an event $-v$ in which it receives the secret; this event (conceptually) admits that the adversary has captured the secret. Then a skeleton that contains a regular node of this kind is an example of a failure of secrecy.

3.2 Dolev-Yao Realizability

The question whether a security goal is achieved by a particular protocol is essentially the question whether, in a set of skeletons representing failures of that goal, there are any that can be completed to a generic bundle by adding adversary activity. The answer may depend on a particular model of the adversary, and perhaps also on models of the particular forms of cryptography in use.

A set $S = \{s_1, \cdots, s_j\}$ of standard adversary strands (as in Definition A.6) *derives* a term t from a set of terms T if it is possible to connect positive and negative nodes of S by arrows $\rightarrow$ so that:

1. the resulting graph (with the added $\rightarrow$ edges and the $\Rightarrow$ edges of the strands in S) is acyclic;

2. t occurs on a positive node on this graph;

3. if n is any node without an incoming $\rightarrow$ arrow, then $\text{term}(n) \in T$.

Given a skeleton defining potential behavior of the regular principals, the adversary may want to "realize" it (complete it to a bundle) to show that the regular principals can be forced to behave in this way. If this can be done using only standard adversary strands, then we say that the skeleton is (Dolev-Yao) realizable.

Definition 3.3 ***(Dolev-Yao Realizable)*** *Let $\mathbb{A}$ be a skeleton, and let n be a negative (regular) node in $\mathbb{A}$. Node n is* Dolev-Yao realizable *in $\mathbb{A}$ if there are positive nodes $m_1, \ldots, m_k$ in $\mathbb{A}$ and standard adversary strands $s_1, \ldots, s_j$ (for some k, j) such that:*

1. *$m_i \prec_{\mathbb{A}} n$ for all $1 \leq i \leq k$;*

2. *any parameter on a M or K strand in $s_1, \ldots, s_j$ has no primary occurrence in $\mathbb{A}$;*

3. *$s_1, \ldots, s_j$ derive $term(n)$ from $\{term(m_1), \ldots, term(m_k)\}$.*

The skeleton $\mathbb{A}$ is Dolev-Yao realizable *if every negative node in $\mathbb{A}$ is realizable in $\mathbb{A}$.*

Proposition 3.4 *$\mathbb{A}$ is Dolev-Yao realizable if and only if there exists a generic bundle $\mathcal{B}$ such that the skeleton of $\mathcal{B}$ is $\mathbb{A}$.*

Thus, the strand space proof methods such as the authentication tests, safe keys, and honest ideals, can be used to establish whether a skeleton is Dolev-Yao realizable. A generic bundle provides a witness to the adversary's ability to break security goals that would rule out $\mathbb{A}$.

In this paper we mainly investigate skeletons which are non Dolev-Yao realizable

Corollary 3.5 *If there is no generic bundle $\mathcal{B}$ such that the skeleton of $\mathcal{B}$ is $\mathbb{A}$, then there is a negative regular node n of $\mathbb{A}$ such that n is not realizable in $\mathbb{A}$.*

Suppose that the skeleton $\mathbb{A}$ is not Dolev-Yao realizable: How can the adversary construct a concrete bundle in which the regular strands are of the form $\pi(s)$ for $s \in \mathbb{A}$? Corollary 3.5 suggests a way to reduce this to local, purely cryptographic questions about non-realizable nodes n.

4 Adversary Strategies

To make this suggestion more precise, we need a formalization of the adversary's approach to synthesizing the non-realizable terms needed to complete $\mathbb{A}$. Moreover, the adversary wants to maximize the probability that it can produce a bundle with spoofed values in place of the messages that would have been sent by regular participants in the intended run of the protocol. To formalize this we will first introduce stochastic elements into our model.

4.1 Stochastic Elements

These elements consist of an underlying probability space, together with some random variables[1] that extract aspects of the behavior. We must assume some constraints, which require either that a random variable is uniformly distributed, or else that random variables are independent of one another.

For convenience, we assume that the probability space (Ω, Pr) is finite, as we may do because the sets of messages (bitstrings of bounded size) are finite and the size of the bundles of interest are bounded. The space (Ω, Pr) encapsulates an array of information including the choice of nonces and of interlocutors by the regular participants.

We can give a probabilistic interpretation to skeletons $\mathbb{A}$ as follows: Associated to each parameter X is a B_{atom}-valued random variable $\mathbf{X}$. We will use the convention that if X is an parameter, $\mathbf{X}$ is the random variable corresponding to X.

4.2 Strategies

An *adversary strategy* consists of a skeleton $\mathbb{A}$ and a *synthesis function* g_n for each negative regular node n. Synthesis functions are used by the adversary to synthesize a value $b \in \mathsf{B}$ from previously seen values that will be regarded as legitimate by the recipient. "Legitimate" means the recipient sees that b has the right syntactic form and particular components within b match

[1] A random variable (sometimes we write just "variable") is a function on the underlying probability space.

previously seen values. It is natural to allow the synthesis functions to be probabilistic functions.

Definition 4.1 *A* randomized adversary strategy *(or simply adversary strategy) G consists of*

1. *A skeleton $\mathbb{A}$;*
2. *A source of randomness $\mathbf{R}$ for the adversary; and*
3. *For each negative regular node n, a function $g_n : \mathsf{B}^k \times \mathsf{R} \rightarrow \mathsf{B}$ for some k, and a list $m_1, \ldots, m_k$ of positive nodes from $\mathbb{A}$ such that for each m_i, $m_i \prec_{\mathbb{A}} n$.*

Naturally, different adversaries may have different collections of synthesis functions g_n available. The minimal set of interest is encryption and decryption with known keys; concatenation and separation; originating new values; and compositions of these operations. This is the Dolev-Yao adversary, and strategies using these synthesis functions can succeed only if $\mathbb{A}$ is Dolev-Yao realizable. At the opposite end of the spectrum, the strongest plausible adversary can select any probabilistic polynomial-time Turing machine to compute the synthesis function to derive term(n). In this paper, we leave the class of possible functions unconstrained.

Our notion of strategy is more limited than allowing the adversary to select as its strategy any polynomial time Turing machine, using regular participants as oracles (cf. e.g. [6]). In our approach, the adversary must start by choosing a finite set of regular strands. The adversary must also choose, for each of the negative regular nodes n, a particular set of earlier positive regular nodes to furnish the materials needed to create a message to deliver to n. These choices are made before any data values are observed.

Consequently, the security guarantees that we establish are weaker. However, they are easier to establish and quite informative. Indeed, it is an open question whether this difference in power has practical significance.

4.3 Model Assumption

We assume that parameters to the protocol are assigned values independently of each other. We also expect regular participants to choose values for cryptoparameters on their strands independently of other parameters and of the protocol-wide parameters. Moreover, they should (ideally speaking) be chosen according to a uniform distribution. Finally, the adversary's source of randomness should be independent of the protocol-wide secret parameters, and independent of the choices of regular participants. These rules codify natural assumptions about lack of causal interactions in the choice of cryptographic values.

Assumption 1 *Let G be an adversary strategy with abstract skeleton $\mathbb{A}$. Let P contain the random variables for secret protocol-wide parameters* secret*, and for the parameters to regular strands in $\mathbb{A}$. We assume:*

1. *The adversary's source of randomness $\mathbf{R}$ is stochastically independent of the random variables for parameters in P, taken jointly.*
2. *If $\mathbf{X} \in P$, then $\mathbf{X}$ is stochastically independent of all other random variables $\mathbf{Y} \in P \setminus \mathbf{X}$, taken jointly.*
3. *If $\mathbf{X} \in P$, then $\mathbf{X}$ is uniformly distributed.*

4.4 The Success Set for a Strategy

Given an adversary strategy G with skeleton $\mathbb{A}$ and a family of synthesis functions $\{g_n\}$ indexed on the negative regular nodes of $\mathbb{A}$, we obtain a probabilistic realization of G by replacing each parameter in the abstract skeleton with the corresponding random variable. This produces a *random* abstract skeleton. For each node n of this random abstract skeleton, $\mathrm{term}(n)$ is a random variable with values in some free cryptoalgebra. Applying the implementation function π to these random terms yields a "prebundle" $\mathcal{B}_G(\omega)$, that is a subgraph of a strand space in which positive and negative nodes do not necessarily match in their terms. Each term $\pi(\mathrm{term}(n))$ of this prebundle is an element of the concrete cryptoalgebra B.

We will use the notation $\mathbf{t}_n$ to denote the random variable $\pi(\mathrm{term}(n))$. Some of these random variables are associated with positive nodes, while others are associated with negative nodes. We write $\mathbf{t}_n^+$ when necessary to emphasize that this random variable is associated with a positive node n.

This prebundle is a bundle if, for each negative node n in the abstract skeleton, with associated list of earlier positive regular nodes $m_1, \ldots, m_k$, the value synthesized by the adversary using g_n is the same as the value that the regular principal expected to receive at n. If the sample point for the random variables is ω, then this means:

$$g_n\big(\mathbf{t}_{m_1}(\omega), \ldots, \mathbf{t}_{m_k}(\omega), \mathbf{R}(\omega)\big) = \mathbf{t}_n(\omega). \tag{1}$$

The *success set* for a strategy is the set of sample points ω for which $\mathcal{B}_G(\omega)$ is actually a bundle, or equivalently the set of ω that (1) is true for all the g_n used in G.

If an adversary strategy G has abstract skeleton $\mathbb{A}$, and $\mathbb{A}$ is Dolev-Yao realizable, then the adversary can certainly find functions g_n that solve all the instances of equation (1) for all values of the parameters. The realization of n from earlier nodes implies that the adversary can implement g_n as a composition of encryptions, decryptions, concatenations, and separations. If G is not realizable, then there exists at least one node n for which the adversary cannot implement g_n in this way (Corollary 3.5). For each remaining equation, the adversary wants to choose a function g_n that will solve the equation in a large set of cases. Thus, the probability that the adversary wins with G is the maximum over all assignments of the available functions to nodes n of the probability of satisfying all of the remaining equations.

For a fixed skeleton $\mathbb{A}$, one wants an upper bound for the probability that (1) holds for all negative non-Dolev-Yao realizable nodes n. In the example we will consider, an upper bound can be obtained which depends only on the size of the skeleton $\mathbb{A}$.

As the skeleton $\mathbb{A}$ varies the set of equations changes, since the terms $\mathbf{t}_{m_i}$ available to use on the left hand side of equation (1) vary. However, the terms will be messages transmitted in this protocol, by other strands or by the same strand before n. Thus, analyzing the risk of a non-Dolev-Yao failure for a protocol reduces to bounding the probability that formulas of the form (1) are true, for terms emitted by the protocol. Again, some limitation on the size of $\mathbb{A}$ may be reasonable.

Definition 4.2 *Suppose that a protocol is defined over* $\mathsf{A_B}$ *and implemented via* $\pi\colon \mathsf{A_B} \rightarrow \mathsf{B}$*. Then* π *is* ϵ-faithful *to a security goal for a class of skeletons if the probability that the goal fails is* $\leq \epsilon$ *for penetrator strategies whose skeletons are in that class.*

In Section 5, we illustrate the usefulness of our approach by showing that a pure authentication protocol such as MAP1is ϵ-faithful for skeletons of bounded size, when the message authentication codes are implemented using universal classes of hash functions following Carter and Wegman [11].

One can also consider families of mappings $\{\pi_i\}_{i\in\mathbb{N}}$, taking values in a family of concrete cryptoalgebras $\{\mathsf{B}_i\}_{i\in\mathbb{N}}$ accommodating increasing large keys and nonces. One could then show that asymptotically, ϵ shrinks very fast as i increases, again using equation (1).

5 Pure Authentication Protocols

We now apply our approach to pure entity authentication protocols.

In MAP1, regular strands use a single nonce each, so N_n is defined only when n is the first positive node of a strand. Likewise, they send a single tagged message each.

Following the recipe described above that associates terms in the abstract cryptoalgebra to B-valued random variables, we identify the following random variables:

1. $\mathbf{F}\colon \Omega \to \mathcal{F}$, determining the secret tagging function $F \in \mathcal{F}$.

2. $\mathbf{N}_n\colon \Omega \to \mathsf{N}$, for each positive regular node n at which a nonce originates, determining the nonce.

3. $\mathbf{B}_n\colon \Omega \to \mathsf{B}$, for each negative regular node n at which a tagged value is received, determining the value that was tagged.

4. $\mathbf{R}\colon \Omega \to \{0,1\}^{\mathbb{N}}$, determining the choice supplied by the adversary's source of randomness.

5.1 Forgery

We use the concept of a randomized adversary strategy G, that is a skeleton $\mathbb{A}$ and a family of synthesis functions $\{g_n\}$ ranging over negative nodes of $\mathbb{A}$. As explained earlier, these elements are used to construct a random prebundle $\mathcal{B}_G(\omega)$. The terms received on negative nodes n have the form $b_n{\cdot}f(b_n)$.

In the general setup described earlier the inputs to each g_n are values transmitted by previous nodes $m_1, \cdots, m_k$ in the skeleton $\mathbb{A}$. In our analysis here we will be very generous and allow the adversary to the synthesize a value from any value ever transmitted by a regular participant,

Accordingly, the input to g_n consists of the following random elements:

1. The vector $\vec{\mathbf{t}}$ of values transmitted on regular positive nodes; we refer to the set of possible values from which $\vec{\mathbf{t}}$ is chosen as $\mathcal{T}$.

2. The adversary's own source of randomness $\mathbf{R}$.

The synthesis functions g_n in the strategy G can be regarded as the adversary's forgery attempts in the prebundle $\mathcal{B}_G(\omega)$. Since g_n captures forgeries, rather than message replay, we assume, without loss of generality, that there is no message b and tags v, v' such that (b, v) is the value some synthesis function g and $(b, v') = \mathbf{t}_n(\omega)$ for n a positive node.

The prebundle $B_G(\omega)$ has a successful forgery if for some negative node n of $\mathbb{A}$ the value of g_n given the arguments $\vec{\mathbf{t}}$ and $\mathbf{R}$ has the form $b \cdot f(b)$ where $f = \mathbf{F}(\omega)$. Thus, given a non-realizable node n, the solutions to equation 1 is the event:

$$\mathsf{forge}_n = \{\omega \colon g_n(\vec{\mathbf{t}}, \mathbf{R}) = \big(\mathbf{B}_n(\omega), \mathbf{F}(\omega)(\mathbf{B}_n(\omega))\big)\}.$$

Thus, if NR is the set of non-Dolev-Yao realizable nodes in $\mathbb{A}$, then we want to show that the event

$$\mathsf{forge} = \bigcap_{n \in NR} \mathsf{forge}_n$$

has small probability.

5.2 Pure Authentication Protocol Model Assumptions

We state three more specific assumptions (Assumptions 2–4) that follow from Assumption 1 and instantiate its content for the random variables of pure authentication protocols. Assumption 5 constrains the choice of hash functions, and Assumption 6 limits the size of the bundles of interest to us.

We regard $\mathbf{N}_n(\omega)$ as a family of nonce-valued random variables indexed by n.

Assumption 2 *For each node n, $\mathbf{N}_n$ is uniformly distributed, and for any $n' \neq n$ the variables $\mathbf{N}_n(\omega)$ and $\mathbf{N}_{n'}(\omega)$ are independent.*

This assumption is used only in Section 5.4.

Let fst be the function that delivers the first component of a pair, so that $\mathsf{fst} \circ \vec{\mathbf{t}}$ is the function that delivers the bodies but not the tags of the tagged messages sent by regular participants.

Assumption 3 *The variable $\mathbf{F}$ is stochastically independent of $\mathbf{R}$ and $\vec{\mathbf{t}}' = \mathsf{fst} \circ \vec{\mathbf{t}}$ taken jointly:*

$$\begin{aligned} \mathsf{Pr}\{\mathbf{F}(\omega) = f \wedge \vec{\mathbf{t}}'(\omega) = t' \wedge \mathbf{R}(\omega) = r\} \; &= \\ \mathsf{Pr}\{\mathbf{F}(\omega) = f\} &\cdot \mathsf{Pr}\{\vec{\mathbf{t}}'(\omega) = t' \wedge \mathbf{R}(\omega) = r\} \end{aligned}$$

Hence, $\mathbf{F}$ and $\mathbf{R}$ are pairwise independent, as are $\mathbf{F}$ and $\vec{\mathbf{t}}'$.

Assumption 4 *The distribution of $\mathbf{F}$ on $\mathcal{F}$ is uniform, that is, for any $E \subseteq \mathcal{F}$*

$$\mathsf{Pr}\{\mathbf{F}(\omega) \in E\} = \frac{\mathsf{card}(E)}{\mathsf{card}(\mathcal{F})}$$

Assumption 5 *For each message b and each $t \in \mathcal{T}$ and $r \in \mathcal{R}$,*

$$\mathsf{Pr}\{\mathbf{F}(\omega)(b) = v | \vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\} \tag{2}$$

does not depend on $v \in \mathsf{V}$.

We show in Section 5.5 that tagging functions satisfy this assumption if they make up a universal class in the sense of Carter and Wegman [11, 32]. A protocol limits the number of new nonces sent by a single regular strand. It also limits the number of signed expressions sent by a single regular strand. And it limits the number of signed expressions that can be received by a single regular strand. In MAP1, all of these numbers equal 1, though in another protocol they may have some maximum ν. Therefore, at most Λ nonces are used where

$$\Lambda = \nu \times \mathsf{card}\{\text{Regular Strands}\}.$$

It follows that the risk of two strands re-using a nonce can be easiliy bounded. The number of samples of the tagging function f that the regular participants show the adversary is limited by Λ. And the number of forgeries that the adversary may submit to the regular participants is bounded by Λ.

Assumption 6 *The number of nonces, tagged values sent, and tagged values received on regular nodes in $\mathbb{A}$ is bounded by some value Λ.*

This assumption is part of the justification for taking Ω to be finite. The existence of an upper bound on the number of regular strands is ultimately justified by a re-keying schedule. We require the participants to agree on a new value of f before the assumed upper bound on the number of sessions can be attained.

5.3 The Probability of Forgery

Given a particular ω, the adversary may observe $t = \vec{\mathbf{t}}(\omega)$ and $r = \mathbf{R}(\omega)$, the first being the tagged messages chosen by the regular participants and the second being the adversary's source of randomness. Each forgery attempt consists in generating and transmitting a tagged message $b \cdot v$.

Proposition 5.1 *For any t and r*

$$\mathsf{Pr}\{\mathbf{F}(\omega)(b) = v | \vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\} = \mathsf{Pr}\{\mathbf{F}(\omega)(b) = v\} = \frac{1}{\mathsf{card}(\mathsf{V})} \tag{3}$$

PROOF. Note that for any t and r:

$$\sum_{v \in \mathsf{V}} \mathsf{Pr}\{\mathbf{F}(\omega)b = v \mid \vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\} = 1$$

Since all terms in the sum have the same value,

$$\mathsf{Pr}\{\mathbf{F}(\omega)b = v \mid \vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\} = 1/\,\mathsf{card}(\mathsf{V})$$

From this follows that for any b, v,

$$\begin{aligned}
&\mathsf{Pr}\{\mathbf{F}(\omega)b = v\} \\
&\quad = \sum_{t \in \mathcal{T}, r \in \mathcal{R}} \mathsf{Pr}\{\mathbf{F}(\omega)b = v | \vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\} \ \ \mathsf{Pr}\{\vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\} \\
&\quad = \frac{1}{\mathsf{card}(\mathsf{V})} \sum_{t,r} \mathsf{Pr}\{\vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\} \quad = \quad \frac{1}{\mathsf{card}(\mathsf{V})}
\end{aligned}$$

Corollary 5.2 *The likelihood of forgery* forge *is at most*

$$\frac{1}{\mathsf{card}(\mathsf{V})}$$

5.4 Likelihood of Anomalies

As we mentioned in Section 2.1, there are two ways that an authentication goal may fail. One is that some attack on the protocol or underlying cryptography occurs. The other is that a participants has (by bad luck) chosen a nonce that has been used before, so that the adversary can re-use some part of a message generated in a non-matching run. In case the protocol is correct viewed abstractly, the first type of failure amounts to the event forge defined in Section 5.1.

The Probability of Nonce Clash The second type of failure arises when the regular participants choose clashing nonces, which we define:

$$\mathsf{clash} = \{\omega \colon \mathbf{N}_n(\omega) = \mathbf{N}_{n'}(\omega) \quad \text{where } n \neq n'\}$$

Since, by Assumption 2, the random variables $\mathbf{N}_n$ are uniformly distributed and mutually independent, determining a bound on the likelihood of a nonce anomaly is a special case of the "birthday problem" [13]. The total number of choices is bounded by Λ, so the likelihood of at least one clash is

$$\mathsf{clash} \leq \frac{\Lambda(\Lambda - 1)}{2\ \mathsf{card}(\mathsf{N})} \tag{4}$$

The Probability of Clashing or Forgery Combining Equation (4) with Proposition 5.2, we have:

Proposition 5.3 *Suppose for a pure authentication protocol that every Dolev-Yao realizable skeleton has matching initiator and responder strands, and Assumptions 2–6 hold.*

Then the protocol is ϵ-faithful for

$$\epsilon \geq \frac{\Lambda(\Lambda - 1)}{2\ \mathsf{card}(\mathsf{N})} + \frac{1}{\mathsf{card}(\mathsf{V})}$$

As an example, consider a tolerance of $\epsilon = 2^{-31}$ for the likelihood of forgeries and clashes together, where we will allocate half of ϵ for each type of anomaly. If nonces are given by 64-bit strings, then $\mathsf{card}(\mathsf{N}) = 2^{64}$. To ensure that independent choices of Λ nonces has probability of anomaly below $\epsilon/2$, it suffices to restrict Λ so that $\Lambda^2/2{\cdot}2^{64} \leq 2^{-32}$, i.e., $\Lambda \leq 2^{17}$, roughly $130,000$. If, for example, we would like to use a shared secret choice of f without change for a year, this would allow 300 strands per day, since $130000/365 > 300$.

For the case of forgeries, $\mathsf{Pr}(\mathsf{forge}) \leq 1/\,\mathsf{card}(\mathsf{V})$. To ensure that $\mathsf{card}(\mathsf{V}) \geq 2^{32}$, we need 32-bit tags.

Thus with $\Lambda = 2^{16}$, the likelihood of an authentication failure is

$$\epsilon \leq \mathsf{Pr}(\mathsf{forge}) + \mathsf{Pr}(\mathsf{clash}) \leq 2^{-32} + 2^{-32} = 2^{-31}$$

Each tag calculation requires substantial computation, but the rekeying is infrequent and the risk of authentication failure is very low. These numbers are only illustrative; the point is that we have described a comprehensive method that yokes abstract protocol design and verification using strand spaces to low-level calculations of the risk of security compromise.

5.5 Achieving Assumption 5

The condition in Assumption 5 (equation 2) can be reformulated as follows. Suppose b is a potential bogus message and v is a potential tag. Let t be a sequence of message and message tag pairs and let $t' = \mathsf{fst} \circ t$ and $t'' = \mathsf{snd} \circ t$ be the sequences of their first and second components, respectively. By the uniformity assumption 4 and the independence Assumption 3 we obtain the following sequence of identities:

$$\begin{aligned}
\mathsf{Pr}\{\mathbf{F}(\omega)b = v|\vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\}& \\
&= \frac{\mathsf{Pr}\{\mathbf{F}(\omega)b = v \wedge \vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\}}{\mathsf{Pr}\{\vec{\mathbf{t}}(\omega) = t \wedge \mathbf{R}(\omega) = r\}} \\
&= \frac{\mathsf{Pr}\{\mathbf{F}(\omega)b = v \wedge \mathbf{F}(\omega)t' = t'' \wedge \vec{\mathbf{t}}'(\omega) = t' \wedge \mathbf{R}(\omega) = r\}}{\mathsf{Pr}\{\mathbf{F}(\omega)t' = t'' \wedge \vec{\mathbf{t}}'(\omega) = t' \wedge \mathbf{R}(\omega) = r\}} \\
&= \frac{\mathsf{Pr}\{\mathbf{F}(\omega)b = v \wedge \mathbf{F}(\omega)t' = t''\}}{\mathsf{Pr}\{\mathbf{F}(\omega)t' = t''\}} \\
&= \frac{\mathsf{card}\{g \in \mathcal{F}_t : g(b) = v\}}{\mathsf{card}(\mathcal{F}_t)}
\end{aligned}$$

where $\mathcal{F}_t$ is the set of hash functions f such that $f(t') = t''$. Using this identity, we can reformulate the condition of Formula (2) in the following way. For any potential bogus message b after the observations t,

$$\frac{\mathsf{card}\{g \in \mathcal{F}_t : g(b) = v\}}{\mathsf{card}(\mathcal{F}_t)} \tag{5}$$

is independent of v. Intuitively what this condition means is that the adversary does no better after observing the sequence of message-tag pairs t than pure random guessing. But this is precisely the property studied in the classic papers by Carter and Wegman [11, 32], which we use in the form:

Definition 5.4 *A set of functions $\mathcal{F} \subseteq Y^X$ is n-strongly universal just in case the following two conditions are met: (1) $\mathsf{card}(X)$ is at least n, and (2) if $x_1, \ldots, x_n$ are any n pairwise distinct values in X, then the distribution of the evaluation mapping $f \mapsto \langle f(x_1), \ldots, f(x_n)\rangle$ is uniform.*

See Appendix B for supplementary details, where Lemma B.5 produces an example of an n-strongly universal class of functions.

6 Conclusion and Related Work

We believe that analysis of security protocols should consist of two independent steps. The first step should study the protocol formally with ideal cryptography and establishes its abstract correctness. The second step then should quantify the probability of the adversary succeeding at undermining a security goal using concrete information about the cryptographic primitives in use. The current paper develops a method for supplementing abstract protocol analysis with information about concrete cryptographic primitives used in one particular implementation.

The conclusions we would derive are akin to those of Bellare and Rogaway [6], who studied protocols without abstracting from cryptography, and established security results for specific protocols directly from the way that specific cryptographic operators are used in them. The newer work is a more convenient way to reach these results. The problem is split into a part specific to the protocol and a separate part specific to the cryptographic primitives.

A number of recent papers have considered how to interrelate cryptographic protocol analysis based on formal methods (roughly, the Dolev-Yao tradition) with more precise results that are sensitive to the cryptographic primitives in use. One of the pioneering papers of this line of work, by Pfitzmann, Schunter, and Waidner [26], develops a state machine model that allows secure composition [27, 28] and formally justified implementations [3], as carried out subsequently. Their work establishes that somewhat complex state machines have almost indistinguishable behavior when an "ideal" cryptographic substructure is replaced with a real cryptographic substructure satisfying certain properties. The notational burden of their state machine model seems quite high, and it is questionable whether they will find it easy to reason abstractly about properties of particular protocols. Pfitzmann et al. also question whether the Dolev-Yao model is cryptographically sound and provide a somewhat contrived counterexample to an intuitive notion of soundness.

Abadi and Rogaway [2, 1] study types of concrete cryptography that do not introduce additional attacks, beyond those predicted by the abstract protocol analysis. More precisely, any strategy of the adversary has a negligible probability of producing an additional attack. Abadi and Rogaway focused on a passive adversary, which observes messages but never generates them. The "concrete security" line of work (e.g. [4, 5]) treats protocols with a restricted adversary model, and with little or no attention to multi-party interaction.

Shoup approaches protocol verification from a primarily cryptographic point of view [30], and the underlying formal model is less sharply defined. Thus, it is hard to see exactly what forms of reasoning are justified at the formal level.

Canetti and Krawczyk [9], following Canetti [8], have proposed a very promising notion of security based on the idea that when a key agreement protocol delivers a session key to some "application protocol," then the application protocol should be able to use it freely without undermining the guarantees of the key agreement protocol. This approach leads to strong guarantees of security, and also provides valuable protocol design guidelines. It remains necessary to carry out the full protocol analysis within the cryptographic model; thus, this approach also does not furnish the separation of concerns between formal protocol analysis and cryptographic analysis of the primitives.

Summary In this paper we have shown that abstract message authentication codes are faithful in the sense that, when a protocol meets its security goals in an abstract model like the strand space model, then the probability that an adversary can defeat it is below a suitable ϵ such as 2^{-31}. Specifically, we have established this in the case in which the cryptographic primitive is Carter-Wegman hashing. The core method, using Corollary 3.5 to reduce protocol attacks to individual local forgeries, and equation 1 identify the cryptographic problem that must be solved to achieve a forgery, appears to be more widely applicable.

Acknowledgment We are grateful for the probing and thoughtful comments of this JOURNAL's anonymous referees.

References

[1] Martín Abadi and Jan Jürjens. Formal eavesdropping and its computational interpretation, 2000. submitted for publication.

[2] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[3] Michael Backes, Christian Jacobi, and Birgit Pfitzmann. Deriving cryptographically sound implementations using composition and formally verified bisimulation. In *Formal Methods Europe '02 (FME)*, volume 2391 of *LNCS*, pages 310–329. Springer Verlag, 2002.

[4] Mihir Bellare. Practice-oriented provable security. In E. Okamoto, G. Davida, and M. Mambo, editors, *First International Workshop on Information Security (ISW 97)*, volume 1396 of *LNCS*. Springer Verlag, 1998.

[5] Mihir Bellare, J. Killian, and Phillip Rogaway. The security of cipher block chaining. In Yvo Desmedt, editor, *Advances in Cryptology — Crypto 94*, volume 839 of *LNCS*. Springer Verlag, 1994.

[6] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – Crypto '93 Proceedings*. Springer-Verlag, 1993. Full version available at http://www-cse.ucsd.edu/users/mihir/papers/eakd.ps.

[7] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society*, Series A, 426(1871):233–271, December 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

[8] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Report 2000/067, Internation Association for Crryptographic Research, October 2001. Extended Abstract appeared in proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS), 2001.

[9] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In *Eurocrypt 2002*, LNCS, pages 337–351. Springer Verlag, 2002.

[10] Ulf Carlsen. Optimal privacy and authentication on a portable communications system. *Operating Systems Review*, 28(3):16–23, 1994.

[11] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18:143–54, 1979.

[12] Daniel Dolev and Andrew Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[13] W. Feller. *An Introduction to Probability Theory and its Applications.* John Wiley and Sons, Inc., New York, 1958.

[14] Joshua D. Guttman. Key compromise and the authentication tests. *Electronic Notes in Theoretical Computer Science*, 2001.

[15] Joshua D. Guttman. Security protocol design via authentication tests. In *Proceedings, 15th Computer Security Foundations Workshop*. IEEE Computer Society Press, June 2002.

[16] Joshua D. Guttman and F. Javier Thayer Fábrega. Protocol independence through disjoint encryption. In *Proceedings, 13th Computer Security Foundations Workshop*. IEEE Computer Society Press, July 2000.

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

[18] James Heather, Gavin Lowe, and Steve Schneider. How to prevent type flaw attacks on security protocols. In *Proceedings, 13th Computer Security Foundations Workshop*. IEEE Computer Society Press, July 2000.

[19] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceeedings of* TACAS, volume 1055 of *Lecture Notes in Computer Science*, pages 147–166. Springer Verlag, 1996.

[20] Gavin Lowe. A hierarchy of authentication specifications. In *10th Computer Security Foundations Workshop Proceedings*, pages 31–43. IEEE Computer Society Press, 1997.

[21] Catherine Meadows. A model of computation for the NRL protocol analyzer. In *Proceedings of the Computer Security Foundations Workshop VII*, pages 84–89. IEEE, IEEE Computer Society Press, 1994.

[22] Jonathan K. Millen. The Interrogator model. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pages 251–60, 1995.

[23] D. Otway and O. Rees. Efficient and timely mutual authentication. *Operating Systems Review*, 21(1):8–10, January 1987.

[24] Lawrence C. Paulson. Proving properties of security protocols by induction. In *10th IEEE Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, 1997.

[25] Adrian Perrig and Dawn Xiaodong Song. Looking for diamonds in the desert: Extending automatic protocol generation to three-party authentication and key agreement protocols. In *Proceedings of the 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, July 2000.

[26] Birgit Pfitzmann, Matthias Schunter, and Michael Waidner. Cryptographic security of reactive systems. *Electronic Notes in Theoretical Computer Science*, 32, 2000.

[27] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proceedings, Seventh ACM*

Conference of Communication and Computer Security. ACM, November 2000.

[28] Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Proceedings, 2001 IEEE Symposium on Security and Privacy*. IEEE CS Press, May 2001. Preprint at `http://iacr.org/2000/066.ps.gz`.

[29] Steve Schneider. Verifying authentication protocols with CSP. In *Proceedings of the 10th IEEE Computer Security Foundations Workshop*, pages 3–17. IEEE Computer Society Press, 1997.

[30] Victor Shoup. On formal models for secure key exchange. Research Report RZ 3120 (#93166), IBM Research, April 1999. A revised version 4, dated November 15, 1999, is available from `http://www.shoup.net/papers/`.

[31] F. Javier Thayer Fábrega, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(2/3):191–230, 1999.

[32] Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–79, 1981.

[33] Thomas Y. C. Woo and Simon S. Lam. Verifying authentication protocols: Methodology and example. In *Proc. Int. Conference on Network Protocols*, October 1993.

A Strands and the Adversary

In this appendix, we define the basic strand space notions used in the body of the paper. This material is derived from [31, 17].

A.1 Strand Spaces

Consider a set A, the elements of which are the possible messages that can be exchanged between principals in a protocol. We will refer to the elements of A as *terms*. We assume that a *subterm* relation is defined on A. $t_0 \sqsubset t_1$ means t_0 is a subterm of t_1. We also assume that A has a concatenation operator $\cdot$ and possibly also a cryptographic operator. We write $\{\!|t|\!\}_K$ for the result of applying the cryptographic operator to t using the secret K.

In a protocol, principals can either send or receive terms. We represent transmission of a term as the occurrence of that term with positive sign, and reception of a term as its occurrence with negative sign.

Definition A.1 *A* signed term *is a pair* $\langle \sigma, a \rangle$ *with* $a \in \mathsf{A}$ *and* σ *one of the symbols* $+, -$. *We will write a signed term as* $+t$ *or* $-t$. $(\pm\mathsf{A})^*$ *is the set of finite sequences of signed terms. We will denote a typical element of* $(\pm\mathsf{A})^*$ *by* $\langle\, \langle \sigma_1, a_1 \rangle, \ldots, \langle \sigma_n, a_n \rangle \,\rangle$.

A strand space *over* A *is a set* Σ *together with a trace mapping* $\mathrm{tr} : \Sigma \to (\pm\mathsf{A})^*$.

By abuse of language, we will still treat signed terms as ordinary terms. For instance, we shall refer to subterms of signed terms. We will usually represent a strand space by its underlying set of strands Σ.

Definition A.2 Fix a strand space Σ.

1. A *node* is a pair $\langle s, i \rangle$, with $s \in \Sigma$ and i an integer satisying $1 \leq i \leq \mathrm{length}(\mathrm{tr}(s))$. The set of nodes is denoted by $\mathcal{N}$. If $n = \langle s, i \rangle \in \mathcal{N}$ then $\mathrm{index}(n) = i$ and $\mathrm{strand}(n) = s$. Define $\mathrm{term}(n)$ to be $(\mathrm{tr}(s))_i$, i.e. the ith signed term in the trace of s.

2. There is an edge $n_1 \rightarrow n_2$ if and only if $\mathrm{term}(n_1) = +a$ and $\mathrm{term}(n_2) = -a$ for some $a \in \mathsf{A}$. Intuitively, the edge means that node n_1 sends the message a, which is received by n_2, recording a potential causal link between those strands.

3. When $n_1 = \langle s, i \rangle$ and $n_2 = \langle s, i+1 \rangle$ are members of $\mathcal{N}$, there is an edge $n_1 \Rightarrow n_2$. Intuitively, the edge expresses that n_1 is an immediate causal predecessor of n_2 on the strand s.

4. Suppose I is a set of unsigned terms. The node $n \in \mathcal{N}$ is an *entry point* for I iff $\mathrm{term}(n) = +t$ for some $t \in I$, and whenever $n' \Rightarrow^+ n$, $\mathrm{term}(n') \notin I$.

5. An unsigned term t *originates* on $n \in \mathcal{N}$ iff n is an entry point for the set $I = \{t' : t \sqsubset t'\}$.

6. An unsigned term t is *uniquely originating in* a set of nodes $S \subset \mathcal{N}$ iff there is a unique $n \in S$ such that t originates on n.

7. An unsigned term t is *non-originating in* a set of nodes $S \subset \mathcal{N}$ iff there is no $n \in S$ such that t originates on n.

A.2 Bundles and Causal Precedence

A *bundle* is a finite subgraph of the graph $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow)\rangle$, for which we can regard the edges as expressing the causal dependencies of the nodes.

Definition A.3 *Suppose $\rightarrow_\mathcal{C} \subset \rightarrow$; suppose $\Rightarrow_\mathcal{C} \subset \Rightarrow$; and suppose $\mathcal{C} = \langle \mathcal{N}_\mathcal{C}, (\rightarrow_\mathcal{C} \cup \Rightarrow_\mathcal{C})\rangle$ is a subgraph of $\langle \mathcal{N}, (\rightarrow \cup \Rightarrow)\rangle$. $\mathcal{C}$ is a bundle if:*

1. *$\mathcal{N}_\mathcal{C}$ and $\rightarrow_\mathcal{C} \cup \Rightarrow_\mathcal{C}$ are finite.*
2. *If $n_2 \in \mathcal{N}_\mathcal{C}$ and term(n_2) is negative, then there is a unique n_1 such that $n_1 \rightarrow_\mathcal{C} n_2$.*
3. *If $n_2 \in \mathcal{N}_\mathcal{C}$ and $n_1 \Rightarrow n_2$ then $n_1 \Rightarrow_\mathcal{C} n_2$.*
4. *$\mathcal{C}$ is acyclic.*

In conditions 2 and 3, it follows that $n_1 \in \mathcal{N}_\mathcal{C}$, because $\mathcal{C}$ is a graph.

Definition A.4 *If $\mathcal{S}$ is a set of edges, i.e. $\mathcal{S} \subset \rightarrow \cup \Rightarrow$, then $\prec_\mathcal{S}$ is the transitive closure of $\mathcal{S}$, and $\preceq_\mathcal{S}$ is the reflexive, transitive closure of $\mathcal{S}$.*

The relations $\prec_\mathcal{S}$ and $\preceq_\mathcal{S}$ are each subsets of $\mathcal{N}_\mathcal{S} \times \mathcal{N}_\mathcal{S}$, where $\mathcal{N}_\mathcal{S}$ is the set of nodes incident with any edge in $\mathcal{S}$.

Proposition A.5 *Suppose $\mathcal{C}$ is a bundle. Then $\preceq_\mathcal{C}$ is a partial order, i.e. a reflexive, antisymmetric, transitive relation. Every non-empty subset of the nodes in $\mathcal{C}$ has $\preceq_\mathcal{C}$-minimal members.*

A.3 Adversary Strands

The actions available to the adversary in the abstract Dolev-Yao model are relative to the set of keys that the adversary knows initially. We encode this in a parameter, the set of adversary keys $\mathsf{K}_\mathcal{P}$.

Definition A.6 *A* adversary trace *relative to $\mathsf{K}_\mathcal{P}$ is one of the following:*

M_t *Text message: $\langle +t\rangle$ where $t \in \mathsf{T}$*

K_K *Key: $\langle +K\rangle$ where $K \in \mathsf{K}_\mathcal{P}$*

$\mathsf{C}_{g,h}$ *Concatenation: $\langle -g,\ -h,\ +g{\cdot}h\rangle$*

$\mathsf{S}_{g,h}$ *Separation: $\langle -g{\cdot}h,\ +g,\ +h\rangle$*

$\mathsf{E}_{h,K}$ *Encryption:* $\langle -K,\ -h,\ +\{\!|h|\!\}_K\rangle$

$\mathsf{D}_{h,K}$ *Decryption:* $\langle -K^{-1},\ -\{\!|h|\!\}_K,\ +h\rangle$

$\mathcal{P}_\Sigma$ is the set of all strands $s \in \Sigma$ such that $\mathrm{tr}(s)$ *is an adversary trace.*

A strand $s \in \Sigma$ is a adversary strand *if it belongs to $\mathcal{P}_\Sigma$, and a node is a* adversary node *if the strand it lies on is an adversary strand. Otherwise we will call it a* regular *strand or node.*

B Carter-Wegman Hash Functions

We now recall the concept of Carter-Wegman universal classes.

Definition B.1 *$\phi : X \to Y$ is* uniformly distributed *iff ϕ maps the uniform distribution on X to the uniform distribution on Y. Thus,*

$$\frac{\mathsf{card}\big(\phi^{-1}(A)\big)}{\mathsf{card}(X)} = \frac{\mathsf{card}(A)}{\mathsf{card}(Y)}$$

for every $A \subseteq Y$.

Alternatively, ϕ is uniform iff the inverse image of each $y \in Y$ has cardinality $\mathsf{card}(X)/\,\mathsf{card}(Y)$.

For any $\phi : X \to Y$, X is the disjoint union of the sets $\phi^{-1}(y)$ for $y \in Y$. Uniform distribution means that all these sets have the same cardinality. Intuitively, uniformly distributed maps decompose X as a "product" $Y \times H$.

Example B.2 *Let V, W be finite dimensional vector spaces over the finite field $\mathbb{F}_q$. An linear map $T : V \to W$ is uniformly distributed iff it is surjective. This will be the case iff $\dim V - \dim\ker T = \dim W$.*

Proof. If T is surjective and $w \in W$, then $T^{-1}(w)$ is an subspace of dimension $T^{-1}(0)$.

Definition B.3 *A set of functions $\mathcal{F} \subseteq Y^X$ is n-strongly universal iff* $\mathsf{card}(X)$ *is at least n and for any pairwise distinct $x_1, \ldots, x_n \in X$, the evaluation mapping*

$$f \mapsto \langle f(x_1), \ldots, f(x_n)\rangle$$

is uniform. Equivalently, for pairwise distinct $x_1, \ldots, x_n \in X$

$$\mathsf{P}\{f \in \mathcal{F} : \langle f(x_1), \ldots, f(x_n)\rangle = \langle y_1, \ldots, y_n\rangle\} = \frac{1}{(\mathsf{card}\, Y)^n}$$

The definition requires that the $x_1, \ldots, x_n$ be pairwise distinct. If some of the x_i's coincide, then $\langle f(x_1), \ldots, f(x_n) \rangle$ lies on a proper subspace of Y^n, in which case the evaluation mapping is non-uniform.

Example B.4 *If $q \geq n$, the space of polynomial functions $p : \mathbb{F}_q \to \mathbb{F}_q$ with $\deg(p) \leq n-1$ is n-strongly universal. This follows from linearity of the evaluation mapping $p \mapsto (p(\theta_1), \ldots, p(\theta_n))$ and Lagrange interpolation.*

As a special case, the space of affine mappings $x \mapsto ax + b$ on finite fields is 2-strongly universal.

Note that the usual definition of n-strong universality does not require that $\mathsf{card}(X)$ be at least n. However, without this assumption, the following lemma fails.

Lemma B.5 *If $\mathcal{F} \subseteq Y^X$ is n-strongly universal then $\mathcal{F}$ is m strongly universal for $m \leq n$.*

PROOF. If $x_1, \ldots x_m$ are pairwise distinct, extend to a pairwise distinct sequence $x_1, \ldots x_n$, which exists since $\mathsf{card}(X)$ is at least n, and use the fact the composition of uniform mappings is uniform. ■

Contents

1 Introduction **1**

2 Strand Spaces **5**
2.1 Strands: Basic Notions 5
2.2 Cryptoalgebras 7
2.3 Representing Protocols in Strand Spaces 9

3 Bundles and Skeletons **11**
3.1 Security Properties 12
3.2 Dolev-Yao Realizability 12

4 Adversary Strategies **14**
4.1 Stochastic Elements 14
4.2 Strategies 14
4.3 Model Assumption 15
4.4 The Success Set for a Strategy 16

5 Pure Authentication Protocols **18**
5.1 Forgery 18
5.2 Pure Authentication Protocol Model Assumptions 19
5.3 The Probability of Forgery 20
5.4 Likelihood of Anomalies 21
5.5 Achieving Assumption 5 22

6 Conclusion and Related Work **23**

A Strands and the Adversary **28**
A.1 Strand Spaces 28
A.2 Bundles and Causal Precedence 30
A.3 Adversary Strands 30

B Carter-Wegman Hash Functions **31**