Process Theory: Strands

A Tutorial Presented by:
Brian Kellogg and Jack G. Nestell

Informal Discussion on Formal Methods

- It is critical that one understand terms, syntax, and semantics. Survey.
- At this point, for our presentations, it is critical to understand the ideas and utilization, and differences, between each FM, specification languages, and support tools.
- How are these FM’s currently being used? What is potential for the “real-world”?
Informal Discussion on Formal Methods

• Two different worlds of computer science: industry/commercial and academia.
  • Industry helps determine what the needs are and how they can/will be used.
  • Academics attempts to develop new technologies for needs.

Interest:
  • Know how the “cool stuff” can be applied to industrial world, especially IT domain.
  • Why and Who cares about something (technology) that has no potential of ever doing anything for folks like me and the companies we work for?

• Two different ways to approach our discussion:
  • Build support for what our FM’s and specification language is capable of or defend the specification language and stating that it is worthy?
  • FM’s and specification languages can used to prove security protocols under a given dynamic and often unique environment.

Informal Discussion on Formal Methods

• Due to the dynamic and unique environment, FM’s need to be put to the test by the authors in a real-world proof, not by a benchmark.
  • I know the value of benchmarks in my world, the computer networking world. I was hoping that the authors actually proved how well and how valuable Formal Methods are, by testing, validating, and proving the correctness of some neat, new, security protocol. This, to me, is what I crave.
  • Ultimately the language has to be used in a real world if it is to prove the correctness of real world protocols.
  • Our job here as researchers, scholars and academics is to prove the correctness of real-world protocols that will keep my company that I work for and our partners from having performance, fault-tolerant and security issues and breaches.

Informal Discussion on Formal Methods

What has it done and where is it used, FM’s have to be proven and designed as well. How was this benchmark designed? Who designed it? What security protocol is it modeled after? Is it modeled after an existing security protocol at all? If not, what good is the benchmark? If not, what good is the specification language? If so, tell me more. Convince me that the purpose of the specification language and the benchmark and therefore the security protocol that my companies uses on our Extranet and VPN’s is all good. Convince me that I am doing my job by using this protocol solution.

“I am not saying to go into detail on a complicated security protocol. But, using it against a benchmark is boring me. Let’s face it, you don’t dissect something at this level to have it be forgotten and deemed useless. I understand the value of a good benchmark and I also understand its purpose. We are studying/testing FM’s, not protocols. Or are we? Or, visa versa? A security protocol benchmark is not something that is used to protect my company and our partners from the real-world. Then what good is it? It is good to test our FM, but for our FM to be proven doesn’t it have to be tested against “real-world” data in a real-world environment? That is my point. I just want to take our studies a step further. Don’t tell me just how (I’ll learn that in text books)............tell me who, where, when, why and for what reasons.”

• This “issue” is much more complex than it may first appear. How are FM’s as applied to network security relate to its usefulness and intent in the world of IT management and industry ............

CONCLUSION: Real world case studies.
Strands Introduction:

- Security protocols are often found later to have flaws.
- Therefore, goal of any FM:
  - find errors in bad protocols and
  - prove correct those protocols that contain no flaws.

- Strand:
  - represents a sequence of events
  - for a particular participant
  - in a particular protocol run.

Strand Space Model: represent the problem domain as:

- SSM Advantage:
  - contains the exact casual participant relation information.
  - derive simply proofs of a protocols correctness.

Proving or disproving a protocol:

- must be expressed in terms of
  - interactions and connections
  - between strands of different kinds.

Assumptions Made by Authors

- Certain data items (nonces and keys) are fresh and a race in more than one run of protocol
- Work with an explicit model of potential penetrator actions
- A strand space models the assumption that some values are impossible for a penetrator to guess.
- A term(message) must be sent before it can be received.
- Keys will be invented only once during the life time of the protocol: protocol depends on the "freshness" and validity of the data, keys, and nonces.
- Only one strand originates uniquely a term when it can play the role of a nonce or session key.

Goal of this tutorial:
- to demonstrate how this method allows one to clearly observe why the Protocol is correct, and the assumptions required.
Strands Concepts:

- **Term**-used to represent the messages in a protocol.
  - **Text terms** (®): principal names, nonces, or data (bank account numbers)
  - **Key terms** (©): a set of keys disjoint from ®.
- **Actions**: The set of actions Act that protocol principals follow throughout the execution of a protocol.
  - Include send(+) and receive(-).
- **Events**: A pair (action, a) where action ∈ Act, and a ∈ A is the argument of the action from the set of terms.
  - +A and (+A)*.

Strands Concepts:

- **Protocol**: defines the sequence of events/rules for each principal’s role.
- **Strand**: A linear structure, the sequence on a principal’s message ‘sends’ and ‘receives’.
- **Strand Space**: Contains all the legitimate executions of the Protocol expected within it’s useful lifetime, together with penetrator Strands.

Penetrator Strand Concepts:

- **Penetrator strand**: term transmissions and receptions that model a basic capability that one may assume the penetrator possees.
- **Penetrator Strands include**:
  - Obtaining a symmetric key and a term encrypted using that key, and then being able to send the result of decrypted message.
  - Obtaining two terms and sending the result of concatenating them.
  - Sending a data item that the penetrator strand may know.
- **Penetrator actions**:
  - modeled by connecting different penetrator strands.
Bundle Concepts:

- **Bundles:** "Agreement on the act of communication."
- A portion of a strand space.
- Hooked together strands
  - where one strand sends a message to another and
  - another receives that same message.

- **IMPORTANT Point:**
  - Reentrant strands may be included in a bundle of a correctly
    proven protocol.
  - However they should not keep legitimate parties from agree on
    data values or protecting the secrecy of the values Chosen.
  - *Why is this?*

- **As compared to strand, a Bundle**
  - is a graph-structured entity.
  - representing the communication between a number of Strands.

Bundle Concepts: (cont’d)

- **Why do we have/need a linear and a graph-structured representation of strands?**
  - To validate the legitimacy of the strands by understanding how they interact with other participants.

- **Under the conditions of a Strand concept:**
  - Proving or disproving a protocol must be expressed in terms of the interactions and connections between strands of different kinds.

Bundle Example:
Syntax, Semantics, and Definitions

- Set A, elements that are possible messages to be exchange between principles.
- T is a term, means t is a subterm of t.
- Transmission (or occurrence) of term: ‘+’ sign and reception with ‘-‘.

**Definition:** A signed term is a pair (σ, α) with α ∈ A and σ one of the symbols ‘+’ or ‘-‘.
- Signed term can be written as +t or -t.
- (+A) is the set of finite sequences of signed terms.

**Definition:** A strand space over A (possible message) is a set Σ together with a trace mapping
  * tr: Σ → (+A)∗.

 Syntax, Semantics, and Definitions

**Definition:** For a fixed strand space:
- A node is a pair (s, l), with s ∈ Σ and l is an integer satisfying 1 ≤ l ≤ length(tr(s)).
  - Set of nodes denoted by N.
- If n = (s, l) ∈ N then index(n) = l and strand(n) = s.
- There is an edge n₁ – n₂ if and only if:
  - term(n₁) = +a and term(n₂) = a for some a ∈ A.
  - n₁ sends a message to n₂,
  - recording a casual link between two strands.
- When n₁ = (s₁, l₁) and n₂ = (s₂, l₂ + 1) are members of N,
  - there is an edge n₁ → n₂.

Syntax, Semantics, and Definitions (cont'd)

- An unsigned term t occurs in n ∈ N iff t ∈ ω term(n).
- Set ω is a set of unsigned terms. The node n ∈ N is an entry point for t iff term(n) = t with some t ∈ ω and whenever n₁ → n₂, term(n₁) ∈ ω.
- An unsigned term t originates on n ∈ N iff n is an entry point for the set
  I = {t; t ∈ ω}. 
  Plays the role of nonce or session key.
- An unsigned term t is uniquely originating iff t originates on a unique n ∈ N.

If a term t originates uniquely in a specific strand space, then it is allowed to be a session key or nonce.
Syntax, Semantics, and Definitions

**Bundles:**
Let \( E \) be the set of edges, and let \( N_e \) be the set of nodes incident with any edge in \( E \). If \( E \) is a bundle:
- \( E \) is finite
- If \( n_1 \in N_e \) and \( \text{term}(n_1) \) is negative, then there is a unique \( n_2 \in N_e \) such that \( n_1 \rightarrow n_2 \).
- If \( n_1 \in N_e \) and \( n_1 \rightarrow n_2 \), then \( n_1 = n_2 \).
- \( E \) is acyclic (open structure).

Formalizes strand process communication model with three properties:
- A process (strand) may send or receive a message, but not concurrently.
- When a strand receives a message \( m \), there is a unique node transmitting \( m \) from which the message was immediately received.
- When a strand transmits a message \( m \), many strands may immediately receive \( m \).

---

**Definition:** Let \( S \) be a set of edges, i.e., subset of the union of \( \rightarrow \) and \( \rightarrow_r \). Then:
- \( \leq \) is the transitive closure of \( S \), and
- \( S \) is the reflexive, transitive closure of \( S \).

**Where**
- \( \leq \) sequence of one or more edges,
- \( \leq_r \) sequence of zero or more edges
- \( ( \leq_r ) \) (partial ordering, not a complete relation).

**Lemma:** Suppose \( E \) is a bundle. Then \( \leq \) is a partial order, i.e., a reflexive, antisymmetric, transitive relation.
- Every non-empty subset of the nodes in \( E \) has \( \leq \) minimal members.
- “What did he know and when did he know it?”

---

**Lemma:** Suppose \( E \) is a bundle, and \( S \subseteq E \) is a set of nodes such that \( \text{un_s}(m) = \text{un_s}(m') \) implies that:
- \( m \rightarrow n \) if \( m' \rightarrow n \), for all nodes \( n \).
- \( n \) is a \( S \)-minimal member of \( S \), then the sign of \( n \) is positive.

**Proof:** If \( \text{term}(n) \) were negative, then by the bundle property,
- \( n \rightarrow \) for some \( n' \in E \) and sign apart, \( \text{term}(n) \rightarrow \text{term}(n') \).
- Hence, \( n \leq \) \( n' \), violating the minimality property of \( n \).

**Lemma:** Suppose \( E \) is a bundle, \( t \in A \) and \( n \in E \) is a \( S \)-minimal element of \( ( m \in E : t \in \text{term}(m) ) \).
- The node \( n \) is an originating occurrence for \( t \).

**Proof:** By Lemma 2.7, the sign of \( n \) is positive.
- If \( n \leq \) \( n' \) and \( n' \) is in the strand of \( t \):
  - then \( n' = n \), therefore by the Minimality property of \( n, t \in \text{term}(m) \).
  - Thus \( n \) is originating for \( t \).
Penetrator Strands

Ideals:
• A method to prove additional bounds on the abilities of the penetrator.
• A penetrator's effectiveness relies on:
  • a set of keys known initially to the penetrator and
  • a set of strands that allow the penetrator to create
    new messages from compromised messages.

Atomic actions that a penetrator can invoke are encoded in a set of
penetrator traces.

Penetrator Strands

Definition: Penetrator traces:

M. Strand has the form \( (+t) \) where \( t \in T \), but \( K \) is not a subterm of \( t \).
F. Flushing: Strand has the form \( (-q) \) and therefore lacks any positive
   nodes.
T. Tree strand has the form \( (-g,-g,-g) \), so no value originates on the
   positive node.
C. Concatenation: Strand has the form \( (-g,-h,-h) \) so no value originates on the
   positive node.
S. Separation into components: \( (-g,-h,-h) \), so no value originates on the
   positive node.
K. Key: \( (+K) \) where \( K \in K_2 \).
E. Encryption: \( (+K,-h,-h) \), Therefore, no key can occur in the positive
   node with having occurred in a previous node.
D. Description: \( (+K^2,-h,-h,-h) \), Therefore, no key can occur in the
   positive node \( w_i \) having occurred in a previous node.

Definition: An infiltrated strand space is a pair \((\Sigma, P)\) with \( \Sigma \) a strand
space and \( P \subseteq \Sigma \) such that \( t(p) \) is a penetrator trace for all \( p \in P \).

Example: Needham-Schroeder
Protocol

1. \( A \rightarrow B (K_a R_a) \)
2. \( A \rightarrow B (K_a R_a K_b) \)
3. \( A \rightarrow B (K_a) \)

Note: The original protocol has 7 steps. This is
reduced to 3 steps by removing communication
to a key server.
Flaw in Needham-Schroeder Protocol

- Introduce an Intruder I
- A establishes a session with I
  - (A→I, (N_A)_{P(I)})
- I establishes a session with B using A’s info
  - (I→B, (N_A)_{P(B)})
- B responds to I
  - (B→I, (N_B, N_A)_{P(A)})
- I uses A as an oracle
  - (I→A, (N_A, N_B)_{P(A)})
- A returns N_A to I
  - (A→I, (N_A)_{P(I)})
- I decrypts N_A and returns it to B
  - (I→B, (N_A)_{P(B)})
- B now believes it has successfully run protocol with A

Needham-Schroeder-Lowe Protocol

![Diagram of the Needham-Schroeder-Lowe Protocol]

NSL Strand Spaces

- Σ is an infiltrated NSL space if Σ is the union of three types of strands:
  1. Penetrator strands s ∈ P
  2. Initiator strands with trace s = Init(A, B, N_A, N_B) defined as: <++; (N_A), x, (N_B)_{y}, y, (N_B)_{z}> where A, B are e T_{Name}, N_A, N_B ∈ T but N_A ∈ T_{Realm}.
     - Initiator A sends its identity and a nonce using B’s public key,
     - A then receives a message that contains its own nonce, B’s nonce and B’s identity encrypted with A’s public key.
     - Finally, A sends B’s nonce encrypted with B’s public key.
  3. Responder strands with trace s = Resp(A, B, N_A, N_B) defined as: <--; (N_A), y, (N_B)_{x}, y, (N_B)_{z}> where A, B are e T_{Name}, N_A, N_B ∈ T but N_A ∈ T_{Name}.
     - English same as above but from B’s perspective.
Example: Responder’s Guarantee

- Suppose the following:
  - $\Sigma$ is an NSL space, $C$ is a bundle in $\Sigma$, and $s$ is a responder strand in $\text{Respl}(A, B, N, N_c)$
  - The private key of $a$ is not held by the penetrator
  - $K_c = K_c$
  - $N_a = N_a$ and $N_b$ is uniquely originating in $\Sigma$

- What we want to prove:
  - $C$ contains the initiator’s strand
    - $t \in \text{Init}(A, B, N, N_a)$

Responder’s Guarantee Continued

- A few definitions:
  - Node $<s, 2>$ outputs the value $(N_a B)_{KA}$ and is referred to as node $n_0$ with term $v_0$
  - Node $<s, 3>$ outputs the value $(N_a)_{K_b}$ and is referred to as node $n_1$ with term $v_1$
  - Nodes $n_a$ and $n_b$ will be introduced in the proof such that:
    - $n_a < n_b < n_c < n_1$

Responder’s Guarantee: First Lemma

- **Lemma 1**: $N_a$ originates at $n_0$
  - By assumptions made on last slide $N_a \notin \sigma_b$ and $n_b$ has a positive sign, so we only need to check that $N_a \in \gamma$, where $\gamma$ is the node $<s, 1>$ that comes before $n_0$ on the same strand.
  - **Proof**:
    - $\gamma' = (N_a A)_{K_b}$ so we just need to make sure that $N_b \neq N_0$ and $A = N_0$
    - $N_b \neq N_0$ is part of the hypothesis
    - $A = N_0$ is true because in the definition $N_b \notin \gamma_{same}$
**Figure for Lemma 1**

- $v_0 \rightarrow (N_0 B)_{\kappa_0}$
- $n_2 \cdots n_k \cdots \rightarrow \{N_i\}_{\kappa_i}$
- $v_1 \rightarrow (N_1 B)_{\kappa_1}$
- $v_1 \rightarrow (N_1 B)_{\kappa_1}$
- $n_k \rightarrow \{N_i\}_{\kappa_i}$

**Responder’s Guarantee: Second Lemma**

- **Lemma 2**: Establish that the crucial step is taken by a regular strand and not a penetrator strand such that set
  
  $S = \{ n \in C : N_b \subseteq \text{term}(N) \wedge v_0 \notin \text{term}(n) \}$
  
  has a $\leq$-minimal node $n$, that is a regular strand with a positive sign.

- **Proof**:
  - We first establish that $S$ is not an empty set
    - Since $n \in C$ and $n$ contains $N_b$
    - But it is not a subterm of $v_0$, $S$ is non-empty
  - Due to lemmas stated earlier, $S$ has at least a minimal element $n$, and it is positive.

**Second Lemma Continued**

- **Now confirm $n_k$ is a normal strand as opposed to a penetrator strand**
  - $M. <t>$ where $t \in T$.
  - For NSL, it has trace $tr(p)$ in the form of $<t>$ so $t = N_0$.
  - This implies $N_0$ originates on this strand
    - But this is a contradiction of our first lemma.
  - $F$. The trace $tr(p)$ has the form $<-g>$.
    - But we have positive nodes so this is not valid.
  - (etc.) Many other atomic actions a penetrator has available must be proven false.
Outline of Rest of Proof

The remainder of the proof is similar to the above adding more definitions and lemmas.

- It goes on to prove n1 precedes n2 on strand t and that t is the initiator strand.
- It also shows that term n1 contains (N_a N_b)_{kA}.
- This satisfies what we originally wanted to prove:

\[ C \text{ contains the initiator's strand} \]
\[ t \in \text{Init}[A, B, N_a, N_b] \]

Tool Support: Athena

- Implemented using Standard ML (Meta Language) of New Jersey
  - ML is a functional programming language written in the 1980's by the Laboratory for Foundations of Computer Science (LFCS)
  - Attractive language for formal methods since it is "safe" i.e. all bugs are caught at compile time or handled gracefully at run time
- Input consists of a protocol description and a set of security properties. You can also specify additional pruning theorems if available.

Tool Support: Athena

- Athena extends the Strand Space Model and uses model checking and theorem proving approaches
  - Able to determine error in the Needham-Schroeder protocol in a fraction of a second
- Example of extension: Adds in the notation of an intern relation \( \prec \), such that \( a_1 \) is an intern of \( a_2 \) if \( a_1 \) can be extracted from \( a_2 \) without needing decryption.
- Paper cites reference for information on Athena, but the link only links to the authors papers and research interest.
  - Other than a link to the paper on Athena, there is no place to download this tool.
Strands as applied to real-world

Need more case studies

Conclusion

**Strand Spaces**
- Designed for proving correctness of protocol.
- Allows one to use mathematical methods to justify protocols.

References

1. F. Javier Thayer Pineda, Jonathan C. Herzog, and Joshua D. Gutfreund
   "Strand Spaces: Proving Security Protocols Correct"; the MITRE Corporation, 1998 IEEE

2. F. Javier Thayer Pineda, Jonathan C. Herzog, and Joshua D. Gutfreund
   (1999) pages 193-235


4. F. Javier Thayer Pineda, Jonathan C. Herzog, and Joshua D. Gutfreund
   "Formal Model of Strand Space"; The MITRE Corporation
