Adventures in Cryptographic Protocol Analysis

Jonathan K. Millen
The MITRE Corporation
ACM CCS, October 2010

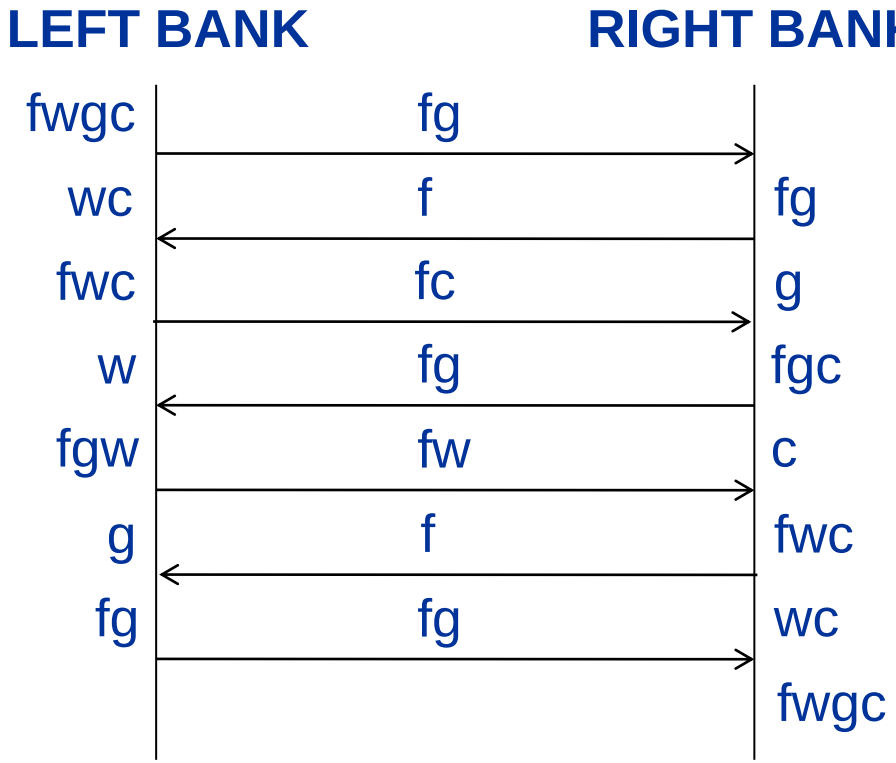
Protocol Security Analysis

- Protocol security analysis has a long and diverse history
- Rather than a survey, let's look at why it was enjoyable to study this kind of problem, and some unconventional approaches to it
- Protocol security analysis is like a logical puzzle...
  - We can use this to understand the power, the demands, and the limits of protocol security analysis

Example: Wolf, Goat and Cabbage

- Also called Fox, Goose, and Corn; or Fox, Chicken, Grain
- There's a river
- A farmer is on one bank with the three possessions and a boat
- The boat is only big enough for the farmer and one possession
- The wolf should not be left alone with the goat, or the goat with the cabbage
- How can the farmer get all his possessions across the river?

The Answer: a Protocol?



f: farmer
w: wolf
g: goat
c: cabbage

Protocol Analyzers Can Solve This

- Protocol analysis includes the ability to solve state reachability problems

- How can this be illustrated?
  - Encode the WGC problem as a protocol
  - Encode the constraints and the goal, but not the answer

WGC Encoding

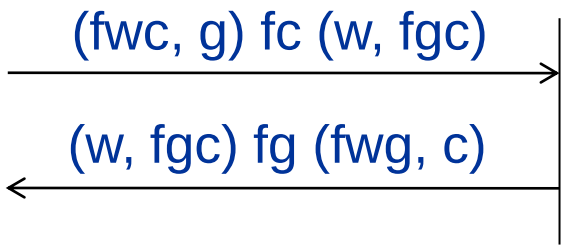
- Basic idea: the global state shows where things are
- In particular, we need:
  - initial state: (fwgc, n)
  - goal state: (n, fwgc)
- All legal states satisfy the constraint that the goat is not left alone with the wolf or cabbage
- How can we get from the initial to final state? (reachability)
- Legal transitions show what goes across in the boat
  - must be the farmer and optionally one possession
  - left to right: (fwgc, n) fg (wc, fg)
  - right to left: (w, fgc) fg (fwg, c)
- Crucial observation: successive transitions link...

Linking Transitions into Roles

(fwc, g) fc (w, fgc) *may be followed by*

(w, fgc) fg (fwg, c)

This link is a short protocol **role**:



- A *role* is executed by one principal (left or right bank)
- A role is a sequence of send and receive actions
- The transitions are the messages
- We will have a role for every possible linked pair of transitions

Connecting Roles into a Path

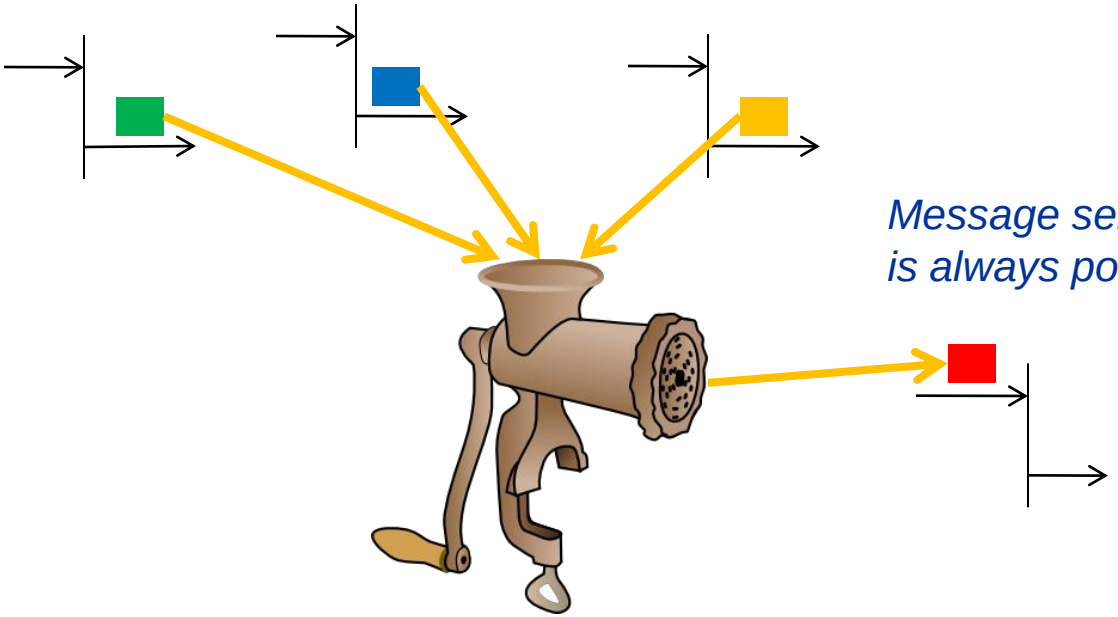
- There are 24 roles in all (12 on the left, 12 on the right)
  - The initial role sends (fwgc, n) fg (wc, fg)
  - The final role receives (fg, wc) fg (n, fwgc)

- The state reachability problem is to connect the initial role to the final role

- Longer sequences occur as sent messages are delivered
  - In a hostile network, message delivery is subject to attacker interference

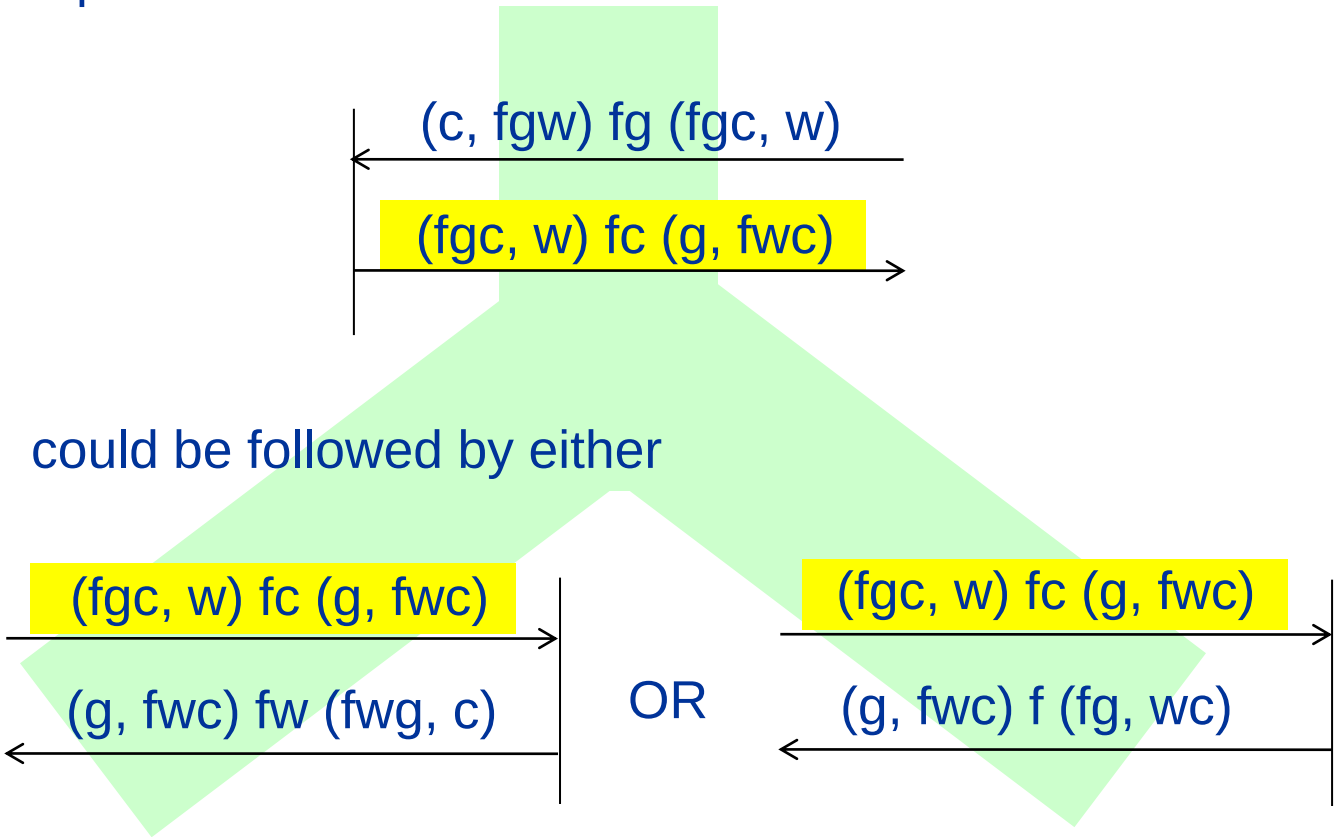
The Network Security Threat

- The "attacker" represents the threat from a hostile network
- Messages may be intercepted, forged, and/or re-routed
- Protocol analysis must define and explore the possibilities
- Messages are (partially) ***ordered*** in such a way that every received message is derivable from previously sent messages
  - Derivability is based on a specification of attacker operations



Alternative Paths

- Example:



Solution With a Protocol Analyzer

- An experiment was carried out with CoProVe[1]
  - by Corin and Etalle (http://es.ewi.utwente.nl/coprove)
  - (an improvement on Millen-Shmatikov Constraint Solver, CCS 2001)
- Input required:
  - scenario(*list of agents*)
    an agent is a named role or role instance
  - initial_intruder_knowledge(*list of constants*)
    not needed for this problem
  - has_to_finish(*list of agent names*)
- "has_to_finish" is just the final role-transition to (n, fwgc)
  - this was important to limit the search

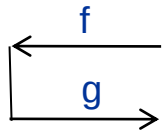
[1]S. Corin and S. Etalle, "An improved constraint-based system for the verification of security protocols", 9th SAS, LNCS 2477, 2002.
Prolog code: http://www.win.tue.nl/~setalle/software/coprove/ and http://homepage.mac.com/j.millen/constraints.html

Role Encoding (details)

- Transition encoding idea:
  - (fgw,c) fg (w,fgc) abbreviates to s(l, gw, g)
  - (farmer on left with wg takes g over; the rest is determined)
  - "s( )" is a constructor operation not forgeable by the attacker, perhaps like a certified signature

- Agent encoding (example)
  - [lfg, [recv(s(r, wc, f)), send(s(l, g, g))]
  - agent name shows the acting principal (l) and items carried (f, g)
  - How many instances of each role?

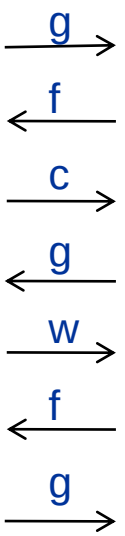
    Need only one agent per role (repeats are loops)



Result

```
4 ?- time(checkone).
SOLUTION FOUND
State: [[init, []], [final, []], [rgf, []], [rcg, []], [rwf, []], [rcf, []], [rwg, []], [rfw, [recv(s(l, g, f)),
send(s(r, wc, w))]], [rfc, [recv(s(l, g, f)), send(s(r, wc, c))]], [rgw, []], [rgc, []], [rwc, []],
[rcw, []], [lfc, []], [lgw, []], [lfg, []], [lwf, []], [lcf, []], [lwg, []], [lcg, []], [lfw, []], [lcw, []], [lwc,
[]], [lgc, []]]
Trace:
[init, send(s(l, wgc, g))]
[rgf, recv(s(l, wgc, g))]
[rgf, send(s(r, g, f))]
[lfc, recv(s(r, g, f))]
[lfc, send(s(l, wc, c))]
[rcg, recv(s(l, wc, c))]
[rcg, send(s(r, gc, g))]
[lgw, recv(s(r, gc, g))]
[lgw, send(s(l, wg, w))]
[rwf, recv(s(l, wg, w))]
[rwf, send(s(r, wc, f))]
[rwc, recv(s(l, wg, w))]
[rwc, send(s(r, wc, c))]
[lfg, recv(s(r, wc, f))]
[lfg, send(s(l, g, g))]
[final, recv(s(l, g, g))]
[final, send(stop)]
[lcg, recv(s(r, wc, c))]
...

% 39,623 inferences, 0.00 CPU in 0.14 seconds (0% CPU, Infinite Lips)
```



Got it!

extra

Observations on WGC Encoding

- Protocol analyzers include the capacity for state reachability
- The "protocol" devised for this was unconventional
  - Lots of single-transition "roles"
  - Messages encoded current state
- Is this useful, or just for puzzles?
  - What else could we do with this approach?

Undecidability of Protocol Security

- A strategy to show undecidability:
  - Find an undecidable problem expressed as state reachability
  - View the problem as a protocol security problem

    construct the protocol

    broadcast the secret from a particular state

- Heintze and Tygar did this with the Post Correspondence Problem[2]
  - There are several other proofs of protocol security undecidability

[2]N. Heintze and J.D. Tygar, "A Model for secure protocols and their compositions", *IEEE Trans. Softw. Eng.* 22, 1 (Jan. 1996), 16-30.

Post Correspondence Problem

- *PCP: given a finite set of word pairs, does there exist a sequence of choices for which the concatenation of left words equals the concatenation of right words?*

- Example:

  P1 = (a, ab)

  P2 = (b, a)

  P3 = (abc, c)

- This one is solvable:

LEFT			
P1	P2	P1	P3
a	b	a	abc

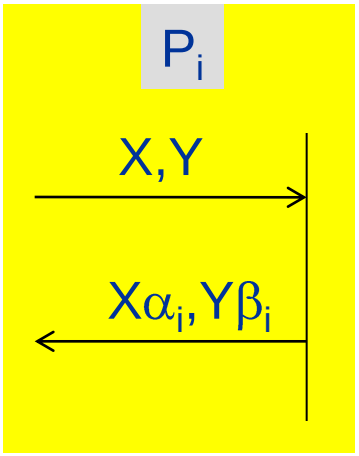
RIGHT			
P1	P2	P1	P3
ab	a	ab	c

- PCP is undecidable because there is no general way to put an upper bound on the length of the sequence

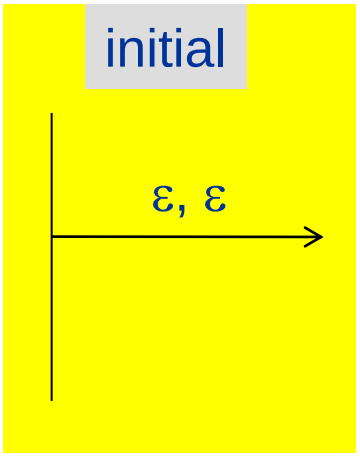
Protocol Security as PCP

- Basic idea: reduce PCP to reachability for a protocol
- Given each pair (α_i, β_i) of words, define a protocol role (a) that concatenates those words on the right of an input pair X, Y (where X, Y are variables)
- Add the initial role (b)
- Add the final role (c) that produces a "secret" on a match

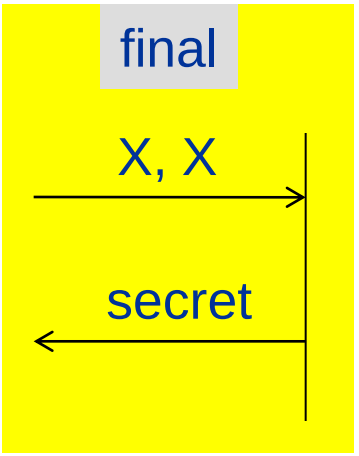
(a)



(b)



(c)



if $X \neq \varepsilon$

(ε is the empty word)

PCP vs. the Attacker/Tool

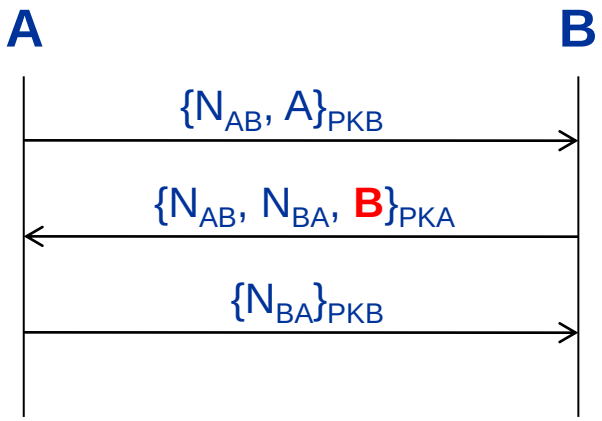
- The attacker is assumed unable to create new pairs (α, β)
  - Otherwise, it could just send (a, a) to the final role
  - we could encode them with an unforgeable function s(α, β)
- As in WGC, the attacker connects role instances to build a path to the final transition
- But it's (much) harder this time, because there are a potentially infinite number of role instances, since X, Y are variable
  - A protocol analysis tool might find a solution, but can't prove there isn't one (otherwise it could solve PCP)
  - Existing tools must impose an instance bound or fail to terminate
  - So protocol security analysis is (in general) undecidable

Two More Examples

- These started out as protocols
- They look more conventional, but they aren't...
- They illustrate challenges for protocol analysis

A Deceptively Normal Example

- Based on the Needham-Schroeder public key protocol (NSPK)
- NSPK has an attack discovered and fixed by Lowe
- Here is the fix: Needham-Schroeder-Lowe[3] (NSL)
- N_{XY} is a "nonce" (could be a session key) created by X to be shared with Y

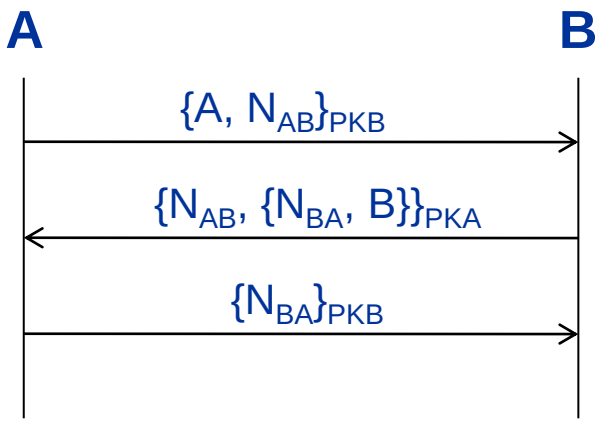


(The red **B** is Lowe's fix)

[3]Lowe, Gavin (November 1995). "An attack on the Needham-Schroeder public-key protocol ". *Information Processing Letters* **56** (3): 131–136

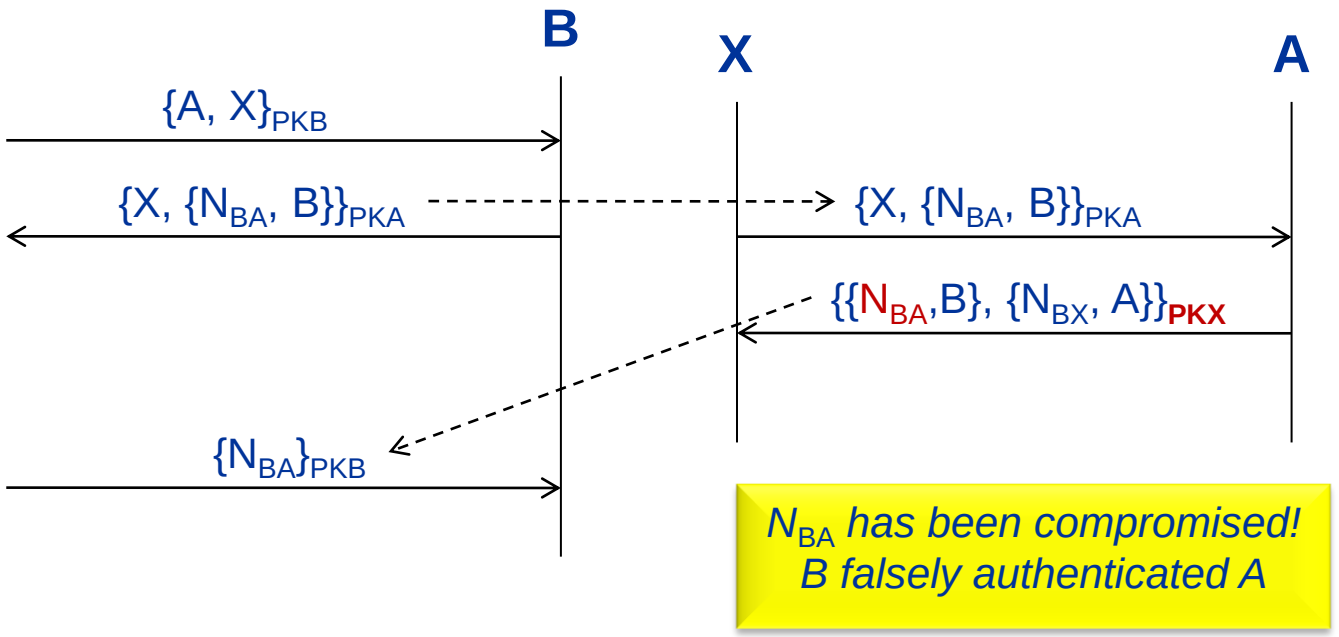
A Slight Variation

- The first message components are reordered
- The triple in the second message is a right-associated pairing



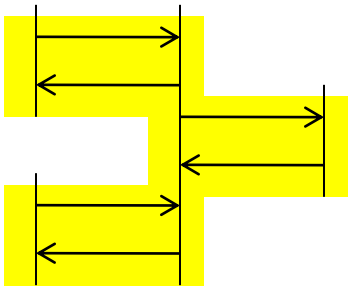
A Type-Confusion Attack

- Suppose X is the attacker's name
- X is used as a nonce in the first message to B
- $\{N_{BA}, B\}$ is viewed as a nonce in the first message to B'
- This was found by the Constraint Solver

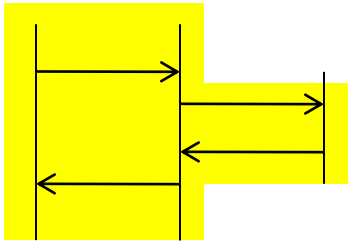


Serial vs. Parallel Attacks

- We have seen examples in which a sequence of role instances led to an "attack" solution
- There are protocols in which multiple instances of the same role are run *concurrently* to have an attack
- When is a parallel attack *necessary*?
- To answer this question, we will use another unconventional protocol



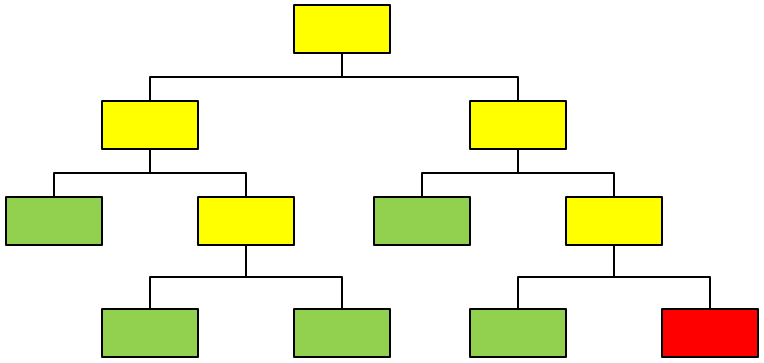
Serial



Parallel

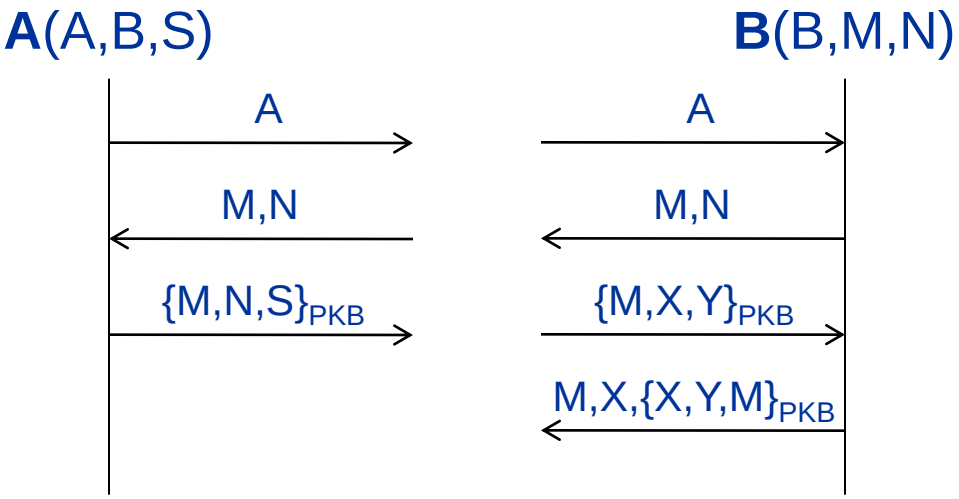
A Necessarily Parallel Attack

- The artificial protocol "ffgg" has only a parallel attack[4]
- Its necessity was shown as follows:
- Start a machine-aided proof of security (using PVS)
- The proof divided into a tree of many cases and subcases
- Security was shown in all but one remaining subcase
- That subcase implied a non-serializable ordering between two instances of the same role



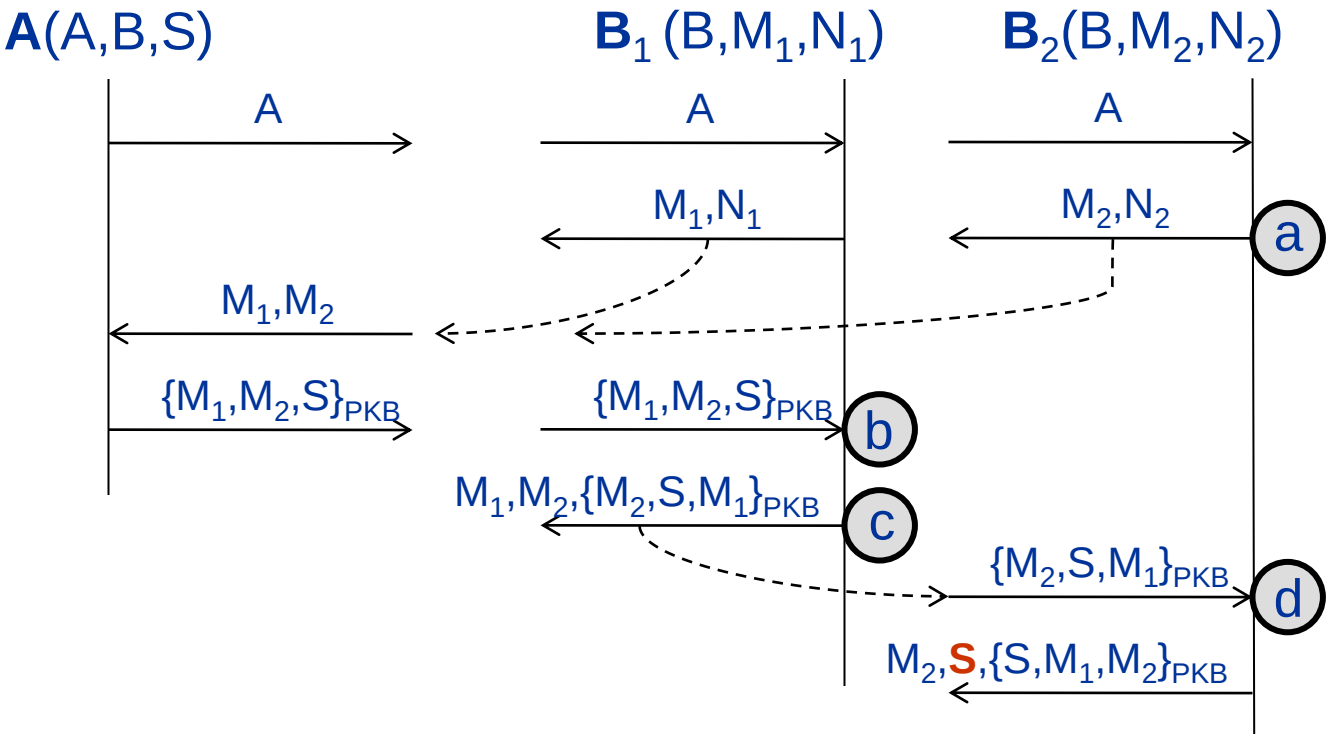
[4]J. Millen, "A necessarily parallel attack", *Proceedings of the Workshop on Formal Methods and Security Protocols* (1999); using PVS: http://pvs.csl.sri.com

The ffgg Protocol



M, N are nonces
PKB is the public key of B
S is supposed to be secret.
Is it revealed?
(Wow, is this a weird protocol)

The Attack



The case that allows this to happen forces a > b > c > d.

This means that B_1 and B_2 are not serializable.

(What about NSL as an example?)

The Security Mindset

- These protocol examples are not "real" protocols
- Yet they are useful to explore the limits of what protocol security analysis can and must accomplish
- Hackers and penetration teams also cultivate a willingness to think "outside the box" and try unconventional inputs to the systems being tested
  - The security mindset is needed for theory and for analysis tool development as well as system security practice