

18732: Secure Software Systems

Software Model Checking for Security I


Anupam Datta

CMU
Fall 2010

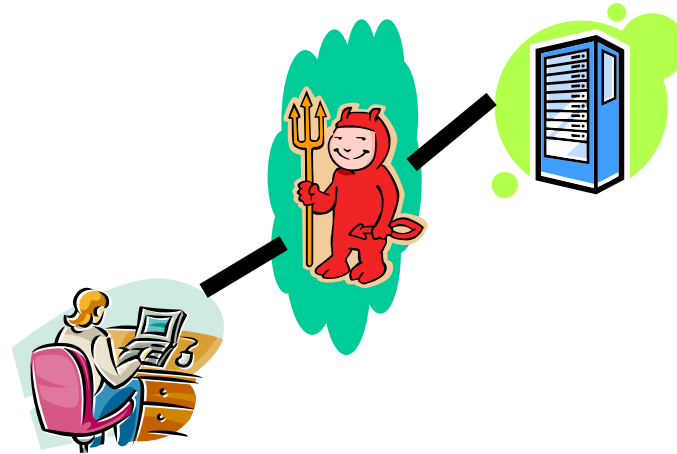
Announcements

- Project 1 due tonight, 11:59PM EST
- In class test in one week: Mon, Sept 27
- Discussion about adversary models for trusted computing systems on blackboard
 - Physical attacks on TPM
 - System Management Mode (SMM) and associated System Management Interrupts
 - Thanks to Aaron, Ashley, Peter, Jon McCune, Lujo

Outline

- Security Protocols 
- Overview of Model Checking
- Model Checking Code

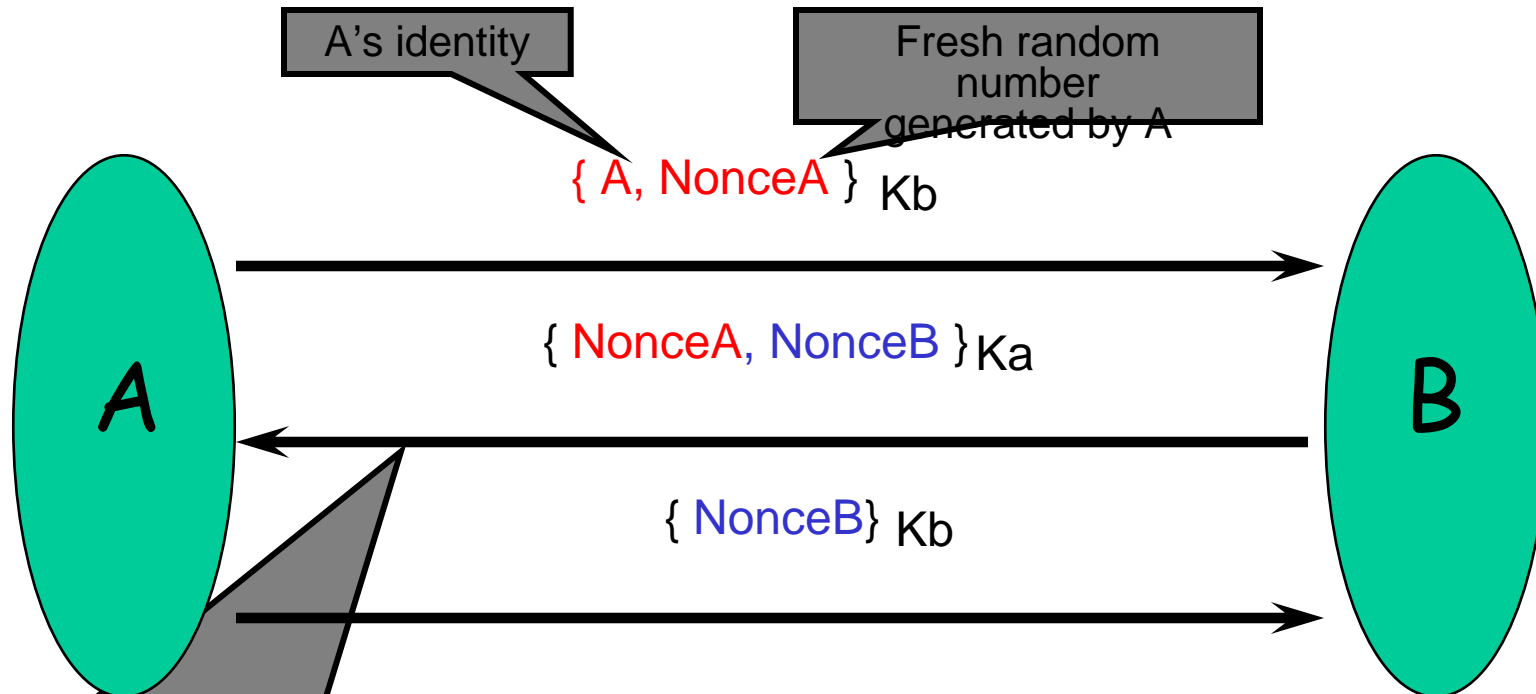
Security Protocols



- How can Alice use cryptography to send an authenticated message to Bob over an untrusted network?
- What can go wrong?
- Many examples: SSL, 802.11i WLAN, Kerberos

[Needham-Schroeder 1978]

Authentication by Encryption

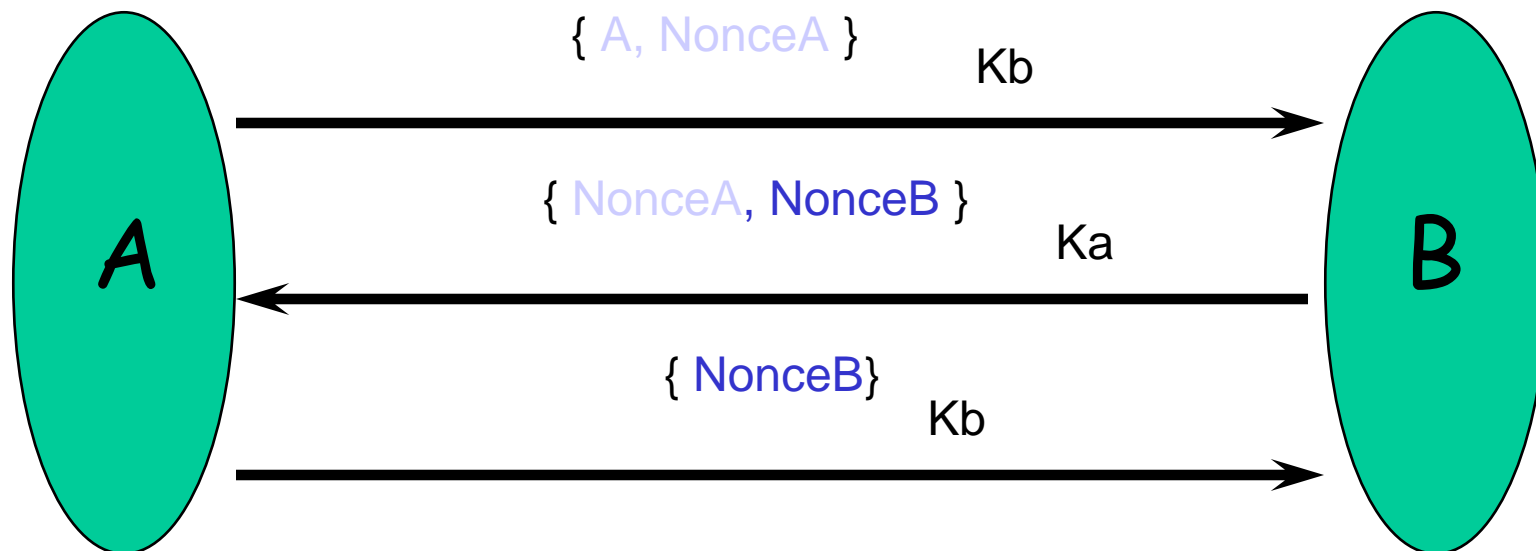


A's reasoning:

The only person who could know NonceA
is the person who decrypted 1st message
Only B can decrypt message encrypted with K_b
Therefore, B is on the other end of the line

B is authenticated!

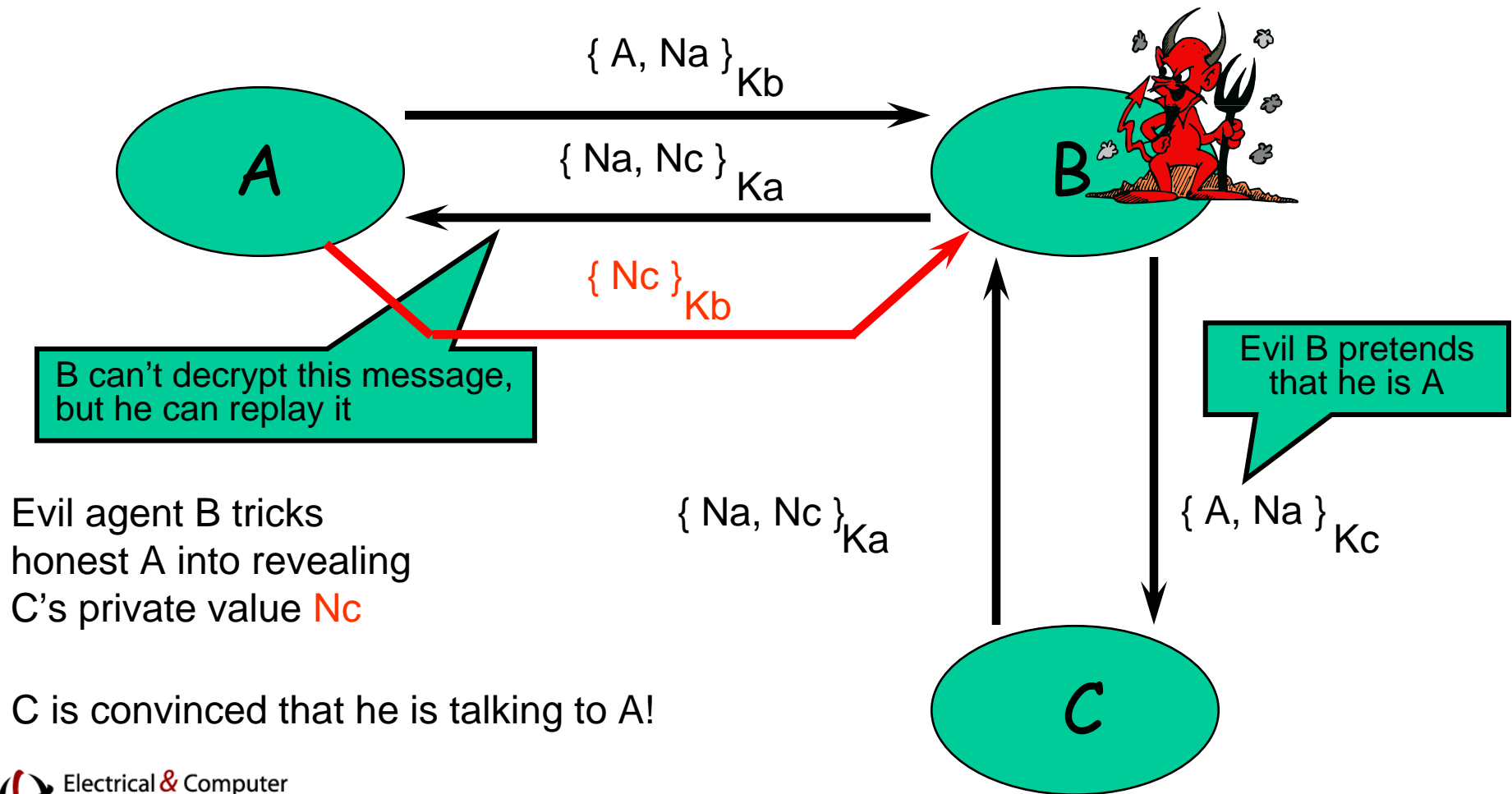
What Does This Protocol Achieve?



- Protocol aims to provide both authentication and secrecy
- After this exchange, only A and B know NonceA and NonceB

Anomaly in Needham-Schroeder

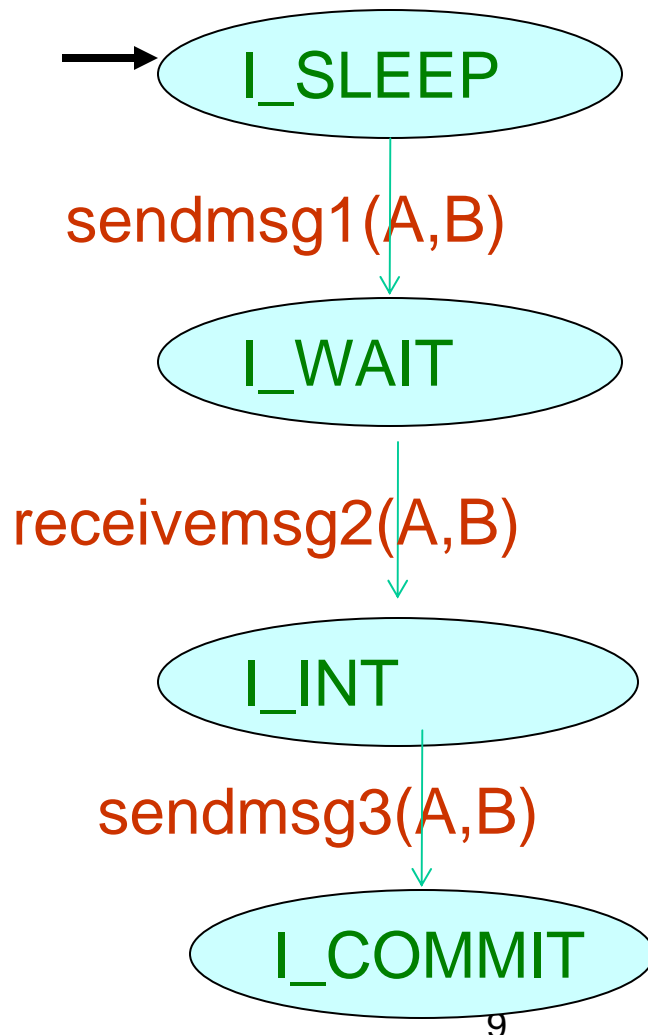
[published by Lowe 1995]



Lessons of Needham-Schroeder

- Classic man-in-the-middle attack
- Exploits participants' reasoning to fool them
- It is important to realize limitations of protocols
 - The attack requires that A willingly talk to adversary
 - In the original setting, each workstation is assumed to be well-behaved, and the protocol is correct!
- Wouldn't it be great if one could discover attacks like this automatically?
 - Enter model checking

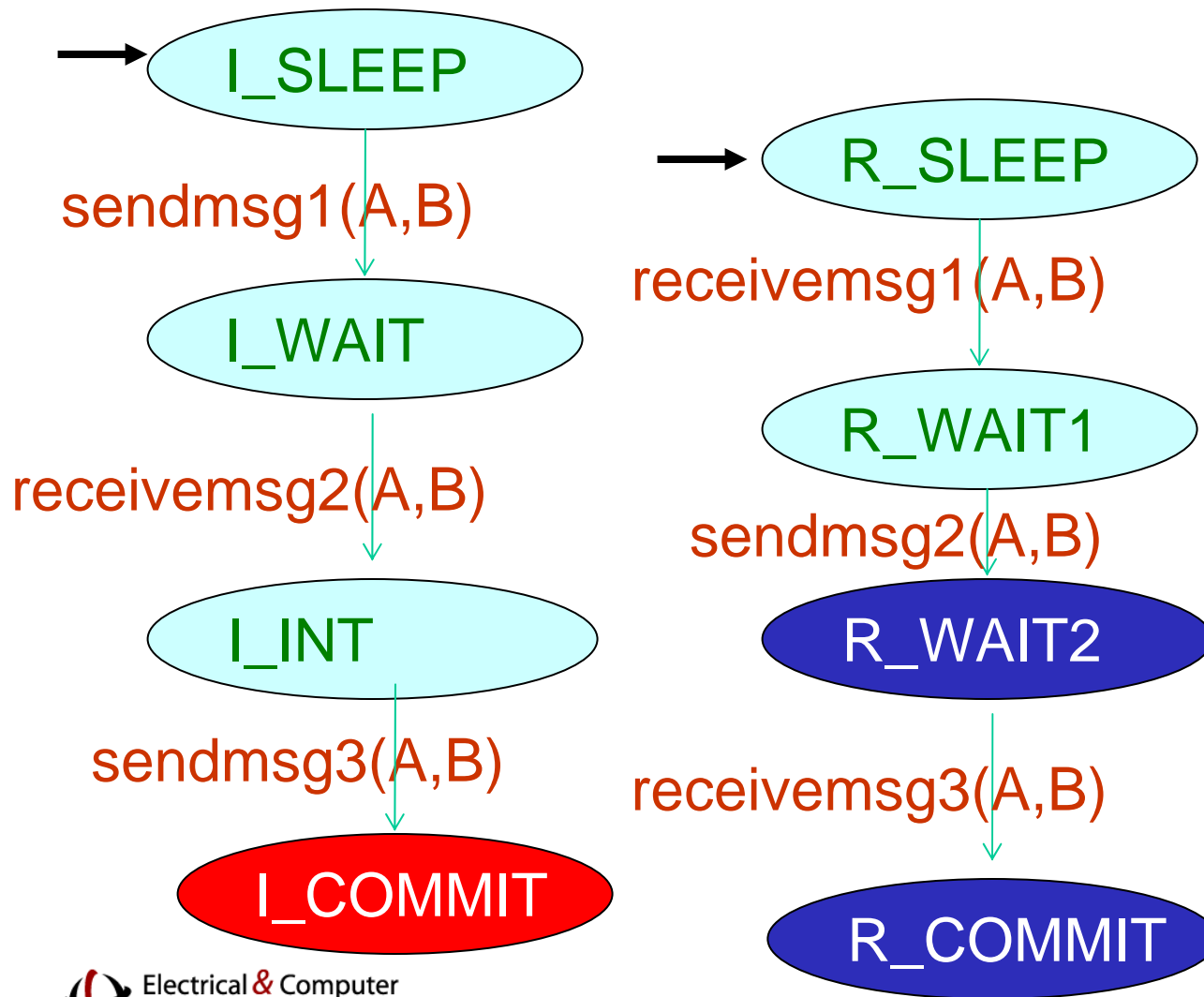
NS Initiator as a State Machine



- State machine (S, s_0, Σ, δ) where
- S : set of states
- s_0 : initial state
- Σ : set of actions
- δ : transition relation

$$\delta: S \times \Sigma \rightarrow S$$

Authentication as state machine property

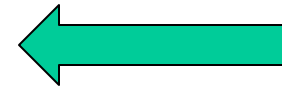


*If I_COMMIT then
R_WAIT2 or
R_COMMIT*

- Multiple concurrent sessions
- Adversary controls network (read, remove, inject messages)

Outline

- Security Protocols
- Overview of Model Checking
- Model Checking Code

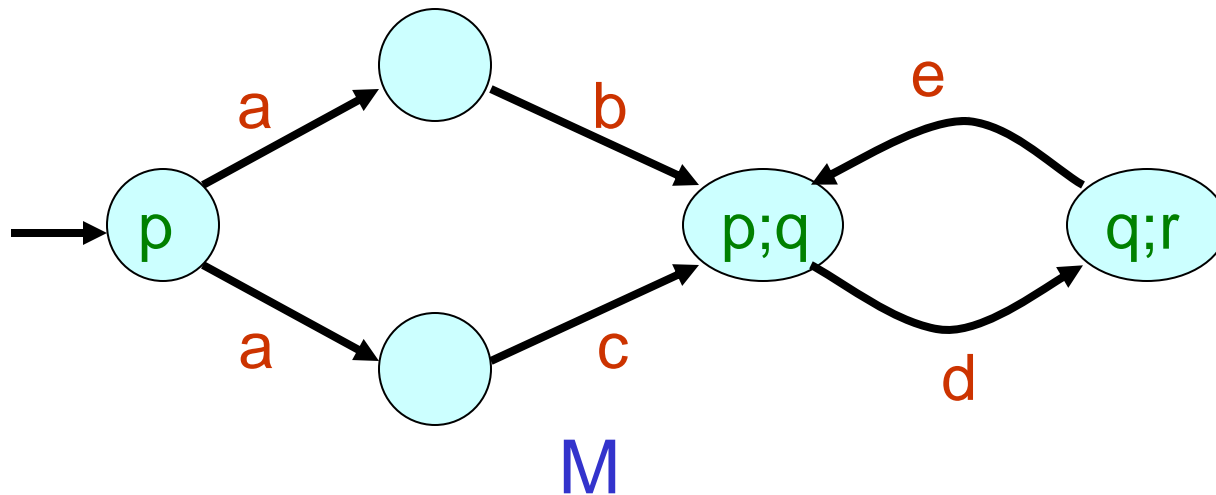


Model Checking

- **Algorithm** for checking **properties** about behaviors of **state machines**
 - Given a state machine M and a property ϕ , does M satisfy ϕ ?
- Discovered independently by **Clarke & Emerson** and **Queille & Sifakis** in the early 1980's

2007 Turing Award to Clarke, Emerson & Sifakis
<http://www.acm.org/press-room/news-releases/turing-award-07/>

Models: Doubly Labeled State Machines



$$\Sigma(M) = \{ a;b;c;d;e;f \}$$

$$AP = \{ p;q;r \}$$

alphabet

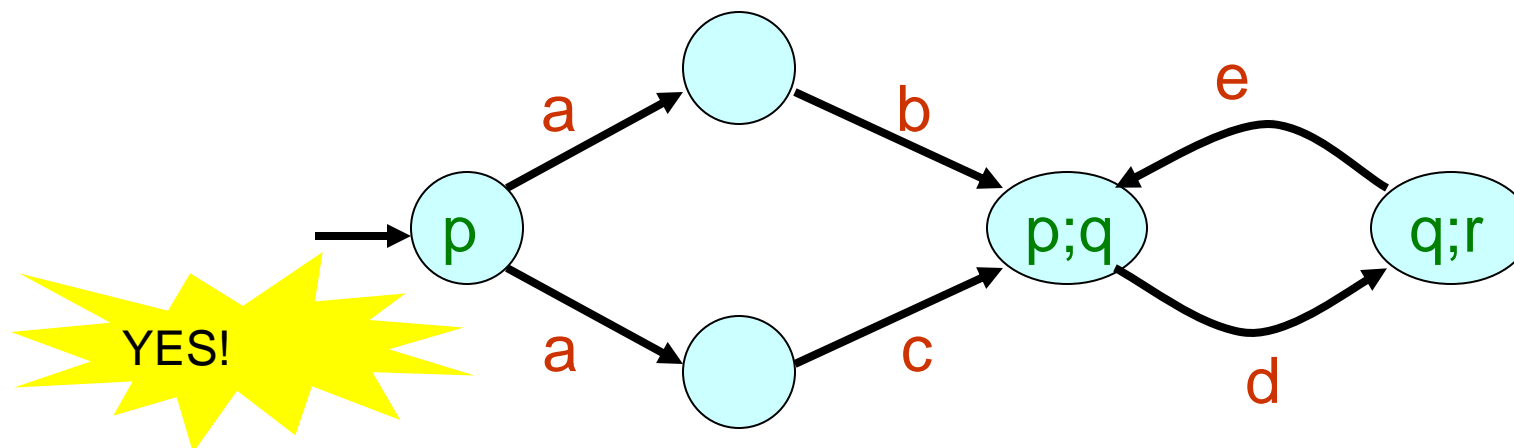
propositions

Typically, models of systems are manually created

Property 1: Safety

p and r are never true at the same time:

$$G(\neg (p \wedge r))$$



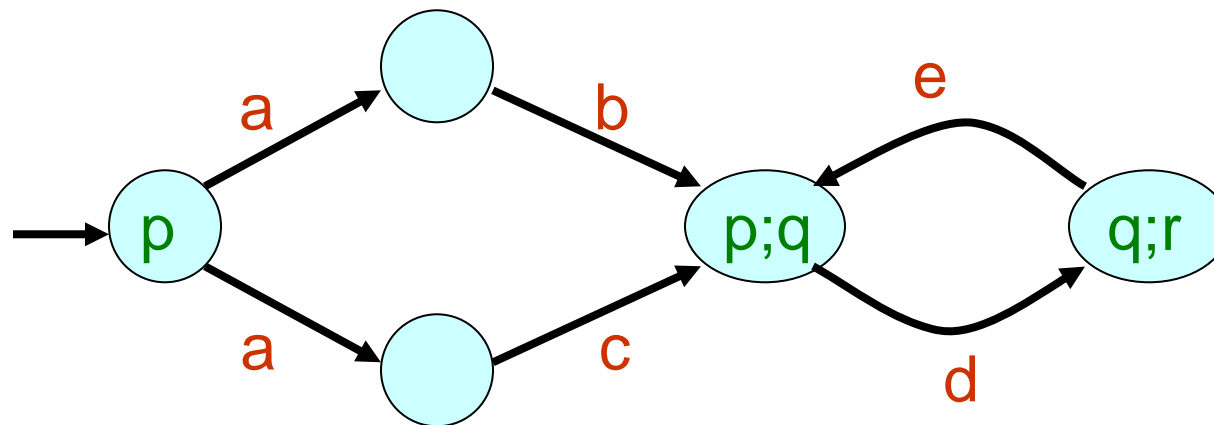
Model checking is a graph search problem

Safety: "Nothing bad happens"

More relevant for security: secrecy, authentication, order of system calls (MOPS)

Property 2: Liveness

Whenever p holds, e happens some time in the future: $G (p \Rightarrow F e)$

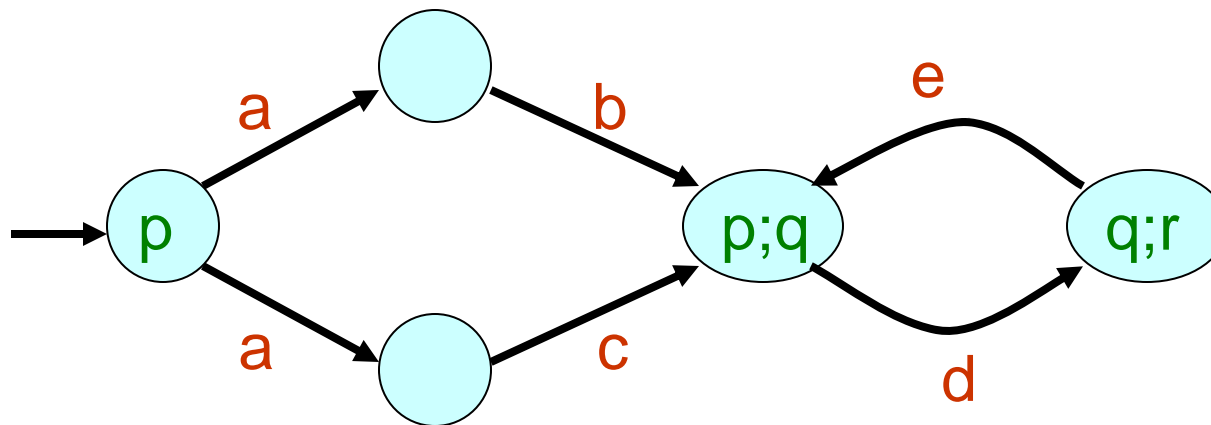


YES!

Liveness: “Something good eventually happens”

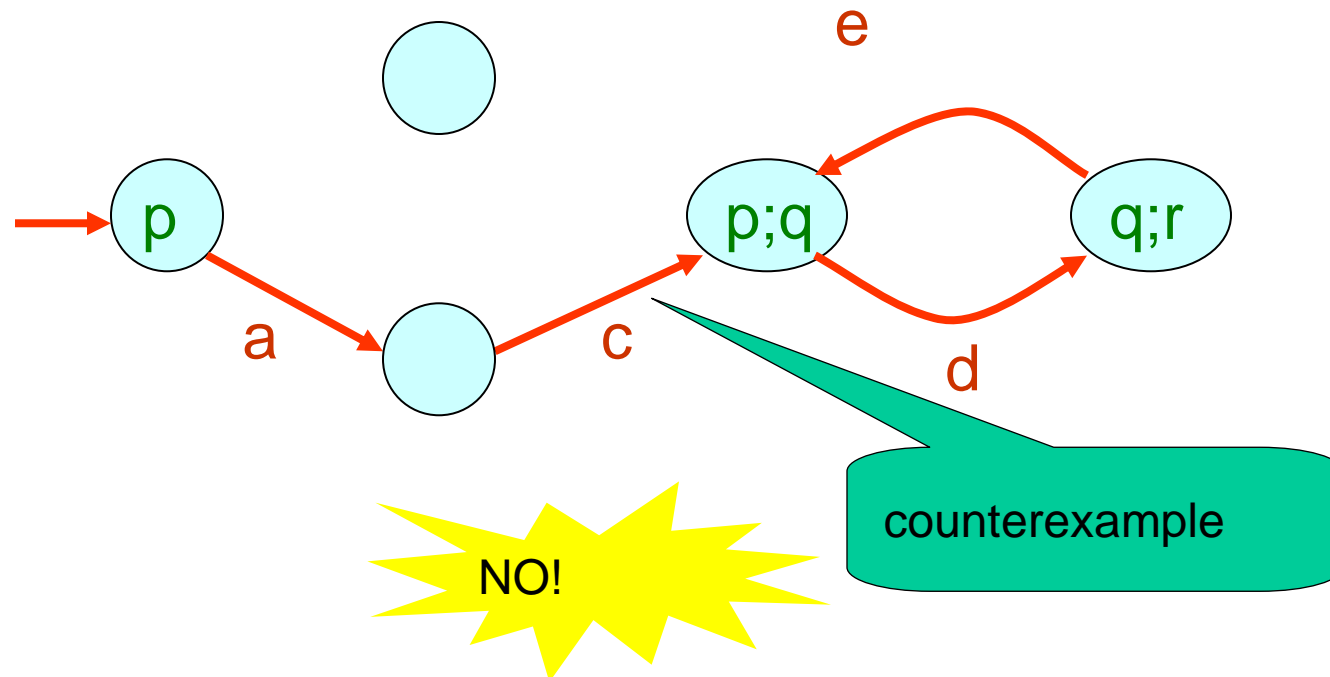
Property 3: Another Example

Whenever p holds, a happens some time in the future: $G (p \Rightarrow F a)$



Counterexample

Whenever p holds, a happens some time in the future: $G (p \Rightarrow F a)$



Counterexample useful for diagnostics

Model Checking: Pros and Cons

- Pros:

- Fully automated
- Fast (relative to similar rigorous methods)
- Counterexample useful for diagnostics

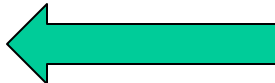
- Cons

- No correctness proof
- Applies to finite model (though some exceptions)
- State space explosion (many techniques to address)

Model Checking for Security

- Model checking security protocols
 - Ryan & Schneider, Modelling and Analysis of Security Protocols, Addison Wesley 2001
 - Mitchell et al, Automated Analysis of Cryptographic Protocols using Murphi, IEEE S & P 1997
- Model checking secure system designs
 - Lie et al, Specifying and Verifying Hardware for Tamper-Resistant Software, IEEE S & P 2003.
 - Ongoing efforts on model checking TPM spec (e.g. at MITRE), security hypervisors (CMU)

Outline

- Security Protocols
- Overview of Model Checking
- Model Checking Code 

Verifying Security of OpenSSL

Client Code

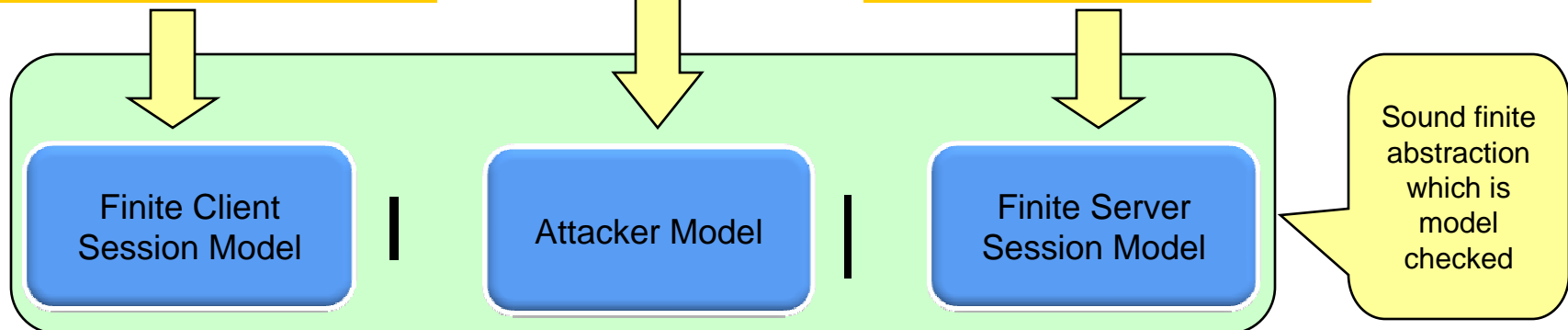
```
int state = 0;
while(1) {
  if(state==0) {
    send_hello();
    state++;
  } else if(state==1) {
    ver = rcv_hello();
    state++;
  } else ...
}
```

Server Code

```
int state = 0;
while(1) {
  if(state==0) {
    rcv_hello();
    state++;
  } else if(state==1) {
    send_hello(ver);
    state++;
  } else ...
}
```



Infinite state system

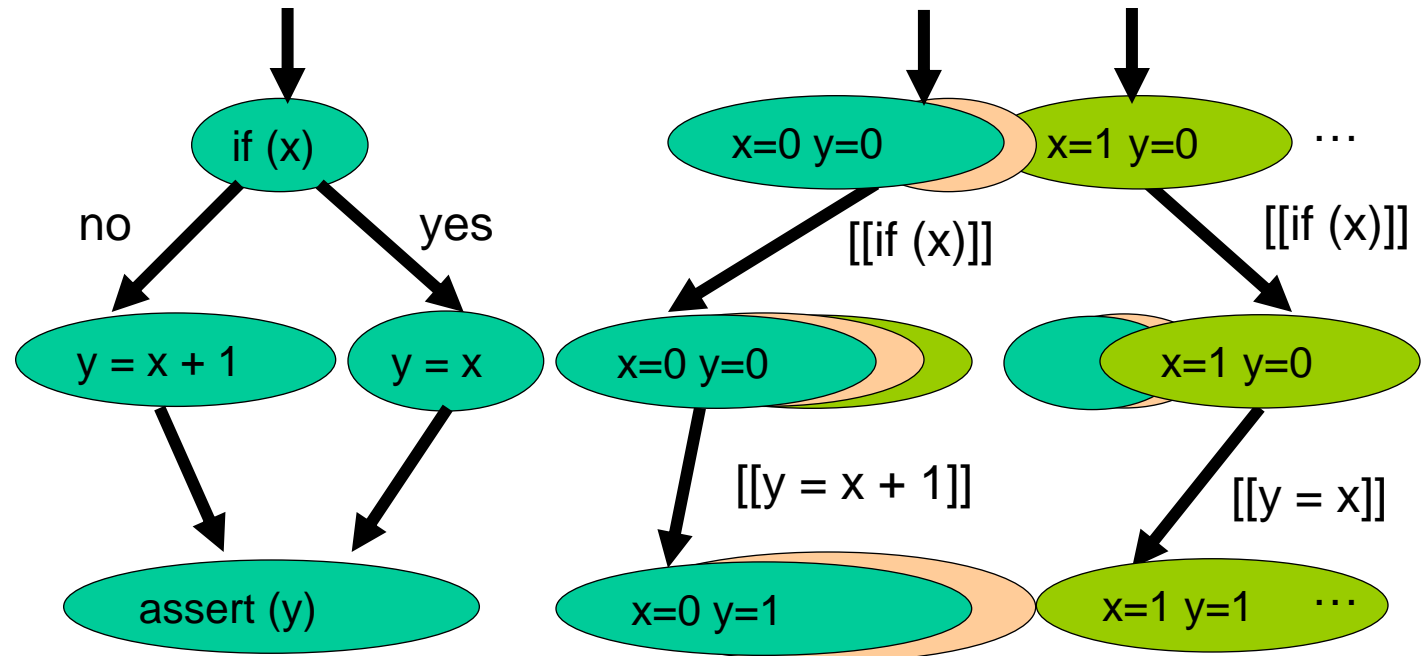


Models of C Code

```

if (x){
y = x;}
else {
y = x + 1;}
assert (y);

```



Program: Syntax

Control Flow Graph

Model: States and transitions

Goal: Extract *finite* state model *automatically* from C code

Infinite State

Questions?