IP Security and Key Establishment

Plan for the Next Few Lectures

- Today: "systems" lecture on IP Security and design of key exchange protocols for IPSec
  - Defending against denial of service
  - "Real-world" considerations for protocol design
  - No formal methods (yet)
    - But see Cathy Meadows' paper on the website
- Monday: no class (Labor Day)
- Next Wednesday: process algebras
  - Homework assigned (using Murφ)
- Then bring all together – use process algebra and rational reconstruction to understand JFK protocol

IP Security Issues

- Eavesdropping
- Modification of packets in transit
- Identity spoofing (forged source IP addresses)
- Denial of service

- Many solutions are application-specific
  - TLS for Web, S/MIME for email, SSH for remote login
- IPSec aims to provide a framework of open standards for secure communications over IP
  - Protect every protocol running on top of IPv4 and IPv6

IPSec: Network Layer Security

IPSec = AH + ESP + IPcomp + IKE

AH + ESP: Protection for IP traffic
- AH provides integrity and origin authentication
- ESP also confidentiality

IPcomp: Compression

IKE: Sets up keys and algorithms for AH and ESP

- AH and ESP rely on existing security association
  - Roughly, peers must share a set of secret keys and agree on each other's IP addresses and crypto schemes
- Internet Key Exchange (IKE)
  - Goal: establish security association for AH and ESP
  - If IKE is broken, AH and ESP provide no protection!

Transport Mode vs. Tunnel Mode

- Transport mode secures packet payload and leaves IP header unchanged
  - Typically, client-gateway (e.g., PC to remote host)

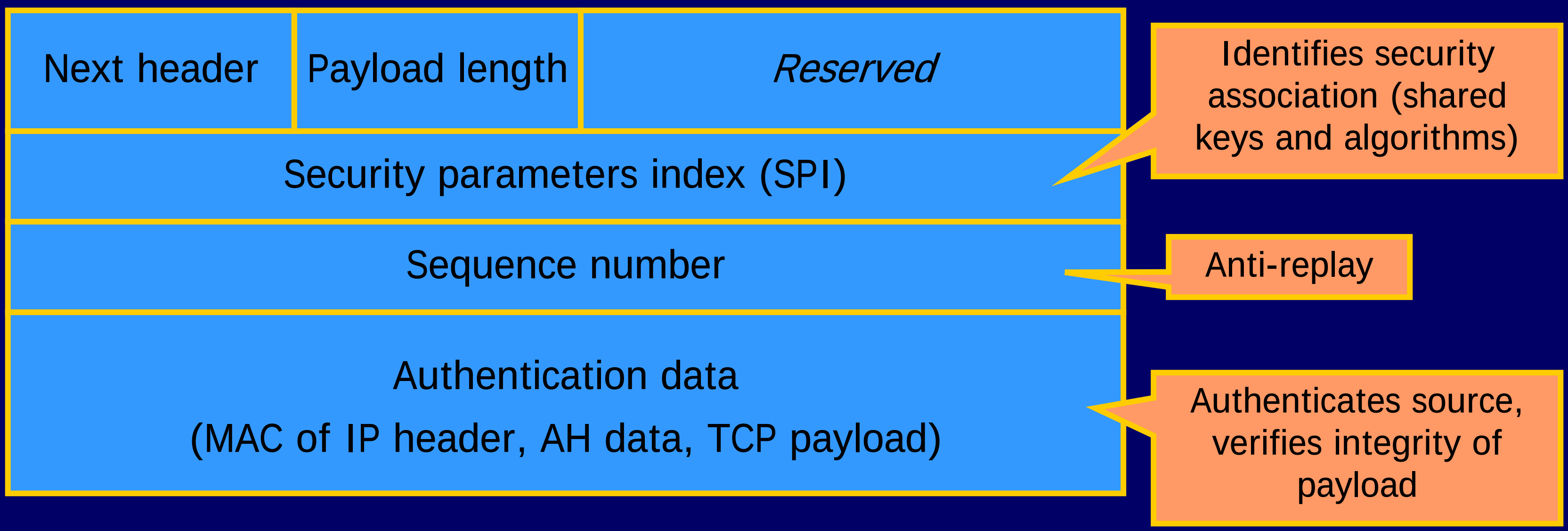
IP header (end-to-end)	IPSec header	TCP/UDP header + data

- Tunnel mode encapsulates both IP header and payload into IPSec packets
  - Typically, gateway-gateway (e.g., router to firewall)

IP header (end-to-end)	IPSec header	IP header (tunnel)	TCP/UDP header + data

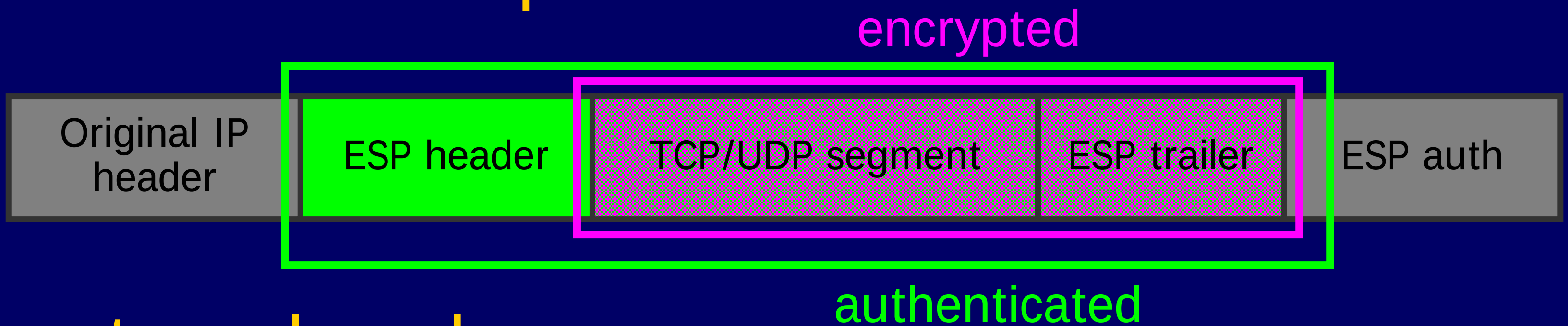
AH: Authentication Header

- Provides integrity and origin authentication
- Authenticates portions of the IP header
- Anti-replay service (to counter denial of service)
- No confidentiality

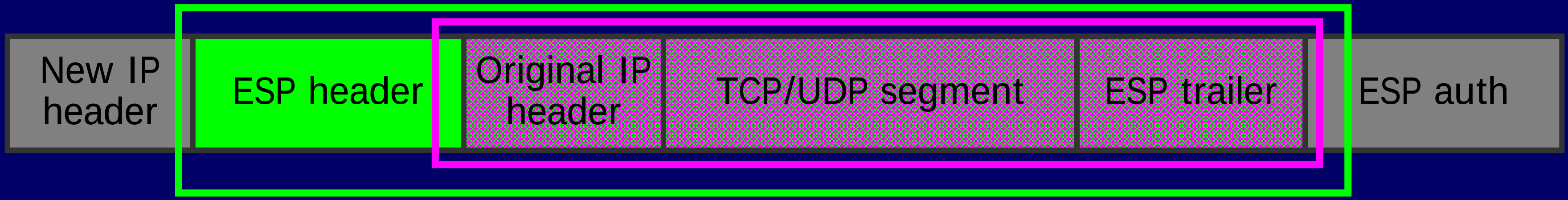


ESP: Encapsulated Secure Payload

- Confidentiality and integrity for packet payload
  - Symmetric cipher negotiated as part of security assoc
- Optionally provides authentication (similar to AH)
- Can work in transport...



- ...or tunnel mode

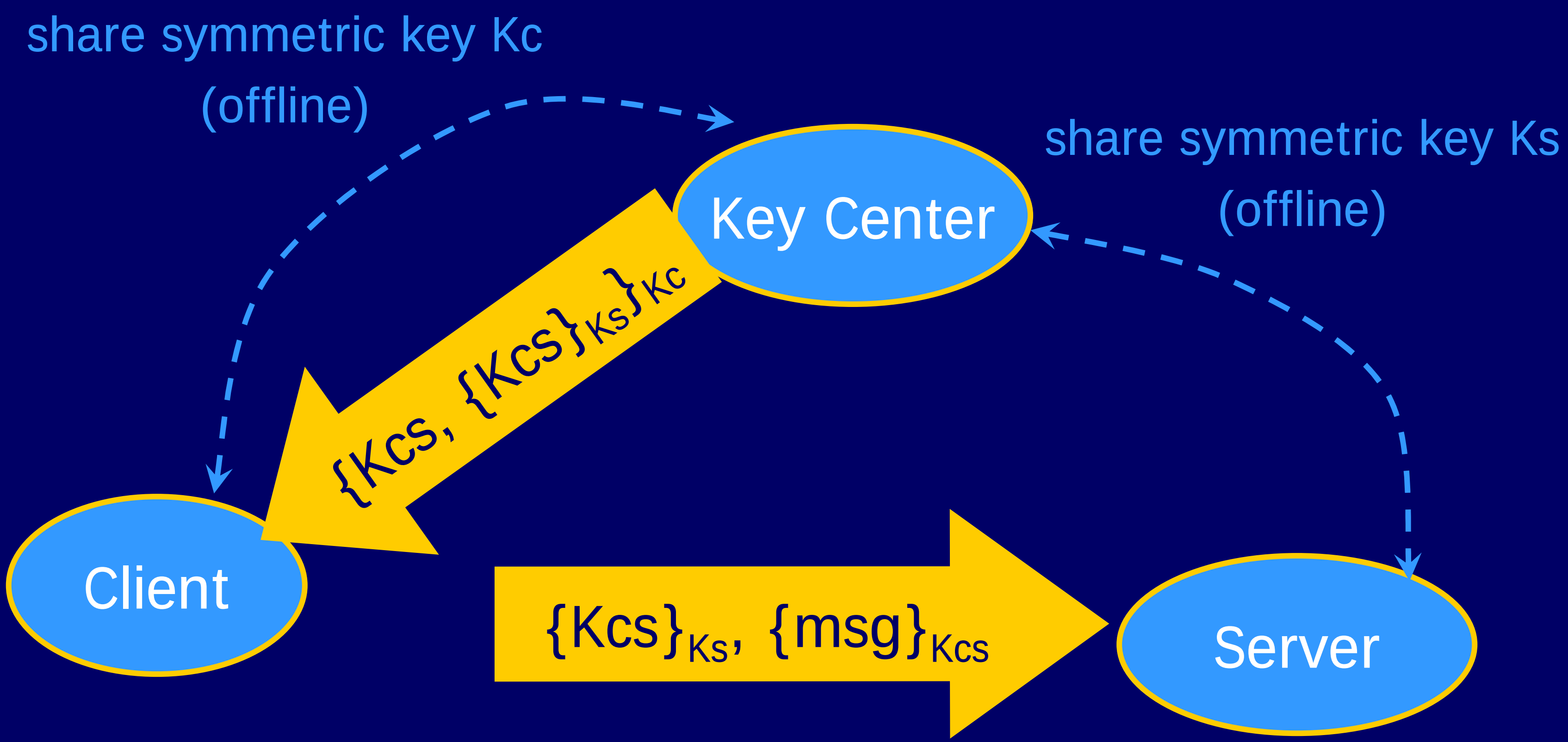


Key Management

Cryptography reduces many problems to key management

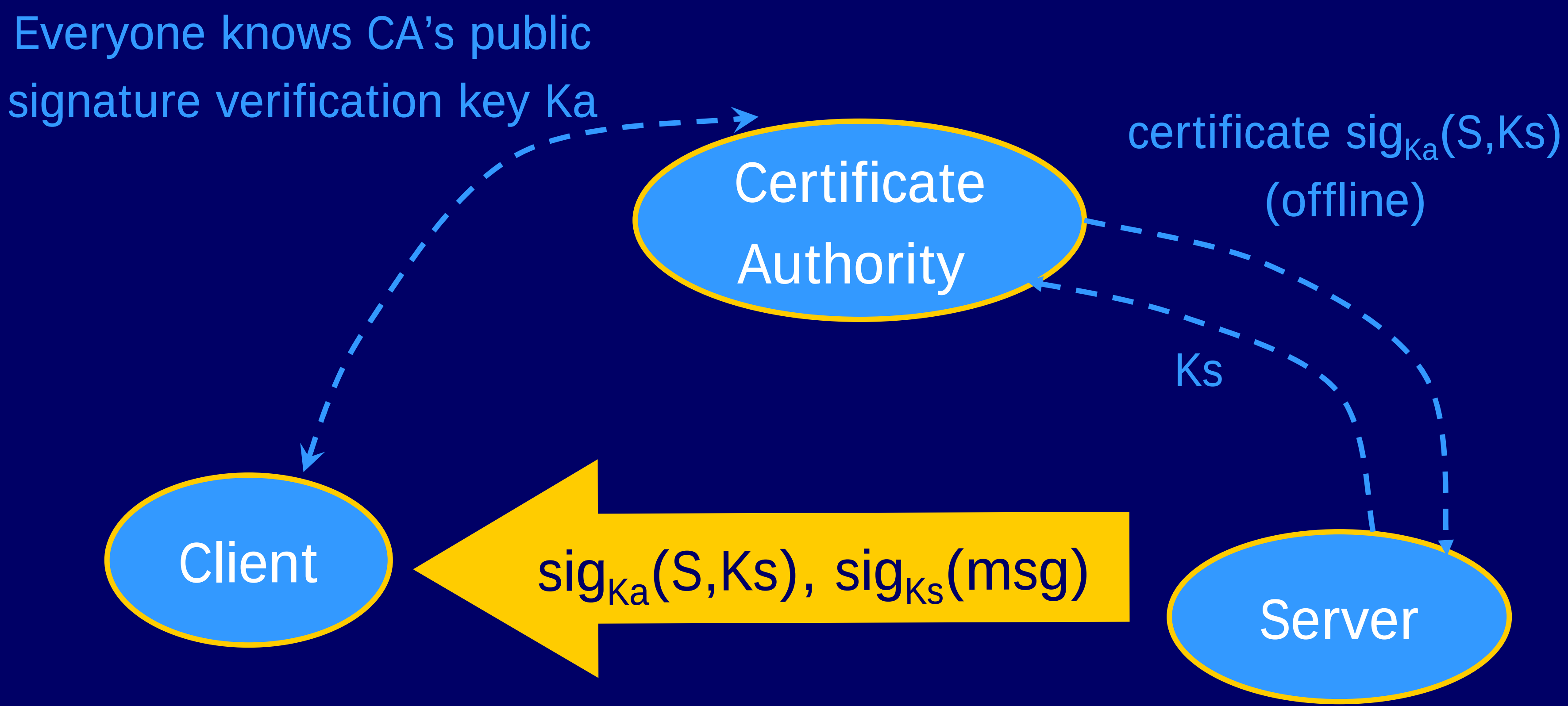
- Out of band
  - Can set up some keys this way (Kerberos)
- Public-key infrastructure (PKI)
  - Leverage small number of public signing keys by using certificate chains
- Protocols for establishing short-lived session keys
  - Avoid extended use of permanent secrets
  - Forward secrecy
    - Compromise of one session key does not help the attacker to compromise subsequent session keys

Key Distribution in Kerberos



Key Center generates session key Kcs and distributes it using shared long-term keys

Public-Key Infrastructure (PKI)



Server certificate can be verified by any client that has CA's public key Ka

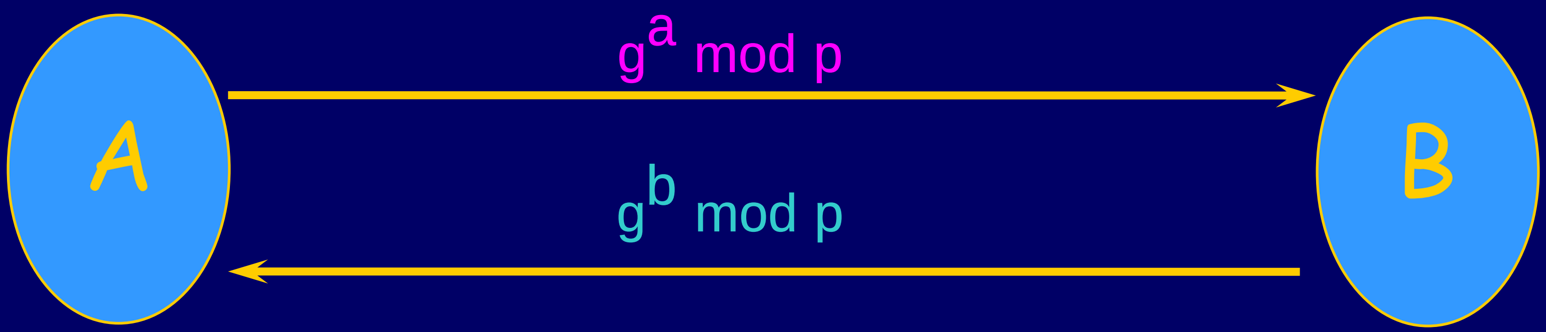
Certificate authority is "offline"

Properties of Key Exchange Protocols

- Goal: generate and agree on session key using some shared initial information
- What other properties are needed?
  - Authentication (know identity of other party)
  - Secrecy (generated key not known to any others)
  - Prevent replay of old key material
  - Forward secrecy
  - Prevent denial of service
  - Protect identities (avoid disclosure to others)
  - Other properties you can think of???

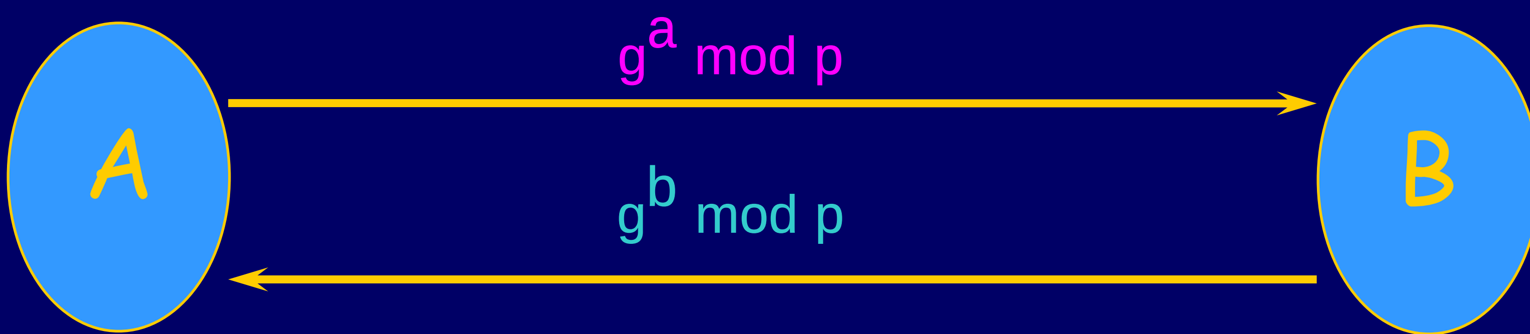
Diffie-Hellman Key Exchange

- Assume finite group $G = \langle S, \bullet \rangle$
  - Choose generator g so every $x \in S$ is $x = g^n$ for some n
  - Example: integers modulo prime p
- Protocol



Alice, Bob share g^{ab} mod p not known to anyone else

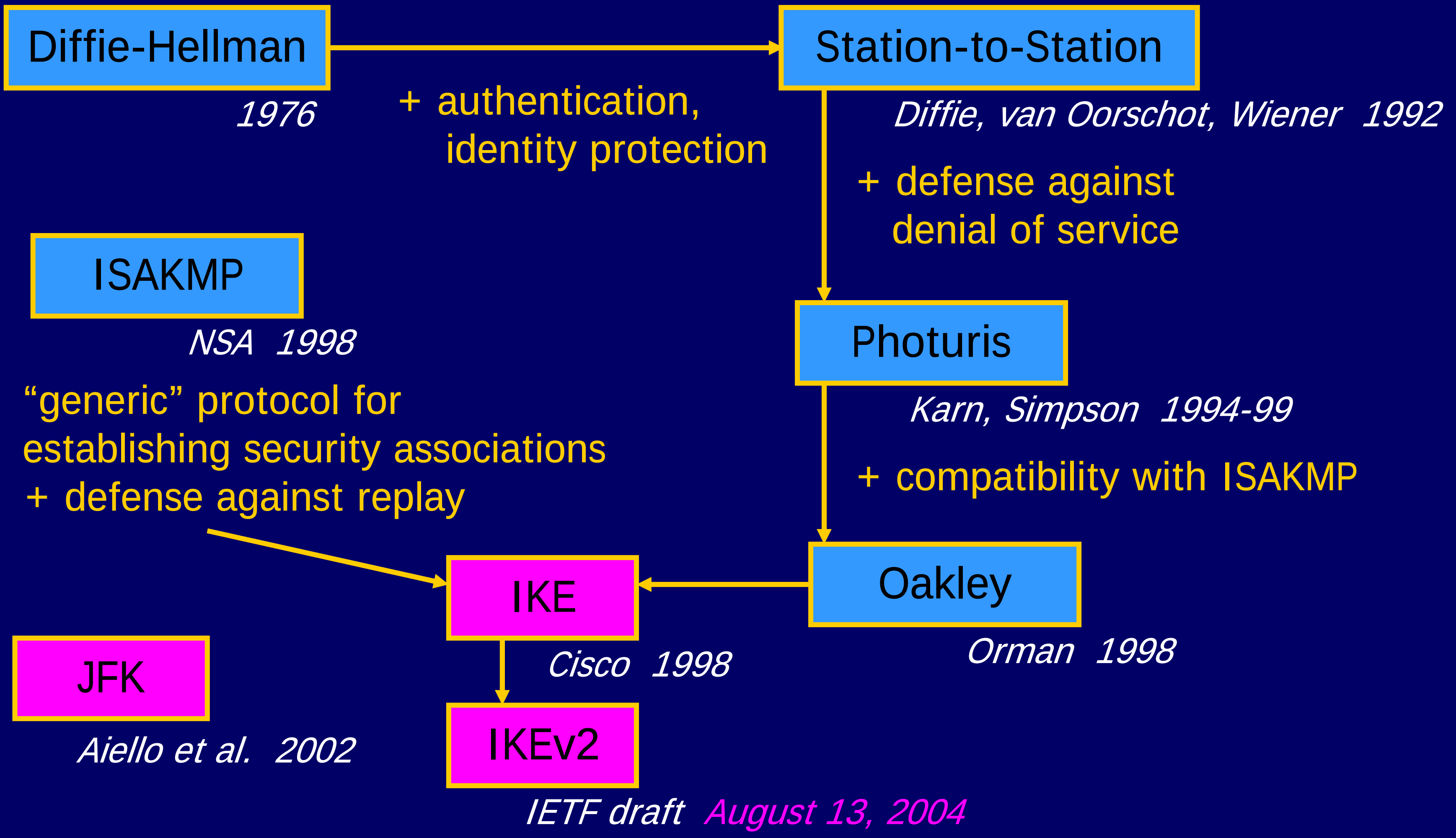
Diffie-Hellman Key Exchange



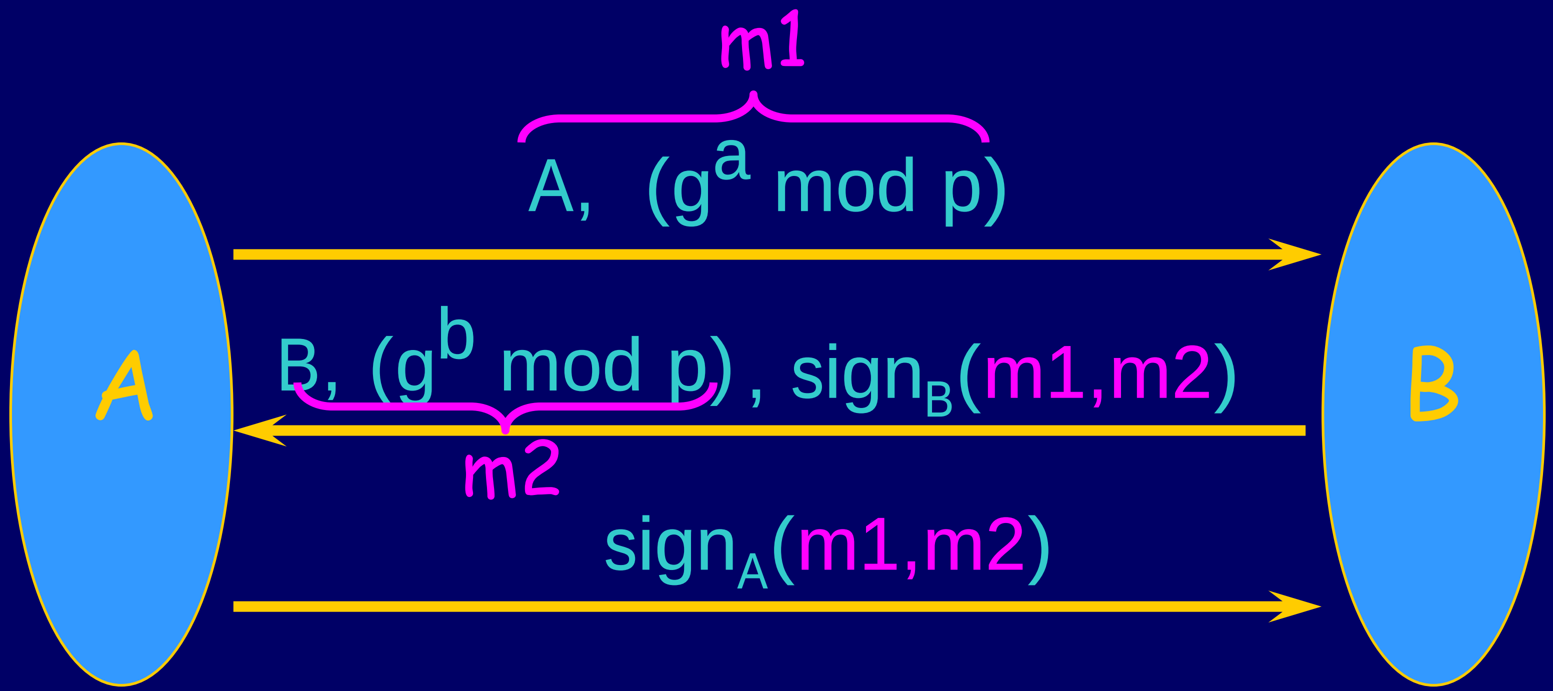
Authentication?	No
Secrecy?	Only against <u>passive</u> attacker
Replay attack?	Vulnerable
Forward secrecy?	Yes
Denial of service?	Vulnerable
Identity protection?	Yes

Participants can't tell g^x mod p from a random number: send them garbage and they'll do expensive exponentiations

IKE Genealogy



Basic Idea

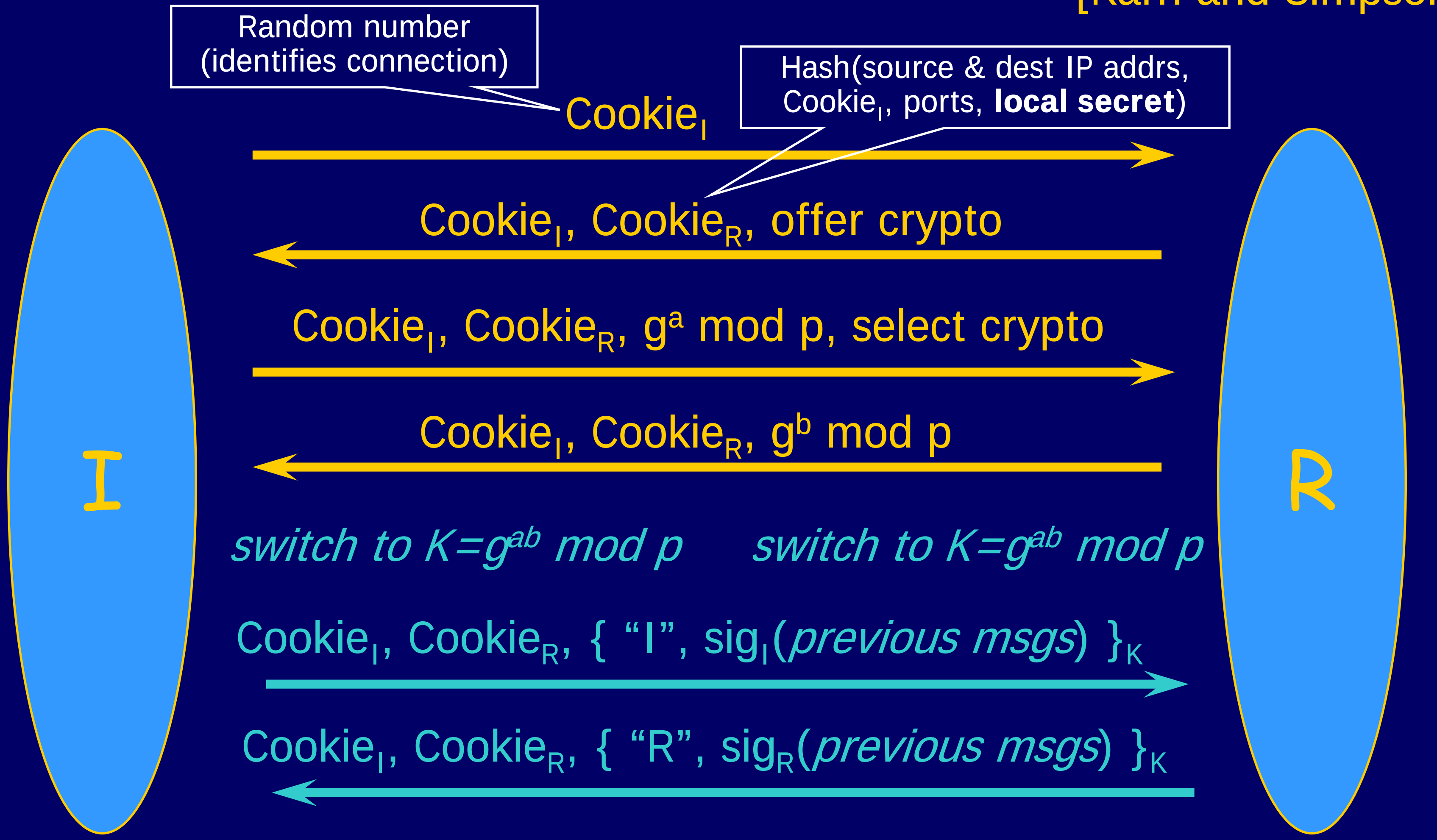


Result: A and B share session key g^{ab} mod p

Signatures provide authentication,
as long as signature verification keys are known

(Simplified) Photuris

[Karn and Simpson]



Preventing Denial of Service

- Resource-clogging attacks are a serious issue
  - If responder opens a state for each connection attempt, attacker can initiate thousands of connections from bogus or forged IP addresses
- Cookies ensure that the responder is stateless until initiator produced at least 2 messages
  - Responder's state (IP addresses and ports of the connection) is stored in a cookie and sent to initiator
  - After initiator responds, cookie is regenerated and compared with the cookie returned by the initiator
  - The cost is 2 extra messages in each execution!

Cookies in Photuris and ISAKMP

- Photuris cookies are derived from local secret, IP addresses and ports, counter, crypto schemes
  - Same (frequently updated) secret for all connections
- ISAKMP requires unique cookie for each connect
  - Add timestamp to each cookie for uniqueness
  - Now responder needs to keep state (“cookie crumb”)
    - Vulnerable to DoS (see Simpson’s rant on the course website)
- Inherent conflict: to prevent replay, need to keep state (remember values that you’ve seen before), but keeping state allows denial of service
  - JFK design gets it right (we’ll talk about JFK later)

IKE Overview

- Goal: create security association between 2 hosts
  - Shared encryption and authentication keys, agreement on crypto algorithms (a-la carte, not like SSL suites)
- Two phases: 1^{st} phase establishes security association (IKE-SA) for the 2^{nd} phase
  - Always by authenticated Diffie-Hellman (expensive)
- 2^{nd} phase uses IKE-SA to create actual security association (child-SA) to be used by AH and ESP
  - Use keys derived in the 1^{st} phase to avoid DH exchange
  - Can be executed cheaply in “quick” mode

Why Two-Phase Design?

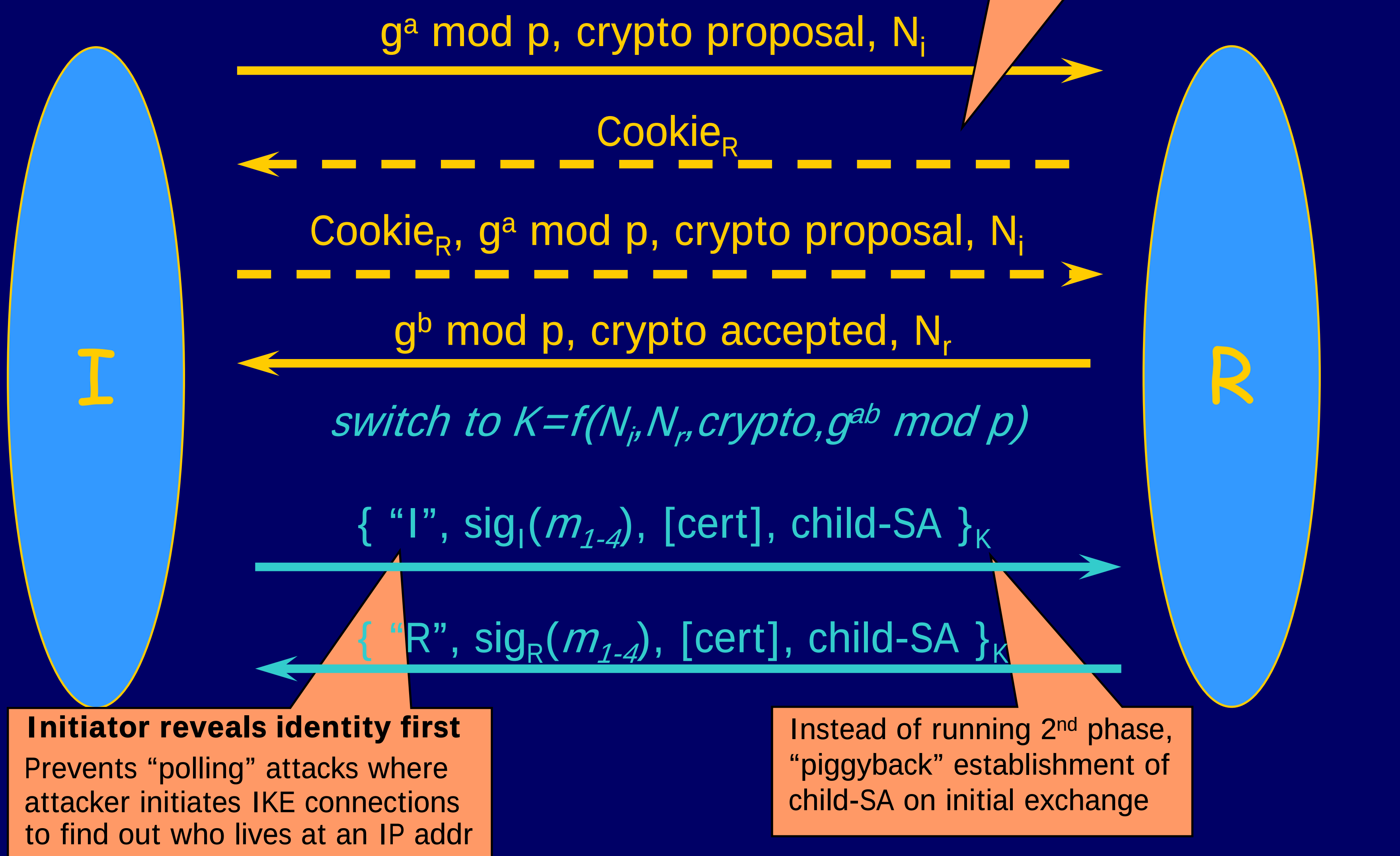
- Expensive 1st phase creates “main” SA
- Cheap 2nd phase allows to create multiple child SAs (based on “main” SA) between same 2 hosts
  - Avoid multiplexing several conversations over same SA
    - For example, if encryption is used without integrity protection (bad idea!), it may be possible to splice the conversations
  - Different conversations may need different protection
    - Some traffic only needs integrity protection or short-key crypto
    - Too expensive to always use strongest available protection
  - Different SAs for different classes of service
- JFK is a single-phase protocol (talk about it later)

IKEv1 Was a Mess

- Two modes for 1st phase: "main" and "aggressive"
  - Fewer messages in "aggressive" mode, but no identity protection and no defense against denial of service
  - Main mode vulnerable to DoS due to bad cookie design
  - Many field sizes not verified; poor error handling
- Four authentication options for each mode
  - Shared keys; signatures; public keys in 2 different ways
- Special "group" mode for group key establishment
- Grand total of 13 different variants
  - Difficult to implement, impossible to analyze
  - Security problems stem directly from complexity

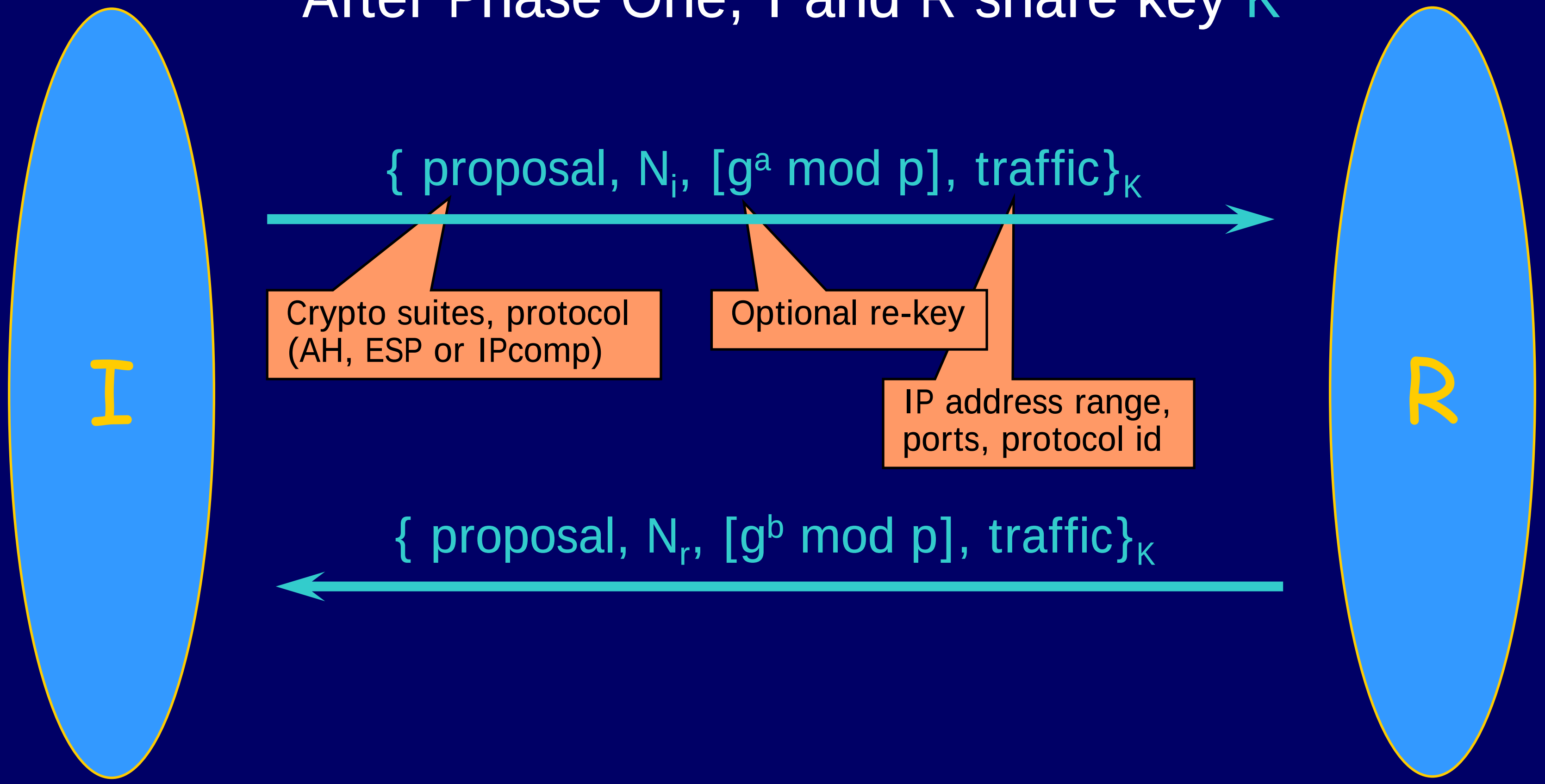
IKEv2: Phase One

Optional: refuse 1st message and demand return of stateless cookie



IKEv2: Phase Two (Create Child-SA)

After Phase One, I and R share key K



Can run this several times to create multiple SAs

Other Aspects of IKE

We did not talk about...

- Interaction with other network protocols
  - How to run IPSec through NAT (Network Address Translation) gateways?
- Error handling
  - Very important! Bleichenbacher attacked SSL by cryptanalyzing error messages from an SSL server
- Protocol management
  - Dead peer detection, rekeying, etc.
- Legacy authentication
  - What if one of the parties does not have a public key?