

Key Exchange Protocols

J. Mitchell

Next few lectures

- ◆ Today 1/17
 - Some possible projects
 - Key exchange protocols and properties
- ◆ Tuesday 1/19
 - Contract-signing protocols
 - Choose project partner, topic?
- ◆ Next Thurs 1/24
 - Wireless security: 802.11i
 - Homework #1 due
- ◆ Tuesday 1/29
 - Probabilistic model checking
- ◆ Project presentation #1 1/31
 - One page, 1-2 slides on your project topic

Course projects

Choose a subject

◆ Network protocol

- Wired networking (VoIP?)
- Wireless
- Mobility

◆ Security system

- Tamper-proof chip
- "Trusted computing"

◆ Privacy, security policy

- HIPAA
- GLBA
- ...

Choose a tool

◆ Murphi

- Standard finite-state tool

◆ PRISM

- probabilistic checker

◆ MOCHA

- Games, temporal logic

◆ Constraint solvers

◆ Avispa tool set

◆ Prolog

◆ Isabelle

- Automated theorem proving

Some project topics

- ◆ Add timestamps to Kerberos
- ◆ Group key handshake from 802.11i
- ◆ IPv6 binding update (already taken?)
- ◆ 802.1af, DKIM, ...
- ◆ TCG protocols register TPM to CA, ...
- ◆ HIPAA, other privacy laws, GLBA, ...
- ◆ Voting machine, voting procedure
 - election board “programs” election, votes cast, counted, possibly recounted ...

Send email or talk with us about choosing a project

Process

- ◆ Choose your project partner
 - You can work alone, or with another person
- ◆ Chose a subject for your project
 - Can be something already familiar
 - Project presentation #1 at this point!
- ◆ Formulate an "abstraction" of system
 - Separate relevant and irrelevant details
 - Identify security properties of interest
- ◆ Choose a tool and method
- ◆ Complete your study. Success!
 - Clear explanation of security goals
 - Possible bugs or insecure configurations
 - Identify proper, improper use

Key Management

◆ Out of band

- Can set up some keys this way (Kerberos)

◆ Public-key infrastructure (PKI)

- Leverage small # of public signing keys

◆ Protocols for session keys

- Generate short-lived session key
- Avoid extended use of important secret
- Don't use same key for encryption and signing
- Forward secrecy

Cryptography reduces many problems to key management

Public-Key Infrastructure

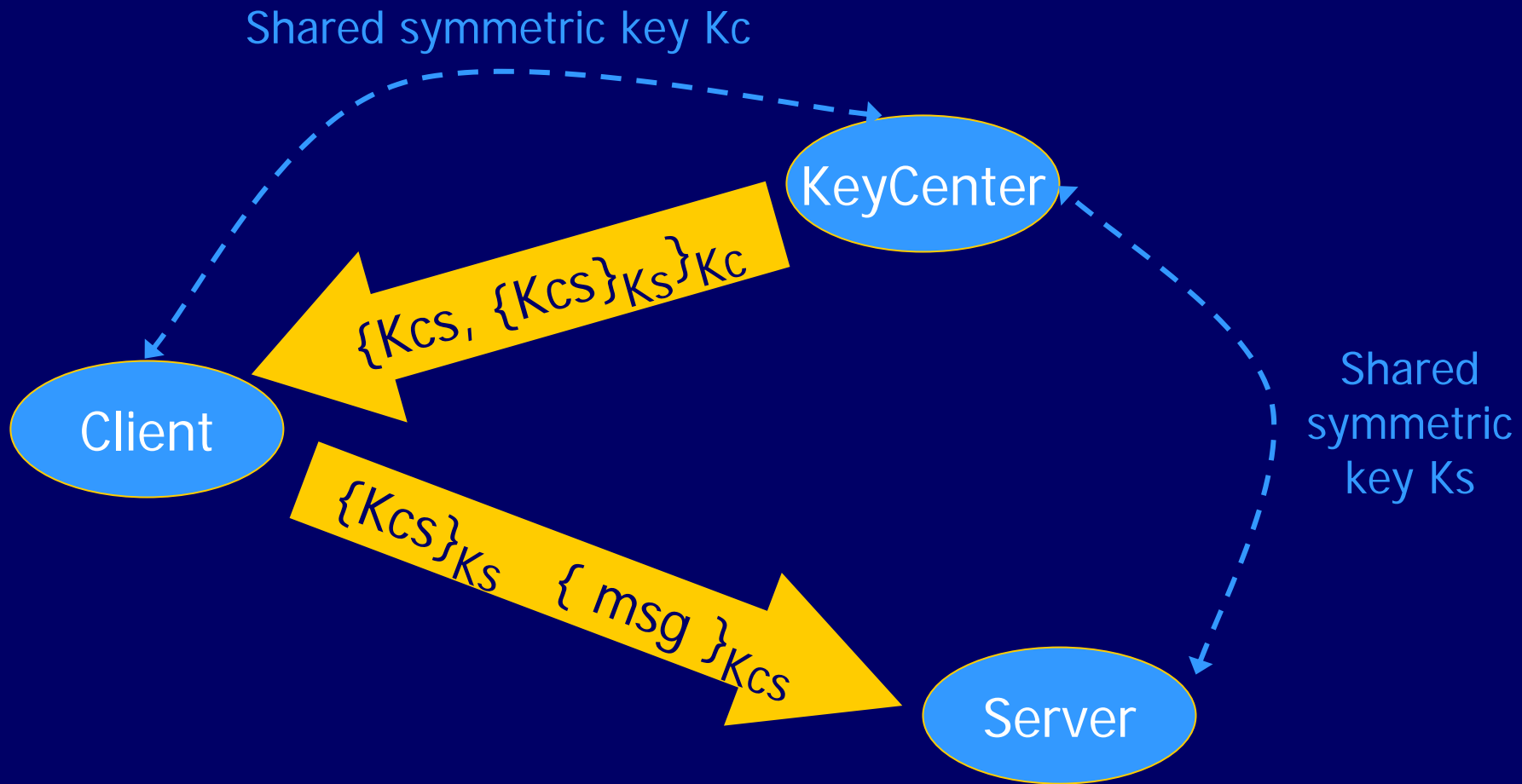
Known public signature verification key K_a



Server certificate can be verified
by any client that has CA key K_a

Certificate authority is "off line"

Key Distribution: Kerberos Idea



Key Center generates session key K_{cs} and distributes using shared long-term keys

Key Exchange

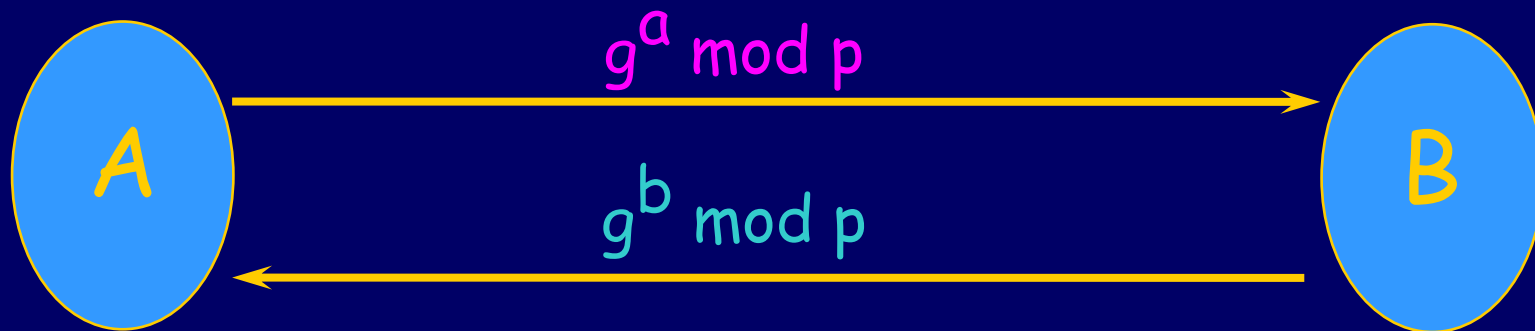
- ◆ Parties may have initial information
- ◆ Generate and agree on session key
 - Authentication - know ID of other party
 - Secrecy - key not known to any others
 - Avoid replay attack
 - Forward secrecy
 - Avoid denial of service
 - Identity protection - disclosure to others
 - Other properties you can think of???

Diffie-Hellman Key Exchange

◆ Assume finite group $G = \langle S, \cdot \rangle$

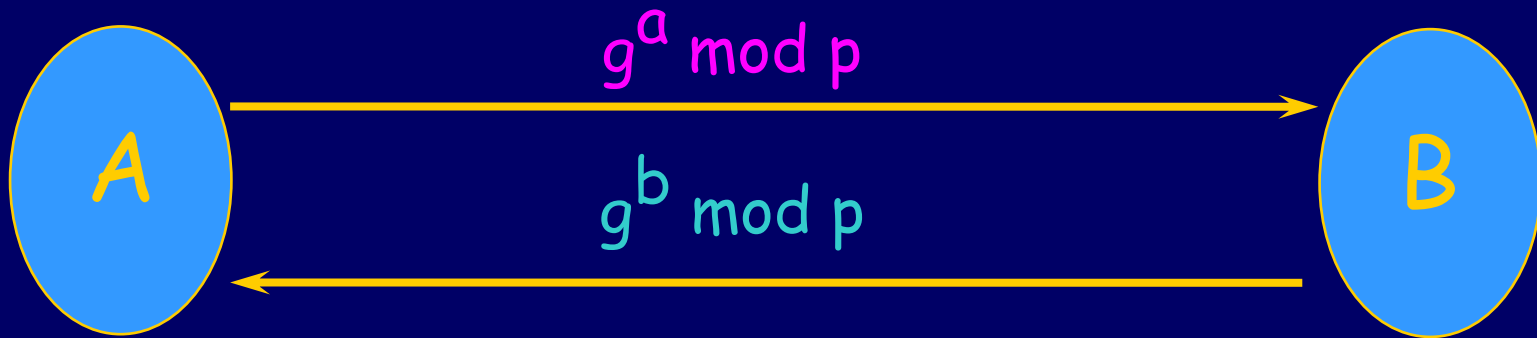
- Generator g so every $x \in S$ is $x = g^n$
- Example: integers modulo prime p

◆ Protocol



Alice, Bob share $g^{ab} \bmod p$ not known to anyone else

Diffie-Hellman Key Exchange



Authentication?

Secrecy?

Replay attack

Forward secrecy?

Denial of service?

Identity protection?

IPSec: Network Layer Security

◆ Authentication Header (AH)

- Access control and authenticate data origins
- replay protection
- No confidentiality

◆ Encapsulated Secure Payload (ESP)

- Encryption and/or authentication

➔ Internet Key Management (IKE)

- Determine and distribute secret keys
- Oakley + ISAKMP
- Algorithm independent

◆ Security policy database (SPD)

- discarded, or bypass

IKE: Many modes

◆ Main mode

- Authentication by pre-shared keys
- Auth with digital signatures
- Auth with public-key encryption
- Auth with revised public-key encryption

◆ Quick mode

- Compress number of messages
- Also four authentication options

Aug 2001 Position Statement

- ◆ In the several years since the standardization of the IPSEC protocols (ESP, AH, and ISAKMP/IKE), ... several security problems..., most notably IKE.
- ◆ Formal and semi-formal analyses by Meadows, Schneier et al, and Simpson, have shown ... security problems in IKE stem directly from its complexity.
- ◆ It seems ... only a matter of time before serious *implementation* problems become apparent, again due to the complex nature of the protocol, and the complex implementation that must surely follow.
- ◆ The Security Area Directors have asked the IPSEC working group to come up with a replacement for IKE.

How to study complex protocol

General Problem in Security

- ◆ Divide-and-conquer is fundamental
 - Decompose system requirements into parts
 - Develop independent software modules
 - Combine modules to produce required system

- ◆ Common belief:
 - Security properties do not compose

Difficult system development problem

Example protocol

Protocol P1

$A \rightarrow B : \{\text{message}\}_{K_B}$

$A \rightarrow B : K_{A^{-1}}$

◆ This satisfies basic requirements

- Message is transmitted under encryption
- Revealing secret key $K_{A^{-1}}$ does not reveal message

Similar protocol

Protocol P2

$B \rightarrow A : \{\text{message}'\}_{K_A}$

$B \rightarrow A : K_{B^{-1}}$

- ◆ Transmits msg securely from B to A
 - Message is transmitted under encryption
 - Revealing secret key $K_{B^{-1}}$ does not reveal message

Composition P1; P2

◆ Sequential composition of two protocols

$A \rightarrow B : \{\text{message}\}_{K_B}$

$A \rightarrow B : K_A^{-1}$

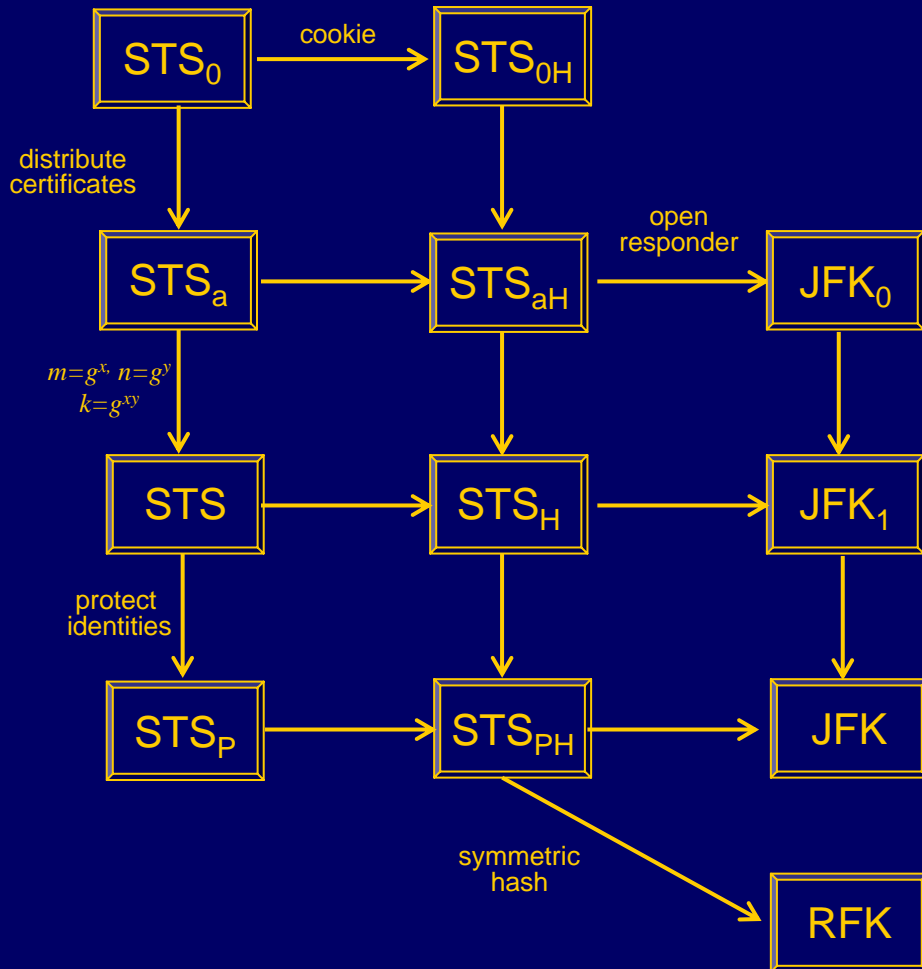
$B \rightarrow A : \{\text{message}'\}_{K_A}$

$B \rightarrow B : K_B^{-1}$

◆ Definitely not secure

- Eavesdropper learns both keys, decrypts messages

STS family



Example

- ◆ Construct protocol with properties:
 - Shared secret
 - Authenticated
 - Identity Protection
 - DoS Protection
- ◆ Design requirements for IKE, JFK, IKEv2 (IPSec key exchange protocol)

Component 1

◆ Diffie-Hellman

$$A \rightarrow B: g^a$$

$$B \rightarrow A: g^b$$

- Shared secret (with someone)

- A deduces:

$$\text{Knows}(Y, g^{ab}) \supset (Y = A) \vee \text{Knows}(Y, b)$$

- Authenticated
- Identity Protection
- DoS Protection

Component 2

◆ Challenge Response:

$A \rightarrow B: m, A$

$B \rightarrow A: n, \text{sig}_B \{m, n, A\}$

$A \rightarrow B: \text{sig}_A \{m, n, B\}$

- Shared secret (with someone)
- Authenticated
 - A deduces: $\text{Received}(B, \text{msg1}) \wedge \text{Sent}(B, \text{msg2})$
- Identity Protection
- DoS Protection

Composition

$$m := g^a$$

$$n := g^b$$

◆ ISO 9798-3 protocol:

$$A \rightarrow B: g^a, A$$

$$B \rightarrow A: g^b, \text{sig}_B \{g^a, g^b, A\}$$

$$A \rightarrow B: \text{sig}_A \{g^a, g^b, B\}$$

- Shared secret: gab
- Authenticated
- Identity Protection
- DoS Protection

Refinement

◆ Encrypt signatures:

$A \rightarrow B: g^a, A$

$B \rightarrow A: g^b, E_K \{\text{sig}_B \{g^a, g^b, A\}\}$

$A \rightarrow B: E_K \{\text{sig}_A \{g^a, g^b, B\}\}$

- Shared secret: gab
- Authenticated
- Identity Protection
- DoS Protection

Transformation

◆ Use cookie: JFK core protocol

$A \rightarrow B: g^a, A$

$B \rightarrow A: g^b, \text{hash}_{KB} \{g^b, g^a\}$

$A \rightarrow B: g^a, g^b, \text{hash}_{KB} \{g^b, g^a\}$
 $E_K \{\text{sig}_A \{g^a, g^b, B\}\}$

$B \rightarrow A: g^b, E_K \{\text{sig}_B \{g^a, g^b, A\}\}$

- Shared secret: gab
- Authenticated
- Identity Protection
- DoS Protection

(Here B must store b in step 2, but we'll fix this later...)

Cookie transformation

◆ Typical protocol

- Client sends request to server
- Server sets up connection, responds
- Client may complete session or not (DOS)

◆ Cookie version

- Client sends request to server
- Server sends hashed data back
 - Send message #2 later after client confirms
- Client confirms by returning hashed data
- Need extra step to send postponed message

Cookie in JFK

◆ Protocol susceptible to DOS

$A \rightarrow B: g^a, A$ $eh1$
 $B \rightarrow A: g^b, E_K \{sig_B \{g^a, g^b, A\}\}$
 $A \rightarrow B: E_K \{sig_A \{g^a, g^b, B\}\}$ $eh2$

◆ Use cookie: JFK core protocol

$A \rightarrow B: g^a, A$
 $B \rightarrow A: g^b, hash_{KB} \{g^b, g^a\}$
 $A \rightarrow B: g^a, g^b, hash_{KB} \{g^b, g^a\}, eh2$
 $B \rightarrow A: g^b, eh1$

Efficiency: Reuse D-H key

◆ Costly to compute g^a, g^b, g^{ab}

◆ Solution

- Keep medium-term g^a, g^b (change ~10 min)
- Replace g^a by pair g^a, nonce

◆ JFKi, JFKr protocols (except cert or grpinfo, ...)

$A \rightarrow B: N_a, g^a, A$

$B \rightarrow A: N_b, g^b, \text{hash}_{K_B} \{N_b, N_a, g^b, g^a\}$

$A \rightarrow B: N_a, N_b, g^a, g^b, \text{hash}_{K_B} \{N_b, N_a, g^b, g^a\},$
 $E_K \{\text{sig}_A \{N_a, N_b, g^a, g^b, B\}\}$

$B \rightarrow A: g^b, E_K \{\text{sig}_B \{N_a, N_b, g^a, g^b, A\}\}$

Note: B does not need to store any short-term data in step 2

Conclusion

◆ Many protocol properties

- Authentication Secrecy
- Prevent replay Forward secrecy
- Denial of service Identity protection

◆ Systematic understanding is possible

- But be careful; easy to make mistakes
- State of the art:
 need to analyze complete protocol

