

IETF Security Tutorial

Radia Perlman
November 2006

(radia.perlman@sun.com)

Why an IETF Security Tutorial?

- Security is important in all protocols; not just protocols in the security area
- IETF specs mandated to have a “security considerations” section
- There is no magic security pixie dust where you can ignore security and then plug in a security considerations section

Know the tools...

- If your protocol runs over TCP, you (mostly) don't need to worry about message retransmission and congestion control
- If your protocol runs over IP, you (mostly) don't need to worry about routing
- Isn't there something similar for security?

Sometimes

- If your protocol runs over SSL/TLS, or IPSEC, and it's reasonable to expect both ends will have appropriate credentials, you might not have to worry about security
- You're unlikely to be this lucky
 - Appropriate credentials
 - A lot of interesting protocols run at layer 3 or below (and IPsec depends on layer 3 and below)

What's hard about credentials?

- Security infrastructure rollout is late. You will probably need something lightweight as an optional alternative.
- Using the credentials from SSL and IPsec is not always easy or appropriate.

Purpose of this tutorial

- A quick intro into a somewhat scary field
- A description of what you need to know vs. what you can trust others to do
- An overview of the security WGs
- Cross-fertilization: there's no cookbook for any area, and different areas need to learn from each other

Agenda

- **Introduction to Security**
- Introduction to Cryptography
- Authenticating People
- Security mechanisms to reference rather than invent
 - Public Key / Secret Key infrastructures
 - Formats
- Security Considerations Considerations
- Security Working Groups

The Problem

- Internet evolved in a world w/out predators. DOS was viewed as illogical and undamaging.
- The world today is hostile. Only takes a tiny percentage to do a lot of damage.
- Must connect mutually distrustful organizations and people with no central management.
- And society is getting to depend on it for reliability, not just “traditional” security concerns.

Security means different things to different people

- Limit data disclosure to intended set
- Monitor communications to catch terrorists
- Keep data from being corrupted
- Destroy computers with pirated content
- Track down bad guys
- Communicate anonymously

Insecurity

*The Internet isn't insecure. It may be unsecure.
Insecurity is mental state. The users of
the Internet may be insecure, and perhaps
rightfully so.....Simson Garfinkel*

Intruders: What Can They Do?

- Eavesdrop--(compromise routers, links, routing algorithms, or DNS)
- Send arbitrary messages (including IP hdr)
- Replay recorded messages
- Modify messages in transit
- Write malicious code and trick people into running it
- Exploit bugs in software to ‘take over’ machines and use them as a base for future attacks

Some basic terms

- Authentication: “Who are you?”
- Authorization: “Should you be doing that?”
- DOS: denial of service
- Integrity protection: a checksum on the data that requires knowledge of a secret to generate (and maybe to verify)

Some Examples to Motivate the Problems

- Sharing files between users
 - File store must authenticate users
 - File store must know who is authorized to read and/or update the files
 - Information must be protected from disclosure and modification on the wire
 - Users must know it's the genuine file store (so as not to give away secrets or read bad data)
 - Users may want to know who posted the data in the file store

Examples cont'd

- Electronic Mail
 - Send private messages
 - Know who sent a message (and that it hasn't been modified)
 - Non-repudiation - ability to forward in a way that the new recipient can know the original sender
 - Anonymity
 - Virus Scanning
 - Anti-spam

Examples cont'd

- Electronic Commerce
 - Pay for things without giving away my credit card number
 - to an eavesdropper
 - or phony merchant
 - Buy anonymously
 - Merchant wants to be able to prove I placed the order

Examples, cont'd

- Routing protocol
 - Handshake with neighbor
 - Is the message from a valid router? (replay?)
 - How do we recognize a valid router?
(autoconfiguration incompatible with security)
 - Routing messages
 - Even valid routers might lie (become subverted)
 - Forwarding (which can also be DDOS'd)

Sometimes goals conflict

- privacy vs. company (or govt) wants to be able to see what you're doing
- losing data vs. disclosure (copies of keys)
- denial of service vs. preventing intrusion
- privacy vs. intrusion detection
- privacy vs. virus scanning

Agenda

- Introduction to Security
- **Introduction to Cryptography**
- Authenticating People
- Security mechanisms to reference rather than invent
 - Public Key / Secret Key infrastructures
 - Formats
- Security Considerations Considerations
- Security Working Groups

Cryptography

- It's not as scary as people make it out to be
- You don't need to know much about it to understand what it can and can't do for you

Features

- Main features
 - Encryption
 - Integrity protection
 - Authentication
- More things
 - Denial of service defense
 - Nonrepudiation
 - Perfect forward secrecy

Cryptography

- Three kinds of cryptographic algorithms you need to understand
 - secret key
 - public key
 - cryptographic hashes
- Used for
 - authentication, integrity protection, encryption

Secret Key Crypto

- Two operations (“encrypt”, “decrypt”) which are inverses of each other. Like multiplication/division
- One parameter (“the key”)
- Even the person who designed the algorithm can’t break it without the key (unless they diabolically designed it with a trap door)
- Ideally, a different key for each pair of

Secret key crypto, Alice and Bob share secret S

- $\text{encrypt} = f(S, \text{plaintext}) = \text{ciphertext}$
- $\text{decrypt} = f(S, \text{ciphertext}) = \text{plaintext}$
- authentication: send $f(S, \text{challenge})$
- integrity check: $f(S, \text{msg}) = X$
- verify integrity check: $f(S, X, \text{msg})$

A Cute Observation

- Security depends on limited computation resources of the bad guys
- (Can brute-force search the keys)
 - assuming the computer can recognize plausible plaintext
- A good crypto algo is linear for “good guys” and exponential for “bad guys”
- Faster computers work to the benefit of the good guys!

Public Key Crypto

- Two keys per user, keys are inverses of each other (as if nobody ever invented division)
 - public key “e” you tell to the world
 - private key “d” you keep private
- Yes it’s magic. Why can’t you derive “d” from “e”?
- and if it’s hard, where did (e,d) come from?

Digital Signatures

- One of the best features of public key
- An integrity check
 - calculated as $f(\text{priv key}, \text{data})$
 - verified as $f(\text{public key}, \text{data}, \text{signature})$
- Verifiers don't need to know secret
- vs. secret key, where integrity check is generated and verified with same key, so verifiers can forge data

Cryptographic Hashes

- Invented because public key is slow
- Slow to sign a huge msg using a private key
- Cryptographic hash
 - fixed size (e.g., 160 bits)
 - But no collisions! (at least we'll never find one)
- So sign the hash, not the actual msg
- If you sign a msg, you're signing all msgs with that hash!

Popular Secret Key Algorithms

- DES (old standard, 56-bit key, slow, insecure)
- 3DES: fix key size but 3 times as slow
- RC4: variable length key, “stream cipher” (generate stream from key, XOR with data), really fast, stream sometimes awkward
- AES: replacement for DES

Popular Public Key Algorithms

- RSA: nice feature: public key operations can be made very fast, but private key operations will be slow. Patent expired.
- DSS: Digital Signature Standard – pushed by U.S. government
- ECC (elliptic curve crypto): smaller keys, so faster than RSA (but not for public key ops). Some worried about patents

Popular Hashes

- Most popular hash today SHA-1 (secure hash algorithm)
- Starting to roll out: SHA-256
- Older ones (MD2, MD4, MD5) still around
- Popular secret-key integrity check: hash together key and data
- One popular standard for that within IETF: HMAC

Hash function security controversy

- Security of a hash function defined in terms of collision resistance
- In most uses, a much lower standard of security is required
- For use in HMAC, lowest of all
- MD2, MD4, MD5 “broken”. SHA-1 has “weaknesses”.
- Beware the New York Times attack!
- Make your protocols “crypto-agile”.

Crypto-agile

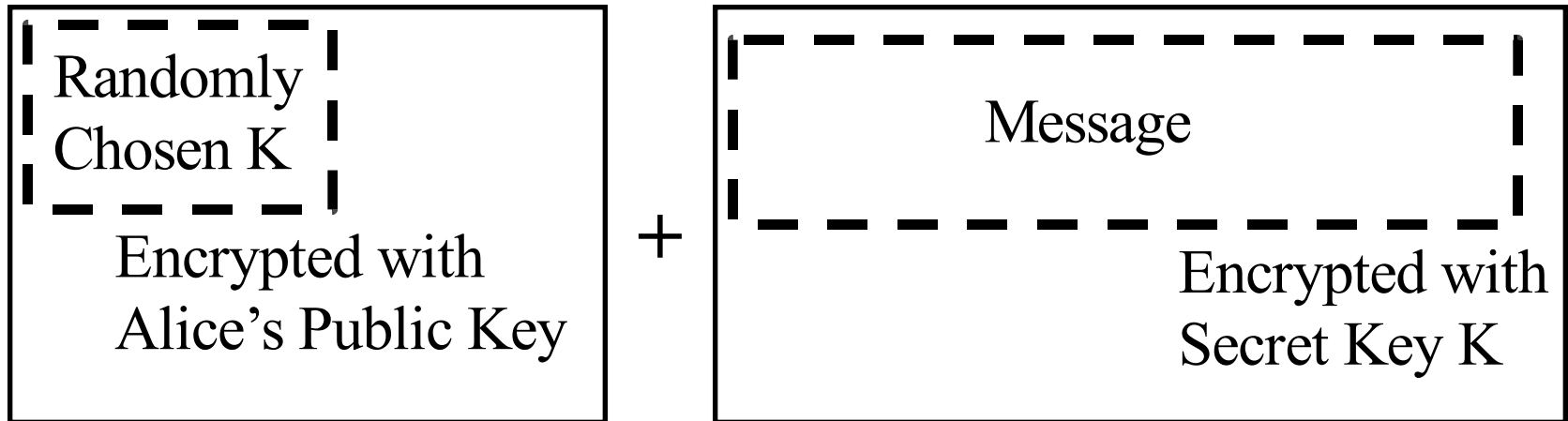
- Notice all the crypto algorithms
- The cryptographers can tell you at any time that the one you picked isn't good
- So you have to design your protocols to be able to switch crypto algorithms
- Which means for interoperability your protocol has to do negotiation

Encrypting with public key

Instead of:

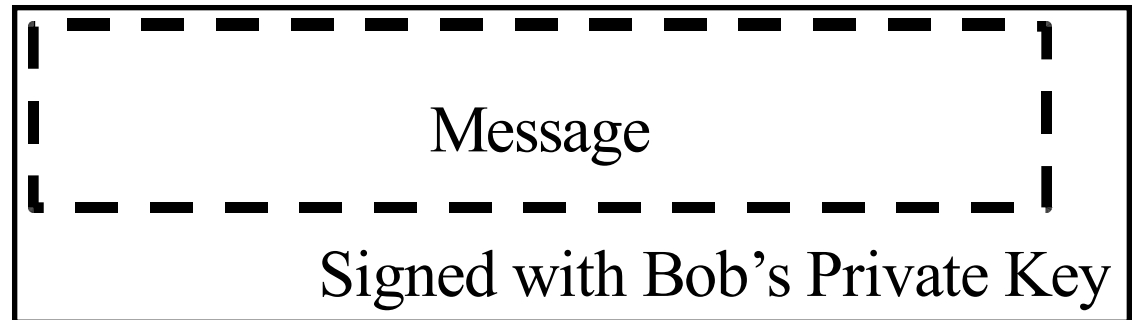


Use:



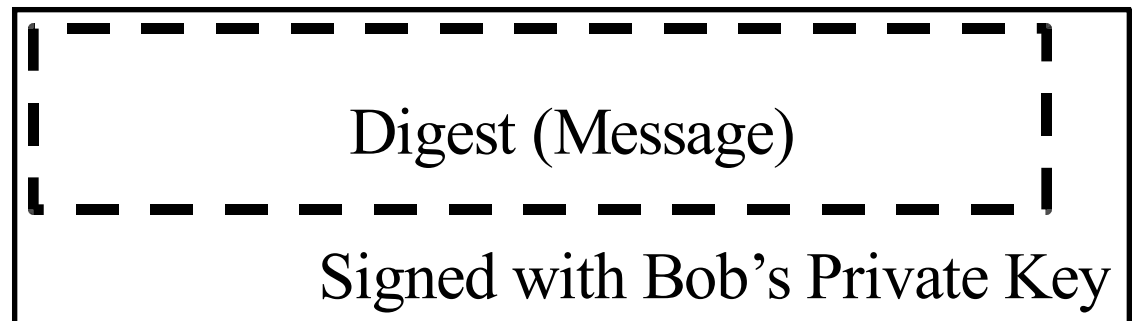
Digital Signatures

Instead of:

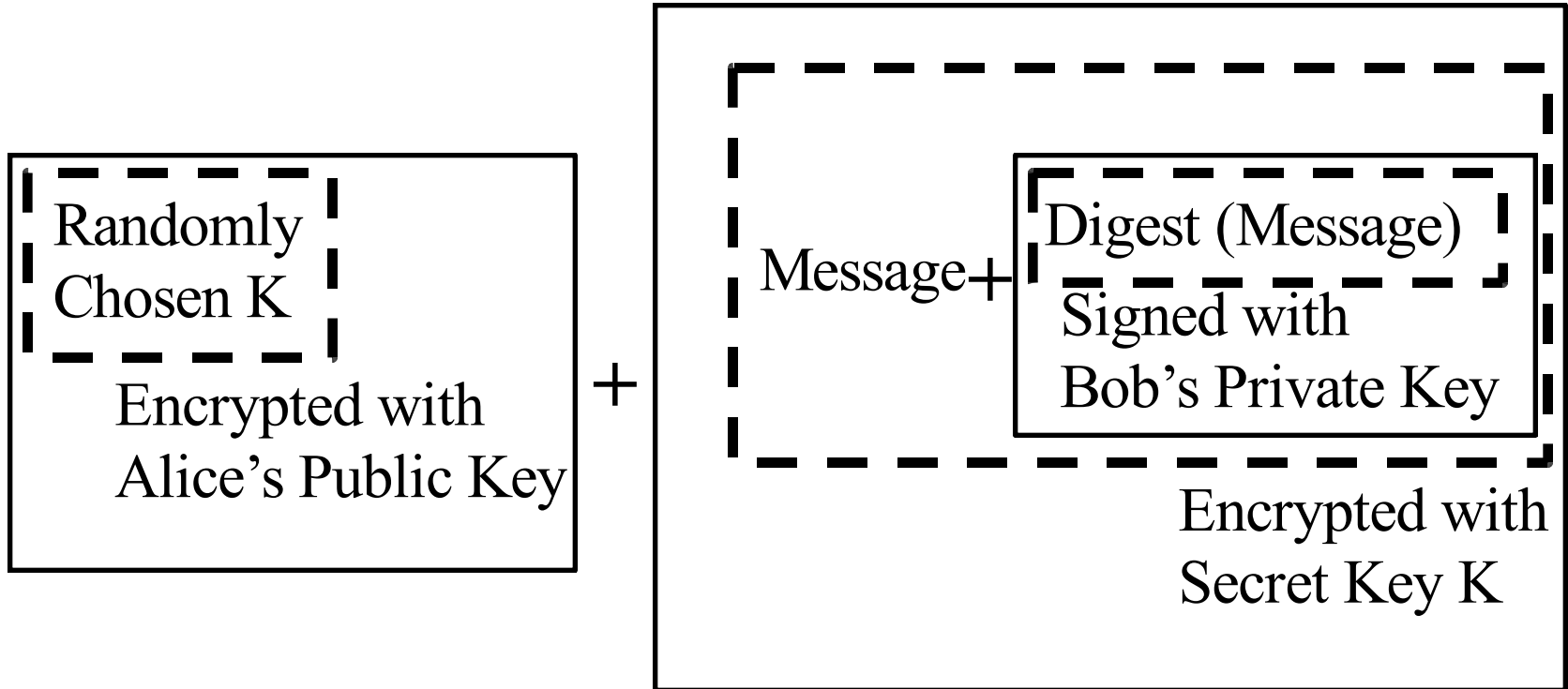


Use:

Message +



Signed and Encrypted Message



Don't try this at home

- No reason (except for the Cryptography Guild) to invent new cryptographic algorithms
- Even if you could invent a better (faster, more secure) one, nobody would believe it
- Use a well-known, well-reviewed standard

Challenge / Response Authentication

Alice (knows K)


Bob (knows K)

I'm Alice



Pick Random R
Encrypt R using K
(getting C)

If you're Alice, decrypt C



R



Non-Cryptographic Network Authentication (olden times)

- Password based
 - Transmit a shared secret to prove you know it
- Address based
 - If your address on a network is fixed and the network makes address impersonation difficult, recipient can authenticate you based on source address
 - UNIX `.rhosts` and `/etc/hosts.equiv` files

Agenda

- Introduction to Security
- Introduction to Cryptography
- **Authenticating People**
- Security mechanisms to reference rather than invent
 - Public Key / Secret Key infrastructures
 - Formats
- Security Considerations Considerations
- Security Working Groups

Authenticating people

- What you know (passwords)
- What you have (smart cards, SecurID cards, challenge/response calculators)
- What you are (biometrics)

Passwords are hard to get right!

- People “can’t” remember passwords with enough cryptographic strength to provide meaningful security as keys
- People reuse passwords, so it is dangerous to have servers storing passwords for their users
- Turn user authentication into real keys as close to the user as possible

People

- “Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed, but they are sufficiently pervasive that we must design our protocols around their limitations.”
 - Network Security: Private Communication in a Public World

Passwords ‘in the clear’ considered harmful

- Assume eavesdropping on the Internet is universal.
- Surest way to get your protocol bounced by IESG.

On-Line Password Guessing

- If guessing must be on-line, password need only be mildly unguessable
- Can audit attempts and take countermeasures
 - ATM: eat your card
 - military: shoot you
 - networking: lock account (subject to DOS) or be slow per attempt

Off-Line Password Guessing

- If a guess can be verified with a local calculation, passwords must survive a very large number of (unauditable) guesses

Passwords as Secret Keys

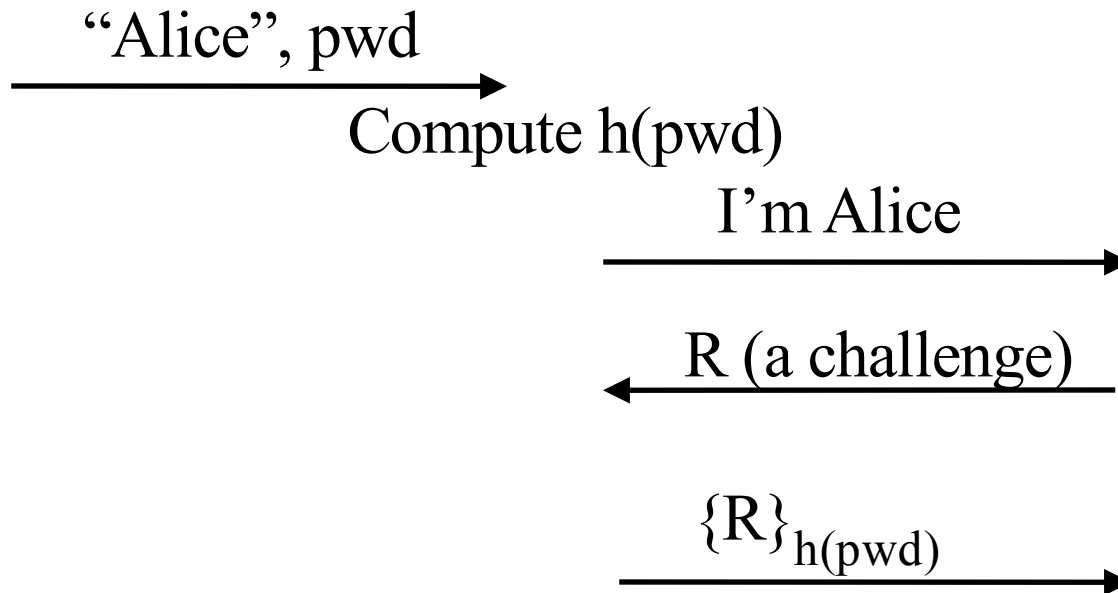
- A password can be converted to a secret key and used in a cryptographic exchange
- An eavesdropper can often learn sufficient information to do an off-line attack
- Most people will not pick passwords good enough to withstand such an attack

Off-line attack possible

Alice
(knows pwd)

Workstation

Server
(knows $h(\text{pwd})$)



Other ways of authenticating people

- OTP
- Tokens (e.g., challenge/response, time-based)
- SASL and EAP are frameworks for negotiating what kind of authentication to do

Agenda

- Introduction to Security
- Introduction to Cryptography
- Authenticating People
- **Security mechanisms to reference rather than invent**
 - **Public Key / Secret Key infrastructures**
 - **Formats**
- Security Considerations Considerations
- Security Working Groups

So what can you do?

- draft-iab-auth-mech-05.txt provides sound guidance

Security Infrastructures to Leverage

- Once your endpoints have keys, life is simpler.
- Public keys seem easier, but both are problematic.
- Kerberos uses secret keys with public key extension; TLS, IPsec, and S/MIME can use PKIX certificates

Key Distribution - Kerberos

- Could configure n^2 keys
- Instead use Key Distribution Center (KDC)
 - Everyone has one key
 - The KDC knows them all
 - The KDC assigns a key to any pair who need to talk

KDC

Alice/Ka

Alice/Ka
Bob/Kb
Carol/Kc
Ted/Kt
Fred/Kf

Ted/Kt

Bob/Kb

Fred/Kf

Carol/Kc

Key Distribution - Secret Keys

Alice

KDC

Bob

A wants to talk to B →

Randomly choose K_{ab}

← $\{\text{"B"}, K_{ab}\}_{K_a}$

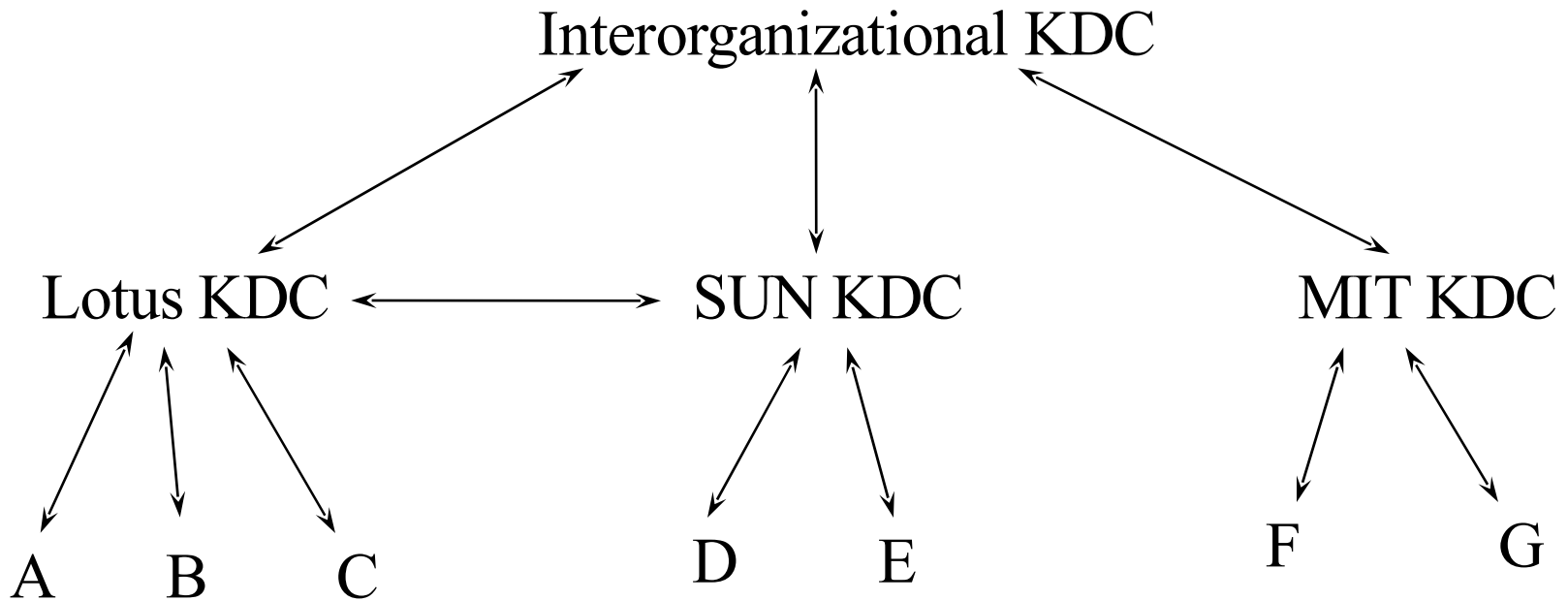
$\{\text{"A"}, K_{ab}\}_{K_b}$ →

→ $\{\text{Message}\}_{K_{ab}}$

KDC Realms

- KDCs scale up to hundreds of clients, but not millions
- There's no one who everyone in the world is willing to trust with their secrets
- KDCs can be arranged in a hierarchy so that trust is more local

KDC Realms



Key Distribution - Public Keys

- Certification Authority (CA) signs “Certificates”
- Certificate = a signed message saying “I, the CA, vouch that 489024729 is Alice’s public key”
- If everyone has a certificate, a private key, and the CA’s public key, they can authenticate

Key Distribution - Public Keys

Alice

Bob

[“Alice”, key=342872]CA

[“Bob”, key=8294781]CA

Auth, encryption, etc.

KDC vs CA Tradeoffs

- KDC solution less secure
 - Highly sensitive database (all user secrets)
 - Must be on-line and accessible via the net
 - complex system, probably exploitable bugs, attractive target
 - Must be replicated for performance, availability
 - each replica must be physically secured

KDC vs CA

- KDC more expensive
 - big, complex, performance-sensitive, replicated
 - CA glorified calculator
 - can be off-line (easy to physically secure)
 - OK if down for a few hours
 - not performance-sensitive
- Performance
 - public key slower, but avoid talking to 3rd party during connection setup

KDC vs CA Tradeoffs

- CA's work better interrealm, because you don't need connectivity to remote CA's
- Revocation levels the playing field somewhat

What formats can you leverage?

- Depends on your “layer in the stack”
- IPsec protects packets
 - Could be end to end or between firewalls
 - Today, most uses are transparent to applications
- TLS & SSH protect sessions
- OpenPGP, S/MIME and CMS, XML-DSIG and XML-encryption, protect messages (needed for store and forward)

IPsec vs. TLS

- IPsec idea: don't change applications or API to applications, just OS
- TLS idea: don't change OS, only change application (if they run over TCP)
- but... unless OS can set security context of application, server applications need to know identity of their clients

IPsec vs. TLS

- IPsec technically superior
 - Rogue packet problem
 - TCP doesn't participate in crypto, so attacker can inject bogus packet, no way for TCP to recover
 - easier to do outboard hardware processing (since each packet independently encrypted)
- TLS easier to deploy
- And unless API changes, IPsec can't pass up authenticated identity

Agenda

- Introduction to Security
- Introduction to Cryptography
- Authenticating People
- Security mechanisms to reference rather than invent
 - Public Key / Secret Key infrastructures
 - Formats
- **Security Considerations Considerations**
- Security Working Groups

Every RFC needs a “security considerations” section

- What do you have to think about?
- Not enough to say “just use IPsec”
- Sometimes (as with VRRP) protecting one protocol in a vacuum is wasted effort
 - putting expensive locks on one window, while the front door is wide open
- We don’t need to protect a protocol. We need to protect the user

Things to put in a security considerations section

- What are the threats? Which are in scope? Which aren't? (and why)
- What threats are defended against? Which are the protocol susceptible
- Implementation or deployment issues that might impact security
- See RFC 3552 “Guidelines for Writing RFC Text on Security Considerations”

Examples

- Putting integrity checks on routing msgs
 - Defends against outsiders injecting routing msgs. That's good, but
 - Doesn't prevent outsiders from answering ARPs, or corrupting DNS info
 - Doesn't protect against "Byzantine failures" (where a trusted thing goes bad)

Examples

- SNMP
- Should be straightforward end-to-end security
- But it has to work when the network is flaky
 - DNS not available
 - LDAP database for retrieving certificates might be down, as might revocation infrastructure

Examples

- Non-crypto things
 - Use up resources
 - DHCP, could request all possible addresses
 - Use all bandwidth on a link
 - Active Content
 - Too many examples of hidden places for active content!

Examples

- Email (much more detail in RFC 2552), but some cute points
 - Trivial to spoof mail
 - Message path leaks information
 - There's a protocol for asking if an email address is valid---useful for spammers
 - Even with S/MIME, header fields not protected

Example

- Kerberos Network Auth Service
- Some excerpts
 - solves authentication
 - does not address authorization or DOS or PFS
 - requires on-line database of keys, so NAS must be physically secured
 - subject to dictionary attack (pick good pwds)
 - requires reasonably synchronized clocks
 - tickets might contain private information
 - NAS must remember used authenticators to avoid replay

Agenda

- Introduction to Security
- Introduction to Cryptography
- Authenticating People
- Security mechanisms to reference rather than invent
 - Public Key / Secret Key infrastructures
 - Formats
- Security Considerations Considerations
- **Security Working Groups**

Working Groups in the Security Area: Descendents of IPsec

- btns Better-Than-Nothing Security
- pki4ipsec Profiling Use of PKI in IPSEC

Working Groups in the Security Area: Descendents of PEM

- smime S/MIME Mail Security
- pkix Public-Key Infrastructure (X.509)
- openpgp Open PGP (Pretty Good Privacy)
- Itans Long-Term Archive and Notary Services
- dkim Domain Keys Identified Mail

Working Groups in the Security Area: Descendents of CAT and SSL

- sasl Simple Authentication and Security Layer
- kitten GSS-API Next Generation
- krb-wg Kerberos
- emu EAP Method Update
- tls Transport Layer Security

Working Groups in the Security Area: Miscellaneous

- hokey Handover Keying
- isms Integrated Security Model for SNMP
- msec Multicast Security
- nea Network Endpoint Assessment
- syslog Security Issues in Network Event Logging

Noteworthy Completed WGs

- dnssec Securing DNS and storing keys in DNS
- otp One-time password protocol

Security Working Groups outside the Security Area

- eap Extensible Authentication Protocol
- pana Protocol for carrying
Authentication for Network Access
- aaa Authentication, Authorization, and
Accounting
- radext Radius Extensions
- dime Diameter Maintenance and Extensions
- rpsec Routing Protocol Security
Requirements

Conclusions

- *Until a few years ago, you could connect to the Internet and be in contact with hundreds of millions of other nodes, without giving even a thought to security. The Internet in the '90's was like sex in the '60's. It was great while it lasted, but it was inherently unhealthy and was destined to end badly. I'm just really glad I didn't miss out again this time.* —Charlie Kaufman