

## Intruders: What Can They Do?

- Eavesdrop
- Send Messages
- Impersonate an address and lie in wait
- Replay recorded messages
- Modify messages in transit
- Write malicious code and trick people into running it

1

## 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)

2

## 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

3

## 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

4

## Sometimes goals conflict

- nonrepudiation vs plausible deniability
- privacy vs company (or govt) wants to be able to see what you're doing
- avoiding false positives vs false negatives
- safety vs functionality
- losing data vs disclosure (copies of keys)
- denial of service vs preventing intrusion

5

## Quick Overview

- We're going to need cryptography
- It allows you to prove you know a secret without divulging it
- If Alice and Bob share a secret:
  - Bob can know he's talking to Alice
  - Bob can encrypt a message for Alice
  - Bob can know nobody has tampered with a message from Alice

6

## Overview cont'd

- Public key crypto allows Alice to prove to Bob she knows her secret without Bob knowing her secret
- Securely distributing keys (secret key systems like Kerberos, vs PKIs)
- Session protocols (e.g., SSH, SSL, IPSEC)
- Email (e.g., S/MIME, PGP)

7

## Overview cont'd

- There are lots of variants of schemes, because multiple independent organizations simultaneously worked on the problems
- There are a few basic crypto tricks that are the basis of all these protocols
- We cover the toolkit of crypto tricks
- Then we explain the specifics of the alphabet-soup of protocols

8

## Active Content: Threat or Menace?

- If you run a program I wrote, it can do things with your rights behind your back
  - Read your private files and send them to me
  - Delete or mangle files you have rights to access
  - Send email composed by me but sent (proveably!) by you
  - Authorize me to do things later (if you can add me to ACLs)

9

## Active Content: What were they thinking!? (or... why can't I only run software from trustworthy sources?)

- Bandwidth and Storage Efficiency
  - A program to generate a graphic could be much smaller than the bitmap (particularly for animations)
- Extensibility
  - Application designer can add capabilities not envisioned by the platform designer
- Push computation out to the client
  - Where CPU cycles tend to be cheaper

10

## ...and even if the source is Trustworthy

- The program must be bug-free or it might introduce security problems
- How do you know this is genuine? (e.g. when downloading from the web)
- You're not just trusting that source, but all sources they've ever trusted...

11

## Digital Pests

- Trojan horse: malicious code hidden inside an otherwise useful program (waiting for someone with interesting privileges to run it)
- Virus: malicious code hidden inside a program that when run "reproduces" by installing copies of itself inside programs the person running it has permission to modify

12

## Digital Pests

- Worm: A program that replicates over a network by finding nodes willing to accept copies and run them
- Trapdoor: An undocumented entry point intentionally written into a program
- Logic Bomb: malicious code triggered on a future event
- Letter Bomb: malicious code executed upon opening an email message

13

## Spreading Pests

- Booting from an infected floppy disk
- Loading and executing infected software from the Internet or other untrusted source
- Extracting and running untrustworthy code from an email message
- Displaying a Postscript or Word file
- Email with “autolaunch” capability
- Bugs (e.g., bounds checking)

14

## What Protection Is There?

- Decent operating systems
- Interpreted languages in “sandboxes”
- Digitally signed content
- Content scanners (at WS or Firewall)
- Connectivity restrictions (through Firewall)
- Educating users
- Genetic diversity

15

## Legal Issues Past (hopefully)

- All Public Key cryptographic algorithms were patented until September 1997.
- RSA patent expired September 20, 2000
- Patents in general a real problem.
- Export controls
- Usage controls

16

## Section: Cryptography

- Three kinds of cryptographic algorithms
  - Secret Key Cryptography (DES, IDEA, RCx, AES)
  - Public Key Cryptography (RSA, Diffie-Hellman, DSS)
  - Message Digests (MD4, MD5, SHA-1)

17

## Secret Key Cryptography

- Originally a way to keep secret data private
  - Encode a message using a secret “key”
  - A long and colorful history
- Today, it has many uses
  - Privacy
  - Authentication
  - Data Integrity

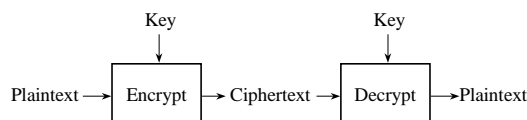
18

## What is Encryption?

- You and I agree on a secret way to transform data
- Later, we use that transform on data we want to pass over an unsafe communications channel
- Instead of coming up with new transforms, design a common algorithm customized with a “key”

19

## Secret Key Encryption for Privacy



20

## How Secure is Encryption?

- An attacker who knows the algorithm we're using could try all possible keys
- Security of cryptography depends on the limited computational power of the attacker
- A fairly small key (e.g. 64 bits) represents a formidable challenge to the attacker
- Algorithms can also have weaknesses, independent of key size

21

## How Practical is Encryption

- Usability depends on being efficient for the good guys
- Cost to the good guys tends to rise linearly with key length
- Cost to search all keys rises exponentially with key length
- Therefore advances in computer speed work to the advantage of the good guys!

22

## How do we know how good an algorithm is?

- A problem of mathematics: it is very hard to prove a problem is hard
- It's never impossible to break a cryptographic algorithm - we want it to be as hard as trying all keys
- Fundamental Tenet of Cryptography: *If lots of smart people have failed to solve a problem then it probably won't be solved (soon)*

23

## To Publish or Not to Publish

- If the good guys break your algorithm, you'll hear about it
- If you publish your algorithm, the good guys provide free consulting by trying to crack it
- The bad guys will learn your algorithm anyway
- Today, most commercial algorithms are published; most military algorithms are not

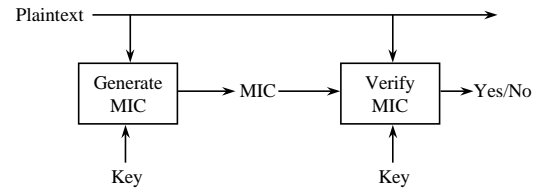
24

## Uses of Cryptography

- Transmitting secret data over an insecure channel
- Storing secret data on an insecure medium
- Message integrity checksum/authentication code (MIC/MAC)
- Authentication: “challenge” the other party to encrypt or decrypt a random number

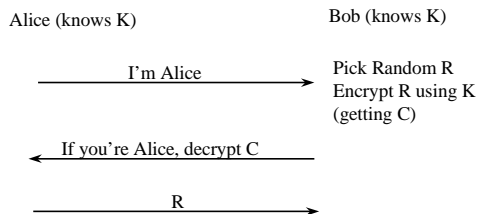
25

## Secret Key Integrity Protection



26

## Challenge / Response Authentication



27

## Secret Key Algorithms

- DES (Data Encryption Standard)
  - 56 bit key (+ 8 parity bits) controversial!
  - Input and output are 64 bit blocks
  - slow in software, based on (sometime gratuitous) bit diddling
- IDEA (International Data Encryption Algorithm)
  - 128 bit key
  - Input and output are 64 bit blocks
  - designed to be efficient in software

28

## Secret Key Algorithms

- Triple DES
  - Apply DES three times (EDE) using K1, K2, K3 where K1 may equal K3
  - Input and output 64 bit blocks
  - Key is 112 or 168 bits
- Advanced Encryption Standard (AES)
  - New NIST standard to replace DES.
  - Public Design and Selection Process. Rijndael.
  - Key Sizes 128,192,256. Block size 128.

29

## Secret Key Algorithms

- RC2 (Rivest's Cipher #2)
  - Variable key size
  - Input and output are 64 bit blocks
- RC4 (Rivest's Cipher #4)
  - Variable key size
  - Extremely efficient
  - Stream cipher - one time use keys
- Many other secret key algorithms exist
- It is hard to invent secure ones!
- No good reason to invent new ones

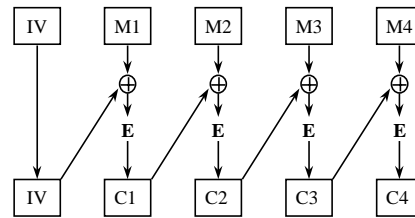
30

## Encrypting Large Messages

- The basic algorithms encrypt a fixed size block
- Obvious solution is to encrypt a block at a time. This is called Electronic Code Book (ECB)
- Repeated plaintext blocks yield repeated ciphertext blocks
- Other modes “chain” to avoid this (CBC, CFB, OFB)
- Encryption does not guarantee integrity!

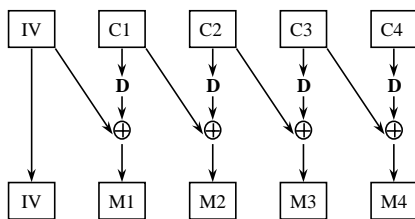
31

## CBC (Cipher Block Chaining)



32

## CBC Decryption



33

## XOR (Exclusive-OR)

- Bitwise operation with two inputs where the output bit is 1 if exactly one of the two input bits is one
- $(B \text{ XOR } A) \text{ XOR } A = B$
- If A is a “one time pad”, very efficient and secure
- Common encryption schemes (e.g. RC4) calculate a pseudo-random stream from a key

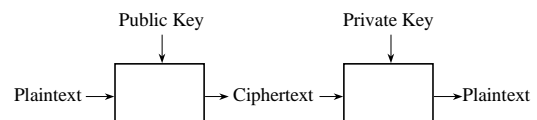
34

## Public Key Cryptography

- Two keys per user: a private key and a public key. The keys reverse each other’s effects.
- Encrypt a message for Alice using her public key
- Decryption requires her private key
- Generating Digital Signatures requires the private key
- Verifying them requires the public key

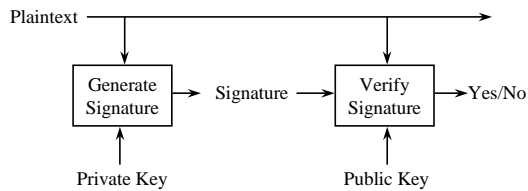
35

## Public Key Encryption for Privacy



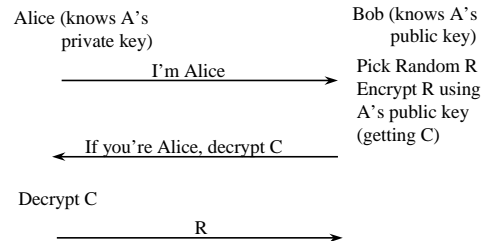
36

## Public Key Integrity Protection



37

## Public Key Authentication



38

## How Public Key Algorithms Work

- We want an algorithm with the following properties:
  - two different numbers: e and d
  - e and d are inverses; using one reverses the effect of the other
  - you shouldn't be able to compute d from e
  - it must be efficient to find a matching pair of keys
  - it must be efficient to encrypt and decrypt

39

## Example (Insecure) Public Key Algorithm

- Multiplication modulo p (where p is a prime)
- For example, let  $p=127$
- Choose e and d so that  $e \cdot d \equiv 1 \pmod{127}$ 
  - e.g.  $e=53$  and  $d=12$
- To encrypt a number, multiply by 53 mod 127
- To decrypt a number, multiply by 12 mod 127
- Decryption must restore the initial value!

40

## Why Isn't This Secure?

- The number 127 is too small. You could compute d from e by trying all possible values
- Modular division is possible - the inverse can be computed quickly even when p is large (Euclid's algorithm...patent long expired)

41

## A Summary of RSA

- Named after its inventors: Rivest, Shamir, and Adelman
- Uses modular exponentiation
- Choose a modulus n and a public exponent e
- Public key encryption is:  
 $\text{ciphertext} = \text{plaintext}^e \pmod{n}$
- Public key decryption is:  
 $\text{plaintext} = \text{ciphertext}^d \pmod{n}$

42

## A Summary of RSA

- If you can find  $d$  from  $e$ , why can't someone else?
- Factoring large numbers is hard
- Finding  $d$  from  $e$  is easy if you can factor  $n$ , but it's hard if you can't
- Pick two large primes and multiply them together to get  $n$ . You can now factor  $n$  because you constructed  $n$
- After computing  $d$  from  $e$ , you can forget the factors of  $n$

43

## Theory of RSA

- Define  $\Phi(n)$  to be the # of integers  $< n$  and relatively prime to  $n$
- If  $p$  is a prime,  $\Phi(p) = p-1$
- Euler proved:  $x^{\Phi(n)} \bmod n = 1$
- So  $x^{\Phi(n)+1} \bmod n = x$
- If we can find  $d \cdot e = 1 \bmod \Phi(n)$ , they'd be "exponentiative inverses"

44

## Theory of RSA, cont'd

- If  $n=p \cdot q$  ( $p, q$  primes),  $\Phi(n)=(p-1)(q-1)$  (remove multiples of  $p$  and multiples of  $q$ )
- Given  $e, Z$ , Euclid's algorithm allows us to compute  $d$  such that  $d \cdot e = 1 \bmod Z$
- So, we can find  $d$  from  $e$  if we know  $\Phi(n)$
- We need to know how to factor  $n$  in order to know  $\Phi(n)$

45

## How to Find Large Primes

- If factoring is hard, how do you find large primes?
- It turns out you can test a number for primality easily even though factoring is hard!
- Pick random large numbers and test them until you find a prime one

46

## Doing exponentiation

- Can't multiply something by itself a gazillion google times!
- Solution: Repeated squaring
- Result: a 512 bit exponent requires between 512 and 1024 multiplications (depending on the number of 1's in the number) (instead of a trillion trillion trillion trillion google multiplications)

47

## How Big Do The Numbers Have To Be?

- Since RSA became popular, major advances in the mathematics of factoring
- The biggest numbers chosen RSA-style to have been factored are of size 400-500 bits. This took thousands of MIP-years on many workstations
- 512 bit RSA keys are commonly used but not entirely safe
- 768-2048 bits are recommended

48



## How Big Do The Numbers Have To Be?

- Without losing security,  $e$  can be small
- This makes public key operations much more efficient than private key operations
- $e = 3$  and  $e = 2^{16} + 1$  are popular
- Care must be taken when  $e = 3$

29

## Our Notation

Encryption: Curly brackets followed by name of key

$\{data\}_{K_{AB}}$

Signed info: Square brackets followed by name of key

$[information]_{Alice}$

30

## An Intuition for Diffie-Hellman

- Allows two individuals to agree on a secret key, even though they can only communicate in public
- Alice chooses a private number and from that calculates a public number
- Bob does the same
- Each can use the other's public number and their own private number to compute the same secret
- An eavesdropper can't reproduce it

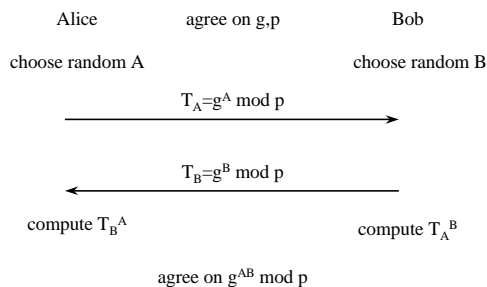
31

## Why is D-H Secure?

- We assume the following is hard:
- Given  $g$ ,  $p$ , and  $g^x \bmod p$ , what is  $x$ ?
- With the best known mathematical techniques, this is somewhat harder than factoring a composite of the same magnitude as  $p$
- Subtlety: they haven't proven that the algorithms are as hard to break as the underlying problem

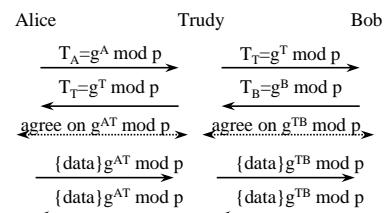
32

## Diffie-Hellman



33

## Man in the Middle



34

# Signed Diffie-Hellman (Avoiding Man in the Middle)

Alice

Bob

choose random A

choose random B

$[T_A = g^A \bmod p]$  signed with Alice's Private Key

$[T_B = g^B \bmod p]$  signed with Bob's Private Key

verify Bob's signature

verify Alice's signature

agree on  $g^{AB} \bmod p$

55

# Cookie Mechanism:

## Some Denial of Service Protection

The diagram illustrates a secure communication protocol between Alice and Bob. It consists of several steps represented by arrows and text:

- Initial State:** Alice and Bob are at the top of the diagram.
- Step 1:** Alice sends a message to Bob: "I'm Alice".
- Step 2:** Bob responds to Alice: "choose cookie C".
- Step 3:** Alice sends a message to Bob:  $C, [T_A = g^A \bmod p]$  signed with Alice's Private Key.
- Step 4:** Bob sends a message to Alice: "verify C".
- Step 5:** Alice sends a message to Bob:  $[T_B = g^B \bmod p]$  signed with Bob's Private Key.
- Step 6:** Bob sends a message to Alice: "verify Alice's signature".
- Step 7:** Alice sends a message to Bob: "verify Bob's signature".
- Final Step:** Both Alice and Bob agree on  $g^{AB} \bmod p$ .

56

## Stateless Cookies

- It would be nice if Bob not only can avoid doing computation, but can avoid using any state until he knows the other side is sending from a reasonable IP address
- A “stateless” cookie is one Bob can verify without maintaining per connection state
- For instance, Bob can have a secret  $S$ , and cookie = {IP address} $S$

57

# Diffie-Hellman for Encryption

Alice	Bob
	choose $g, p$
	choose random $B$
	publish $g, p, T_B = g^B \bmod p$
choose random $A$	
compute $T_A = g^A \bmod p$	
compute $T_B^A$	
encrypt message using $g^{AB} \bmod p$	
send $T_A$ , encrypted msg	compute $T_A^B$
	decrypt message using $g^{AB} \bmod p$

58

## DSS / ElGamal Signatures

- ElGamal's alg: signatures using Diffie-Hellman keys
- Govt modified it and standardized it as DSS
- DSS similar, better performance, but doesn't use Diffie-Hellman keys
- Initially only published DSS signatures but later also published encryption (KEA Key exchange algorithm)

59

## IETF Politics

- Diffie-Hellman patent expired in September 1997. DSS patent is freely licensed.
- RSA licensed on onerous terms.
- Prior to 1997, Diffie-Hellman was licensed on even more onerous terms, so most deployed systems use RSA (which is also technically superior in most uses)

## IETF Politics

- IESG was “reluctant” to approve anything mandating use of RSA
- Instead mandated DSS for signatures and Diffie-Hellman for encryption
- Now that RSA expired (9/20/00), heated debate over MUST and SHOULD vis a vis RSA and DH/DSS

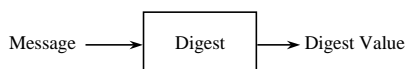
61

## Message Digest Functions

- Also known as cryptographic hashes
- Non-reversible function
- Takes an arbitrary size message and mangles it into a fixed size digest
- It should be impossible to find two messages with the same MD, or come up with a message with a given MD
- Useful as a shorthand for a longer thing

62

## Message Digest Functions



63

## Message Digest Functions

- The most popular ones are MD2, MD4, and MD5
- All produce 128 bit digests
- MD4 was recently “broken” and MD2 and MD5 have significant weaknesses
- SHA-1 was proposed by the U.S. government. It produces a 160 bit digest
- Message digests are not difficult to design, but most are not secure

64

## Keyed Message Digests

- If you combine a Message with a secret quantity and digest the combination, you get a MIC (Message Integrity Code)
- This may have better performance and similar security than a MIC based on a secret key cryptographic algorithm

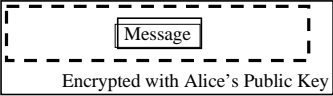
65

## Combining Cryptographic Functions for Performance

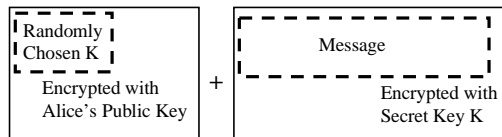
- Public key cryptography is slow compared to hashes and secret key cryptography
- Public key cryptography is more convenient & secure in setting up keys
- Algorithms can be combined to get the advantages of both

66

## Hybrid Encryption

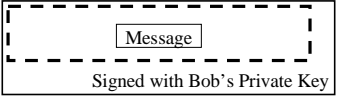
Instead of: 

Use:



67

## Hybrid Signatures

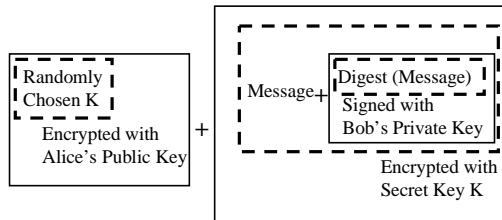
Instead of: 

Use:



68

## Signed and Encrypted Message



69

## Section: Authentication

- Non-cryptographic authentication
- Special problems with people
- Cryptographic authentication
- Key distribution: KDCs and CAs

70

## Non-Cryptographic Network Authentication

- Password based
  - Transmit a shared secret to prove you know it (e.g. cellular phones)
- 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

71

## Authentication of People

- What you know
- What you have
- What you are

72

## What You Know...

- Mostly this means passwords
  - Subject to eavesdropping
  - Subject to theft of password database
  - Subject to on-line guessing
  - Subject to off-line guessing
- How can you force people to choose good passwords?

73

## What You Have...

- Passive Devices (physical key, mag stripe card)
- Smart Cards
  - PIN activated memory
  - Display on card (no reader necessary)
  - Display and keyboard on card
  - Special reader w/electrical connection
    - Crypto on the card
    - Secret never leaves the card
    - PCMCIA, SmartDisk, ISO format

74

## What You Are...

- Biometric Devices
  - Retinal Scanners
  - Signature Verifiers
  - Fingerprint Readers
- Limitations
  - Expensive
  - Users hate them
  - Not useful for network authentication (though possibly as an adjunct)

75

## On-Line Password Guessing

- If guessing must be on-line, password need only be mildly unguessable
- ATM machine eats your card after third wrong PIN
- Military: they arrest you after one wrong attempt
- Computers
  - Lock out account after 'n' tries
  - Process attempts slowly
  - Audit failed attempts and alert an administrator

76

## Off-Line Password Guessing

- If a guess can be verified with a local calculation, passwords must survive a very large number of guesses
- Unix password database was world readable and held one-way hashes of passwords
- Once you read the database, you can take it back to all the Crays in your basement and have them guess passwords

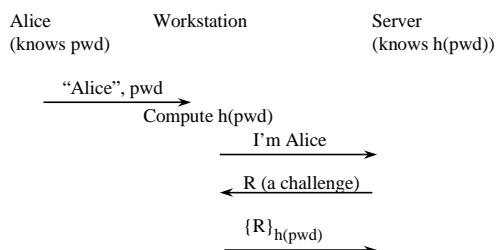
77

## 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

78

## Sample Protocol



79

## Password Guessing

- "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

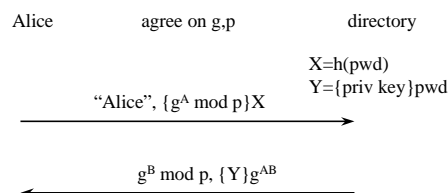
80

## Obtaining Human's Private Key

- Encrypt the private key using the password as a key
  - Put the encrypted private key on a floppy or mag-stripe card
  - Post the encrypted private key in a public place
  - Post the encrypted private key on a server that limits pwd guessing with a careful protocol
- Put the private key on a smart card

81

## Obtaining private key



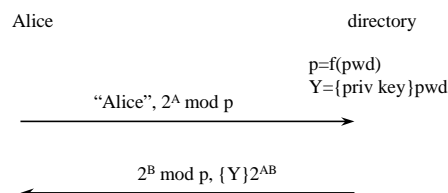
82

## Our new scheme PDM

- "Password Derived Modulus"
- Previous slide used protocol based on EKE, patented by Bellare-Meritt
- Another appropriate scheme is SPEKE, patented by Jablon, in which  $g = h(pwd)$
- PDM, unpatented, chooses  $p$  based on the password, uses 2 as the base.

83

## PDM



84

## PDM Properties

- Expensive at client (10 sec for 500-bit p)
- But fast enough, and single char “hint” chooses even 1000-bit prime in <10 sec
- Less expensive at server, since smaller p is secure enough
- Server performance most important
- Unpatented

85

- ## PDM Properties
- Expensive at client (10 sec for 500-bit p)
  - But fast enough, and single char “hint” chooses even 1000-bit prime in <10 sec
  - Less expensive at server, since smaller p is secure enough
  - Server performance most important
  - Unpatented
- 85

28

# Key Distribution - Secret Keys

- What if there are millions of users and thousands of servers?
- Could configure  $n^2$  keys
- Better is to use a Key Distribution Center
  - Everyone has one key
  - The KDC knows them all
  - The KDC assigns a key to any pair who need to talk

86

- # Key Distribution - Secret Keys
- What if there are millions of users and thousands of servers?
  - Could configure  $n^2$  keys
  - Better is to use a Key Distribution Center
    - Everyone has one key
    - The KDC knows them all
    - The KDC assigns a key to any pair who need to talk
- 86

92

# Key Distribution - Secret Keys

Alice                      KDC                      Bob

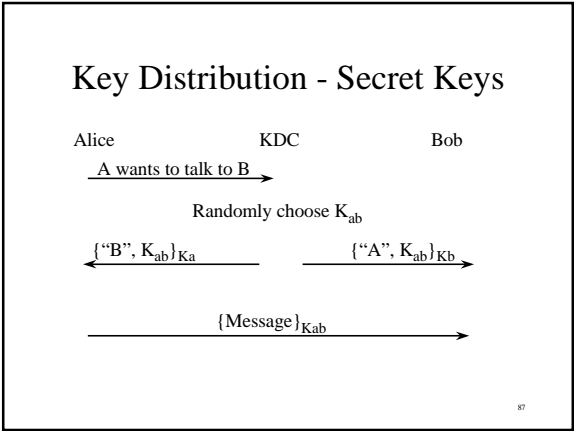
A wants to talk to B →

Randomly choose  $K_{ab}$

←  $\{\text{"B"}, K_{ab}\}_{K_a}$                        $\{\text{"A"}, K_{ab}\}_{K_b}$  →

\_\_\_\_\_  $\{\text{Message}\}_{K_{ab}}$  →

87



87

# A Common Variant

Alice                      KDC                      Bob

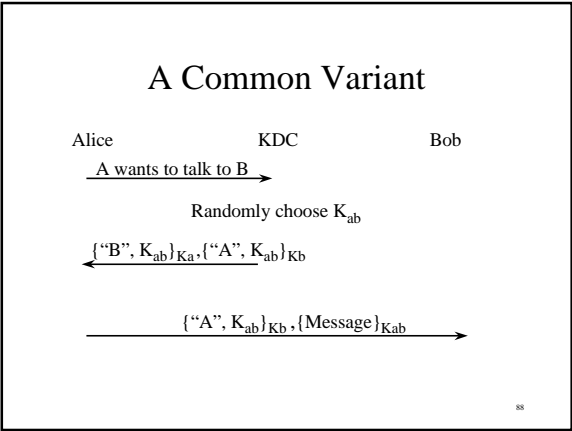
A wants to talk to B →

Randomly choose  $K_{ab}$

←  $\{\text{"B"}, K_{ab}\}_{K_a}, \{\text{"A"}, K_{ab}\}_{K_b}$

→  $\{\text{"A"}, K_{ab}\}_{K_b}, \{\text{Message}\}_{K_{ab}}$

88



2

## 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
- 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

89

# KDC Realms

```
graph TD; Interorganizational_KDC[Interorganizational KDC] <--> Lotus_KDC[Lotus KDC]; Interorganizational_KDC <--> SUN_KDC[SUN KDC]; Interorganizational_KDC <--> MIT_KDC[MIT KDC]; Lotus_KDC <--> SUN_KDC; Lotus_KDC --> A; Lotus_KDC --> B; Lotus_KDC --> C; SUN_KDC --> D; SUN_KDC --> E; MIT_KDC --> F; MIT_KDC --> G;
```

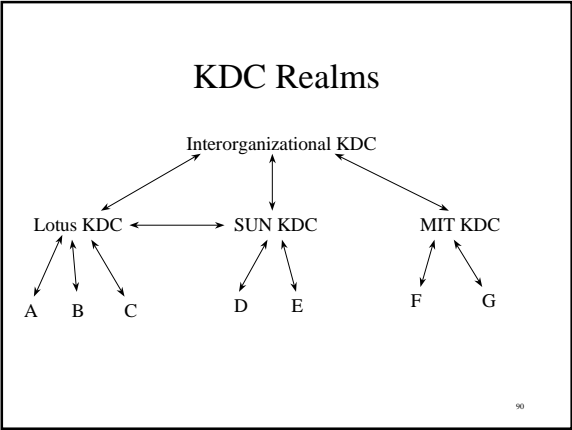
Diagram illustrating KDC Realms:

- Interorganizational KDC (Central Realm)
- Lotus KDC (Left Realm)
- SUN KDC (Middle Realm)
- MIT KDC (Right Realm)

Connections:

- Interorganizational KDC is connected to Lotus KDC, SUN KDC, and MIT KDC.
- Lotus KDC is connected to SUN KDC.
- Lotus KDC is connected to A, B, and C.
- SUN KDC is connected to D and E.
- MIT KDC is connected to F and G.

90



90

## KDC Hierarchies

- KDCs on the path between two principals can impersonate them to one another and decrypt eavesdropped conversations
- KDCs high in the hierarchy can compromise lots of interactions, but not those within organizations
- Finding the appropriate KDCs along the path can be difficult

91

## Key Distribution - Public Keys

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

92

## KDC vs CA Tradeoffs

- Stealing the KDC database allows impersonation of all users and decryption of all previously recorded conversations
- Stealing the CA Private keys allows forging of certificates and hence impersonation of all users, but not decryption of recordings
- Recovering from a CA compromise is easier because user keys need not change

93

## KDC vs CA Tradeoffs

- KDC must be on-line and have good performance at all times
- CA need only be used to create certificates for new users
  - It can be powered down and locked up, avoiding network based attacks
- CA’s work better interrealm, because you don’t need connectivity to remote CA’s

94

## KDC vs CA Tradeoffs

- Public Key cryptography is slower and (used to) require expensive licenses
- The “revocation problem” levels the playing field somewhat

95

## Strategies for CA Hierarchies

- One universally trusted organization
- Top-Down, starting from a universally trusted organization’s well-known key
- No rules (PGP, SDSI, SPKI). Anyone signs anything. End users decide who to trust
- Many independent CA’s. Configure which ones to trust

96



## One CA

- Choose one universally trusted organization
- Embed their public key in everything
- Give them universal monopoly to issue certificates
- Make everyone get certificates from them
- Simple to understand and implement

97

## One CA: What's wrong with this model?

- Monopoly pricing
- Getting certificate from remote organization will be insecure or expensive (or both)
- That key can never be changed
- Security of the world depends on honesty and competence of the one organization, forever

98

## One CA Plus RAs

- RA (registration authority), is someone trusted by the CA, but unknown to the rest of the world (verifiers).
- You can request a certificate from the RA
- It asks the CA to issue you a certificate
- The CA will issue a certificate if an RA it trusts requests it
- Advantage: RA can be conveniently located

99

## What's wrong with one CA plus RAs?

- Still monopoly pricing
- Still can't ever change CA key
- Still world's security depends on that one CA key never being compromised (or dishonest employee at that organization granting bogus certificates)

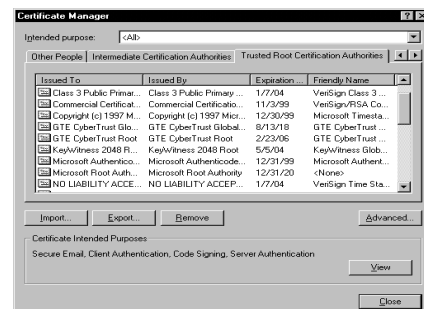
100

## Oligarchy of CAs

- Come configured with 50 or so trusted CA public keys
- Usually, can add or delete from that set
- Eliminates monopoly pricing

101

## Default Trusted Roots in IE



102

## What's wrong with oligarchy?

- Less secure!
  - security depends on ALL configured keys
  - naïve users can be tricked into using platform with bogus keys, or adding bogus ones (easier to do this than install malicious software)
- Although not monopoly, still favor certain organizations

103

## CA Chains

- Allow configured CAs to issue certs for other public keys to be trusted CAs
- Similar to CAs plus RAs, but
  - Less efficient than RAs for verifier (multiple certs to verify)
  - Less delay than RA for getting usable cert

104

## Anarchy

- Anyone signs certificate for anyone else
- Like configured+delegated, but user consciously configures starting keys
- Problems
  - won't scale (too many certs, computationally too difficult to find path)
  - no practical way to tell if path should be trusted
  - too much work and too many decisions for user

105

## Top Down with Name Subordination

- Assumes hierarchical names
- Each CA only trusted for the part of the namespace rooted at its name
- Can apply to delegated CAs or RAs
- Easier to find appropriate chain
- More secure in practice (this is a sensible policy that users don't have to think about)

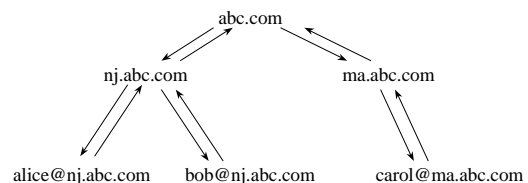
106

## Bottom-Up Model

- Each arc in name tree has parent certificate (up) and child certificate (down)
- Name space has CA for each node
- "Name Subordination" means CA trusted only for a portion of the namespace
- Cross Links to connect Intranets, or to increase security
- Start with your public key, navigate up, cross, and down

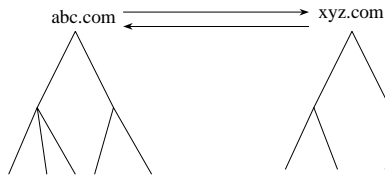
107

## Intranet



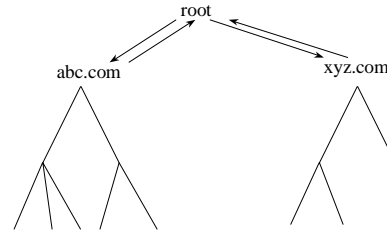
108

## Extranets: Crosslinks



109

## Extranets: Adding Roots



110

## Advantages of Bottom-Up

- For intranet, no need for outside organization
- Security within your organization is controlled by your organization
- No single compromised key requires massive reconfiguration
- Easy configuration: public key you start with is your own

111

## Name Constraints

- Standard certificate format allows “name constraint” extension
  - allowed names
  - excluded names
- You can build any of these models (or any hybrid) with this

112

## Bridge CA Model

- Similar to bottom-up, in that each organization controls its destiny, but top-down within organization
- Trust anchor is the root CA for your org
- Your org’s root points to the bridge CA, which points to other orgs’ roots

113

## What is X.509?

- A clumsy syntax for certificates
  - No rules specified for hierarchies
  - X.509 v1 and v2 allowed only X.500 names and public keys in a certificate
  - X.509 v3 allows arbitrary extensions
- A dominant standard
  - Because it is flexible, everyone willing to use it
  - Because it is flexible, all hard questions remain

114

## X.509 Certificate Contents

- version # (1, 2, or 3)
- Serial Number
- Effective Date
- Expiration Date
- Issuer Name
- Issuer UID (not in V1)
- Subject Name
- Subject UID (not in V1)
- Subject Public Key Algorithm
- Subject Public Key
- Signature Algorithm
- Signature
- Extensions (V3 only)

115

## X.509 V3 Extensions

- Public Key Usage
  - Encryption
  - Signing
  - Key Exchange
  - Non-repudiation
- Subject Alternate Names
- Issuer Alternate Names
- Key Identifiers
- Where to find CRL information
- Certificate Policies
- “Is a CA” flag
  - path length constraints
  - name constraints
- Extended key usage
  - specific applications

116

## Other Certificate Standards

- PKIX: an IETF effort to standardize extensions to X.509 certificates
  - Still avoids hard decisions
  - Anything possible with PKIX
- SPKI: a competing IETF effort rejecting X.509 syntax
- SDSI: a proposal within SPKI for certificates with relative names only

117

## Revocation Problem

- Suppose a bad guy learns your password or steals your smart card...
- Notify your KDC and it will stop issuing “tickets”
- Notify your CA and it will give you a new certificate
- How do you revoke your old certificate?

118

## Revocation Problem

- Tickets can have short lifetimes; they can even be “one-use” with nonces
- Certificates have expiration dates, but it is inconvenient to renew them frequently
  - If sufficiently frequent and automated, CA can no longer be off-line
- Supplement certificate expirations with Certificate Revocation Lists (CRLs) or a blacklist server

119

## Why not put CA on-line?

- On-line revocation server is less security sensitive than an on-line CA
- The worst it can do is fail to report a revoked certificate
- Damage is more contained
- Requires a double failure
- With CRLs, limits OLRs damage

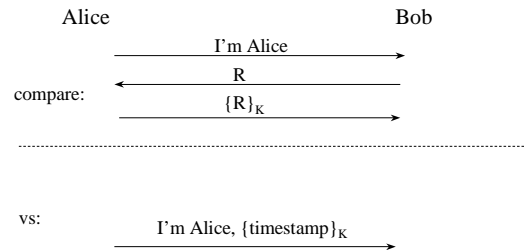
120

## Cryptographic Handshakes

- Once keys are known to two parties, a cryptographic handshake to authenticate
- Goals:
  - Mutual authentication
  - Immune from replay and other attacks
  - Minimize number of messages
  - Establish a session key as a side effect

121

## Challenge/Response vs Timestamp



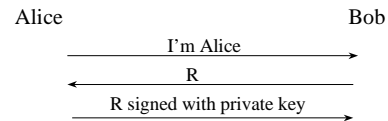
122

## Challenge/Response vs Timestamp

- Second protocol saves messages, fits more easily into existing protocols that expect passwords
- First protocol does not require synchronized clocks
- Second protocol must keep a list of unexpired timestamps to avoid replay

123

## Pitfalls with Public Key



This might trick Alice into signing something, or possibly decrypting something

124

## Eavesdropping/Server Database Stealing

- pwd-in-clear, if server stores  $h(\text{pwd})$ , protects against database stealing, but vulnerable to eavesdropping
- Standard challenge/response, using  $K=h(\text{pwd})$ , foils eavesdropping but  $K$  is pwd-equivalent so server database vulnerable
- Lamport's hash solves both

125

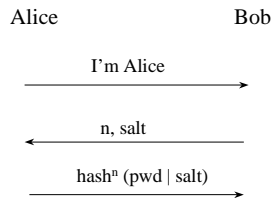
## Salt

- Protects a database of hashed passwords
- Salt is non-secret, different for each user
- Store  $h(\text{pwd}, \text{salt})$
- Users with same pwd have different hashes
- Prevents intruder from computing hash of a dictionary, and comparing against all users

126

## Lamport's Hash (S/Key)

Bob's database holds:  
n, salt,  $\text{hash}^{n+1}(\text{pwd} \mid \text{salt})$



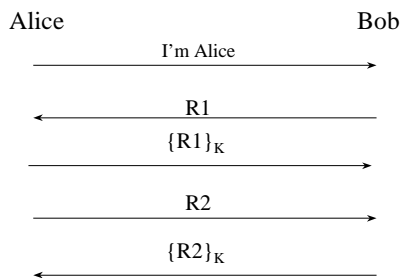
127

## Lamport's Hash (S/Key)

- Offers protection from eavesdropping and server database reading without public key cryptography
- No mutual authentication
- Only finitely many logins
- Small n attack: someone impersonates Bob

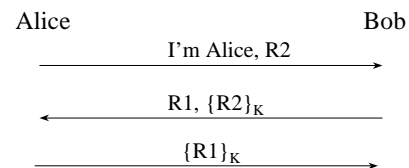
128

## Mutual Authentication



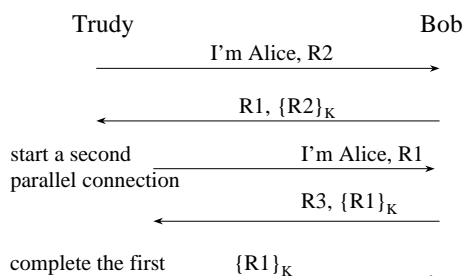
129

## More Efficient Mutual Authentication



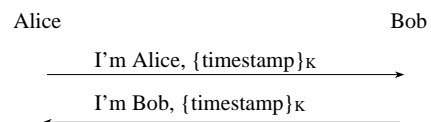
130

## Reflection Attack



131

## Timestamp Based Mutual Authentication



Two messages instead of three  
Must assure Bob's timestamp is different

132

## Random Number Pitfalls

- A surprising number of commercial and amateur systems have been broken by bad random number generators
- If you can guess the random number chosen for use as key, you have the key
  - Seed too small
  - Predictable seed (time of day)
  - Bad pseudo-random number algorithm

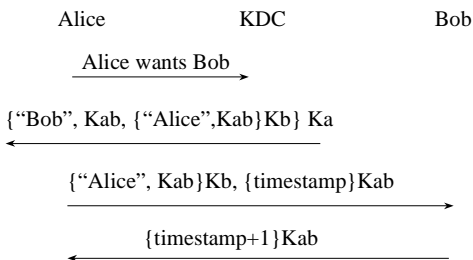
133

## Kerberos

- Developed at MIT
- Based on secret key cryptography
- Code is publicly available (until recently not legally exportable from the U.S.)
- Embedded in a variety of commercial products

134

## Kerberos Authentication (Basic)



135

## Ticket Granting Tickets

- It is dangerous for the workstation to hold Alice's secret for her entire login session
- Instead, Alice uses her password to get a short lived "ticket" to the "Ticket Granting Service" which can be used to get tickets for a limited time
- For a login session >8 hours, she must enter her password again

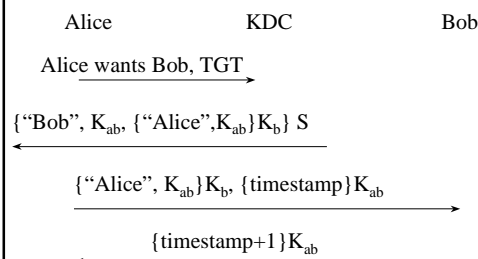
136

## Ticket Granting Tickets

- TGT looks just like ticket but encrypted with KDC's key
- WS keeps  $TGT = \{\text{"Alice"}, S\} K_{kdc}$  and  $S$

137

## Kerberos Authentication (with $TGT = \{\text{"Alice"}, S\} K_{kdc}$ )



138

## PreAuthentication

- Anyone can request a ticket on behalf of Alice, and the response will be encrypted under her password
- This allows an off-line password guessing attack
- Kerberos V5 requires an encrypted timestamp on the request
  - Only an eavesdropper can guess passwords

139

## Kerberos Cousin: OSF DCE

- Built on top of Kerberos V5
- Adds UID and group membership info into Kerberos tickets in “authorization” field
- Client learns server name. Server learns client UID and group memberships

140

## Another Kerberos Cousin: Windows 2000 Kerberos

- Also shares syntax with Kerberos V5
- Also adds UID and GID to authorization field (but incompatibly with OSF DCE)
- Implements KDC “referrals” to centralize cross-realm configuration at KDCs and hence simplifies client configuration

141

## Kerberos Cousins: Are they “really Kerberos”?

- OSF DCE Kerberos, Windows 2000 Kerberos, and MIT Kerberos all follow the spec, but they don’t interoperate
- Some people claim that the free implementation available from MIT is the one true Kerberos

142

## Delegation

- Kerberos was designed for a Unix environment, where remote login is common
- The server needs the user’s key in order to access other servers (e.g. the file system)
- Kerberos V5 creates special “forwardable” tickets for this purpose

143

## Interrealm

- Kerberos V4 supported multiple realms but no hierarchy - each pair of communicating realms share a configured key
- Kerberos V5 supports transit realms
  - Initiator must find the path
  - Path is in ticket
  - Target decides if path is acceptable
  - Spec implies they had hierarchy in mind

144



## Authorization with ACLs

- ACL lists who has access
- Easier with groups, and wildcarded names (\*@Sun.com)
- Groups might be members of groups
- Might be slow to verify if someone is a member of a deep group

145

## Authorization with Capabilities

- Suggested for OS world, never caught on
- Idea is your certificate says what you're allowed to do, not who you are
- "Capabilities are the access control mechanism for the future and always will be"

146

## Authorization Today

- Server keeps membership list of groups. ACL cannot list a group not stored on that server
- KDC keeps track of groups for each user, stores in ticket (DCE, Win2K)

147

## Authorization Tomorrow?

- On-line group membership servers
- Group name locates group membership server
- If group on ACL, either server verifies user is a member when user accesses it (and server caches info), or user requests certificate, and caches

148

## Section: Electronic Mail

- Features you might want
- Distribution Lists
- PEM (Privacy Enhanced Mail)
- PGP (Pretty Good Privacy)
- S/MIME

149

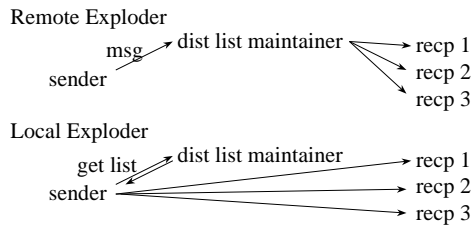
## Electronic Mail Security: What might you want?

- Privacy
- Authentication
- Integrity
- Non-repudiation

150

## Distribution Lists

- Simple multi-recipient messages
- Named distribution lists



## Remote vs Local Exploding

- Local Advantages
  - easier to prevent loops
  - sender can eliminate duplicates
  - sender can know the cost of sending
- Remote Advantages
  - secret membership lists
  - one copy over expensive link
  - parallelism exploited

## Store and Forward Mail

- May not go directly from source machine to destination mailbox
  - Path from source to destination may be intermittent
  - Security Firewall may prevent direct connections
  - Incompatible protocols may require translation

## User Keys

- Public keys most popular, though secret keys possible with KDC
- For scalability, need multiple intermediaries
- Finding chains a challenge
  - PEM and PGP take polar opposite approaches
- Revocation a hard problem (mostly unaddressed)

## End-to-end Privacy

- End-to-end means the mail delivery can't read or alter the message (it can fail to deliver it)
- All systems use public and secret keys
- {msg} secret key encrypted using random S
- {S} public key encrypted using the public key of each recipient

## Privacy with Dist. Exploders

- Exploder trusted to see msg: can add additional recipients
- Exploder has key  $K_r$
- It receives {msg}S, {S} $_{K_r}$
- Need not decrypt {msg}, though it could
- Must decrypt {S} and reencrypt under each recipient's key

## Source Authentication/Integrity

- Public key's are nice! Sender's digital signature works with all recipients.
- Exploder need not modify message.
- Non-enhanced mail recipients can ignore signature

157

## Non-Repudiation vs Plausible Deniability

- Non-Repudiation: ability to prove to 3rd party the message came from sender
- Plausible deniability protects sender. The receiver knows who sent it but can't prove it to a 3rd party
- Non-Repudiation easy with public key
- Plausible deniability easy with secret key
- Can do vice versa, but difficult

158

## Text Formatting Annoyance

- "Clever" mail gateways muck with message. Turn tabs into space, change end of line character, remove "white space", damage "parity" bit. Havoc with decryption and signatures!
- Solution: mail program must encode data 6 bits to a byte to avoid "dangerous" chars
- examples: uuencode, base64, ascii armor

159

## Mail Archives

- May want to prove mail was valid when received. (e.g., PO, but user has since declared private key compromised)
- A timestamp in the msg can be forged by the person who stole the key
- Even CA key could be compromised
- Solution: notary signs and dates msg, certs, CRLs. Must keep all those

160

## PEM

- Encrypted, MIC-ONLY, and MIC-CLEAR
- MIC-CLEAR not encoded. Readable with unmodified mailer, but signatures might not work if helpful mailer mucked with msg
- Format is vaguely human-readable labels, followed by encoded cybercrud
- Uses RSA and DES

161

## Sample (MIC-CLEAR)

To: Bob  
Subject: I'll be away tomorrow  
Date: Wed, 01 Apr 98 10:12:37 -0400  
From: Alice

-----BEGIN PRIVACY-ENHANCED MESSAGE-----

Proc-Type: 4, MIC-CLEAR  
Content-Domain: RFC822  
Originator-ID-Asymmetric: MEMxCzAJBgNVBAYTA1VTMSYw  
JAYDVQQKEsIEaWdpdGFsIEVxdW1wbWVudCBDb3Jwb3JhdGlvdj  
EMMAoGaURCcMDTEiH, 02  
MIC-Info: RS-MD5, RSA, u1OHP1RwLqePAoaN5nPk9W7Q2EjfaP+  
yDBSaLyMcSgcMK2YicGSAqLz4701+TUR4YmMd/JnHMTs  
CJerV2QFtFQ==

I'll be at a meeting in Cambridge all day

-----END PRIVACY-ENHANCED MESSAGE-----

162

## PEM Certificate Hierarchy

- Very rigid, defined in RFC 1422
- Three types of CA policy: High Assurance, Discretionary Assurance, No assurance
- All CA's in chain must follow same policy
- The rigid hierarchy is what killed PEM

163

## S/MIME

- MIME: IETF mail standard, with various body types (e.g., video, audio, text)
- S/MIME (Secure MIME): Add body types "encrypted", and "signed". Seems to be predominant standard today. Original used RSA/RC2. IETF added unencumbered suites mandatory, but commercial products use RSA and RC2 or 3DES

164

## Pretty Good Privacy

- Originally designed by Phil Zimmermann
- Became the focus of US export control debate
- Uses RSA and IDEA
- 6 message types (encoding optional): encrypted, signed, encrypted+signed

165

## PGP Certificate hierarchy

- Anyone can sign a cert for anyone
- Many geek events host PGP key signing "parties"
- Have a database with a huge wad of certs
- End users decide which to trust
- Certs distributed informally (email) and/or posted on key servers

166

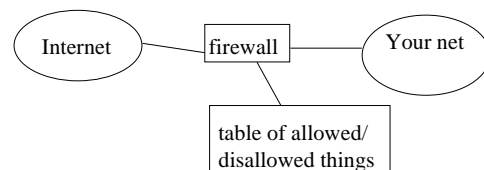
## Firewalls

- Paranoid (i.e. sensible) conn. to Internet
- Sits between your net and Internet and protects you, somehow
- Packet filter (limited access to your net to outsiders)
- Application gateway (also outsiders)
- Encrypted tunnel: full access to "insiders"

167

## Packet Filter Firewall

Allows some packets and discards others based on addresses, protocols, and ports. Transparent to allowed applications.



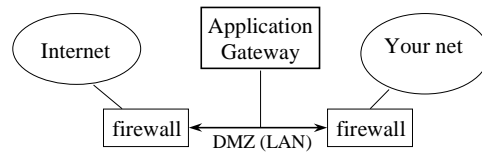
168

## Application Gateway

- Application-aware
- Keeps state as to entire transaction
- Usually you need rights to login/transfer files through firewall
- Usually you have to cryptographically authenticate to firewall
- Can do things like web caching too

169

## Application Gateway in a DMZ (DeMilitarized Zone)



170

## Which is better?

- Packet filter transparent to applications, and therefore easier to configure and use. By definition, will work with any application.
- Application gateway knows more about what is going on and can therefore make decisions about what is allowed with finer granularity

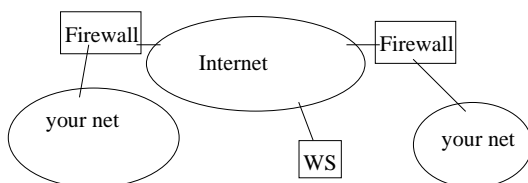
171

## Encrypted Tunnels/VPN

- Untrusted net used as a link between trusted portions of your network
- Can be private net or WS “dialing in”
- Also called “Virtual Private Net” (VPN). Requires no changes to nodes in your net
- WS connectivity requires special software on WS, but transparent to servers

172

## Encrypted Tunnel



173

## Why Firewalls Don't Really Work

- Assume all bad guys are on the “outside”
- Can be defeated by malicious software “on the inside” (perhaps delivered by virus)
- Clumsy granularity prevents many legitimate applications
- Protocols take on unexpected uses to get through (system management through HTTP, file dist through email)

174

## What's the Right Solution?

- Security enforced end to end, on end systems and servers inside the corporate net
- Firewalls should serve only as a second layer of protection (guard at the door vs lock on the filing cabinet)
- Until that happens, firewalls are the only game in town

175

## Will/Should Firewalls go away?

- Short term very necessary (we don't have end to end security everywhere)
- Forever, useful to prevent nuisance traffic and denial-of-service attacks
- Layered defense (belt and suspenders)

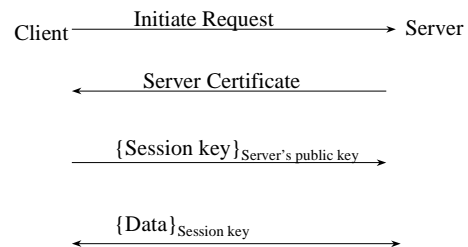
176

## SSL/TLS

- Secure Socket Layer/Transport Layer Security
- Used with http and recently others
- As deployed implements server side authentication with X.509 certs
- Client side starting to appear
- CA infrastructure issues not worked out

177

## SSL V2



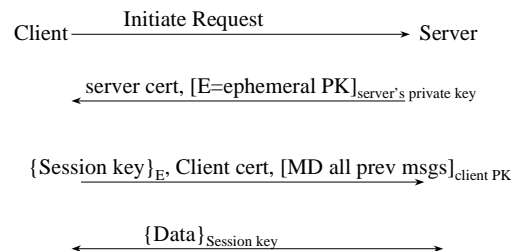
178

## Perfect Forward Secrecy

- Prevents me from decrypting a conversation even if I break into both parties after it ends
- Ex. with PFS: session key negotiated with Diffie-Hellman
- Ex. without PFS: A chooses key S, encrypts it with B's public key and sends it to B
- IESG strongly encourages PFS in protocols

179

## SSL V3 (one variant)



180

## SSL

- Commonly uses RSA and RC4
- Key sizes 1024/512 and 128/40 for US/export. 40=128 bit key with 88 in clear!
- V3 enhancements
  - ephemeral keys
  - Client authentication
  - Certificate chains, hierarchies

181

## SSL

- Ironically, ephemeral keys were to weaken the exportable version, not for PFS!
  - the ephemeral key is always short (512 bits)
  - so the domestic version doesn't use the ephemeral key when talking to a domestic client
  - Note that domestic-domestic doesn't get PFS, though TLS has that option (but nobody uses it)

182

## Server Gated Cryptography

- Some exportable servers are allowed to use strong crypto (e.g., financial institutions)
- Exportable clients are allowed to use strong crypto to these blessed servers
- Exportable clients recognize these servers because they have been given special certs from VeriSign: "step-up certificates", or "SGC (Server Gated Cryptography)"

183

## TLS

- SSL was Netscape
- Microsoft designed similar thing called PCT
- IETF, realizing bad for industry to have 2 similar things designed a third called TLS
- Based on SSL, but changed crypto suite to avoid encumbered technology
- SSL more widely deployed than TLS

184

## Naming Confusion

- X.509 certs contain X.500 names
- user types DNS name
- Now browsers enforce that "distinguished name" portion of X.500=what user typed. Kludge
- X.509 V3 allows DNS names, so hopefully ugliness of using X.500 names to carry DNS names will go away

185

## DNSSEC

- DNS is the naming system that holds names like www.microsoft.com
- Mainly used to look up addresses and mail forwarding information
- A significant vulnerability in the Internet
- IETF standard way to add digital signatures
- Also defines "Public Key" as a data type to be looked up

186

## IP Security (IPSEC)

- IETF standard for Network Layer security
- Deployed in some firewall products
- Certificate infrastructure issues still to be worked out
- Should be possible process-process, but only deployed machine-machine

187

## IP Security (IPSEC)

- Encryption and integrity protection for IP packets (layer 3)
- Pieces include:
  - AH and ESP
  - ISAKMP/IKE
  - Endless documents about specific encryption formats, etc

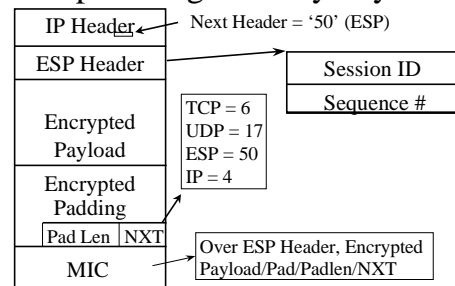
188

## AH / ESP

- extra header between layers 3 and 4 (IP and TCP) to give dest enough info to identify “security association”
- AH does integrity only - includes source and destination IP addresses
- ESP does encryption and integrity protection

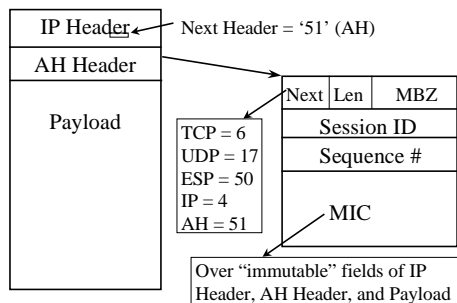
189

## ESP Encapsulating Security Payload



190

## AH (Authentication Header)



191

## ESP / AH

- Payload may be TCP, UDP, or some other ‘higher layer’ protocol (transport mode)
- Payload may be IP datagram (tunnel mode)
- Payload may be ESP/AH again (recursive encapsulation)
- If it’s important to protect IP header, ESP with tunnel mode will do that

192



## Why AH?

- AH and ESP designed by different groups. AH designers were IPv6 supporters
- AH looks more like IPv6
- AH also protects “immutable” fields in IP header.
- Originally, ESP just encryption
- Encryption without integrity has flaws

193

## Why AH, con't

- Then integrity protection added to ESP.
- Excuses for keeping AH
  - protects IP header (nobody has a credible security reason why, and ESP-tunnel can too.
  - Makes NAT harder, which pleases IPv6 fans)
  - with AH, firewalls and routers that want to look at layer 4 info (like ports) know it's not encrypted. With ESP, can't tell from packet

194

## Why Not AH?

- IPSEC way too complex.
- Layer 4 info should be hidden from routers, firewalls, and can't be integrity protected en route anyway
- You could peek inside ESP and almost always tell if it's encrypted or not. A flag might be nice (reserved SPIs would work)

195

## IPSEC Key Exchange Contenders

- Photuris: Signed Diffie Hellman, stateless cookies, optional hiding endpoint ids
- SKIP: Diffie-Hellman public keys, so if you know someone's public key  $g^B$ , you automatically know a shared secret  $g^{AB}$ . Each msg starts with per-msg key  $S$  encrypted with  $g^{AB}$
- And the winner was...

196

## ISAKMP

- Internet Security Association and Key Management Protocol
- Gift to the IETF from NSA
- A “framework”, not a protocol. Complex encodings. Flexible yet constraining.
- Two “phases”. Phase 1 expensive, establishes a session key with which to negotiate multiple phase 2 sessions

197

## IKE

- Mostly a rewriting of ISAKMP, but not self-contained. Uses ISAKMP
- Since both so complex, some inconsistencies
- Since both so badly written, hasn't gotten thorough review
- Many options

198

## IKE

- Two phases, like ISAKMP
- Phase 1 has two “modes”: aggressive, and main
- Main has 6 messages, aggressive has 3
- Main does more, sometimes, like hiding endpoint identifiers
- Phase 2 known as “quick mode”

199

## IKE Key types

- For each of main and aggressive, protocols are defined for each of the following key types:
  - pre-shared secret key
  - public signature keys
  - public encryption keys (old cruffy way)
  - public encryption keys (new improved method)

200

## Of the IKE Variants

- The one required variant (main mode, pre-shared secret keys): nobody uses it
  - ID transmitted, encrypted with key which is a function of pre-shared key! Can’t decrypt unless you can guess who you’re talking to!
  - So ID=IP Address. Useless for “road warrior”
- Instead, the most commonly used one is aggressive mode, pre-shared secret keys

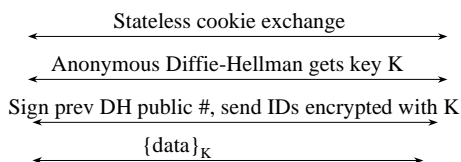
201

## IKE Problems

- Terminal complexity
- Lost ability to do stateless cookies!
- Many exchanges could be shortened without loss of functionality
- Many exchanges could have hidden endpoint identifiers, or had other features, but don’t

202

## What IKE Should Be



Note: eavesdropper can’t see who’s talking.  
Active attacker (man in middle) can see who attempted to talk (but they’ll abort)

203

## Layer 3 vs Layer 4

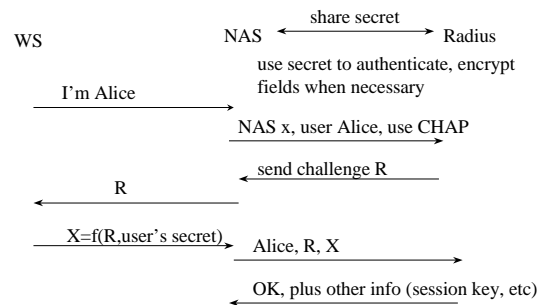
- Layer 4 philosophy: Don’t change OS. Implement over TCP
- Layer 3 philosophy: Don’t change apps
- Layer 4 has rogue packet problem: TCP accepts data since it isn’t involved in the crypto
- Layer 3 has problems with specifying user. API should change.

204

## Radius

- Idea: access intranet through NAS (network access server)
- Keep security information about users in Radius server
- Many authentication systems supported
- One example on next slide

205



206

## CHAP

- “Challenge Handshake Authentication Protocol”
- meant for PPP
- simple shared secret challenge-response

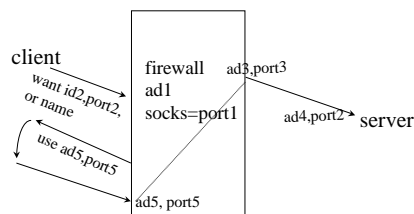
207

## SOCKS v5

- Intended for firewalls
- First negotiate how, and then authenticate to firewall
- Then say IP address or DNS name, plus TCP port, of what you want to talk to
- DNS name works if beyond a NAT box
- Traffic between client and firewall might be authenticated, encrypted, neither, both

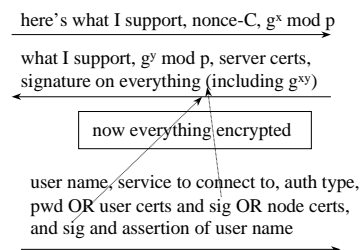
208

## Port and Address Translation, SOCKS



209

## SSH (secure shell)



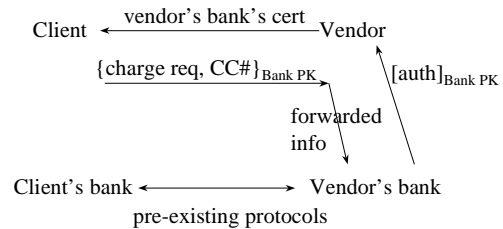
210

## Internet Commerce

- Most widely done today by sending credit card # and expiration date to vendor
- With SSL, info is protected from eavesdroppers
- Goals of better protocols:
  - hide credit card info (and client identity) from vendor
  - Use stronger crypto than US export allows

211

## SET (Secure Electronic Transactions)



212

## Lotus Notes

- RSA keys, RC2/4 for encryption
- Email, authentication of users, encryption on stored documents, encrypted databases
- User's private key kept in "ID file" on user's WS, encrypted with user's pwd. Works disconnected from net. Less convenient if user changes WSs

213

## Lotus Notes

- "North American": 64 bit RC2/RC4 keys and 630 bit RSA keys
- exportable uses 512 bit RSA keys for privacy, 630 bits for authentication and signatures. Encryption uses 64 bits but a "US govt workfactor reduction field" includes 24 bits of the key encrypted with US gov public key, so it's 40 bits to US gov

214

## Lotus Notes

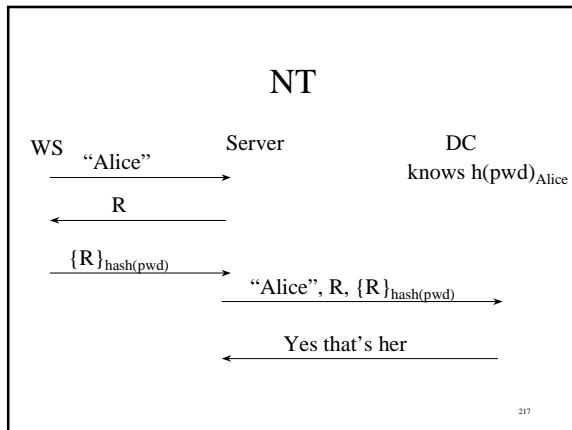
- CA Hierarchy is bottom-up. Need cross-links between organizations
- Cute protocol for authentication: Server has a single secret  $S$ . After expensive authentication with lots of public key crypto and cert chains, gives client  $X = \text{hash}(S, \text{client's name})$ . Client can remember  $X$  and bypass expensive authentication.

215

## Microsoft

- LAN Manager
  - $h(\text{pwd})$  at each server, used as session key
- NT
  - Still secret key, but security info stored in Domain Controller (similar to KDC)
  - Compatible with LAN Manager clients
  - Servers talk to DC via encrypted connection
  - cross-domain possible, but not transit domains

216



### Windows 2000

- Kerberos, plus nonstandard extensions for authorization
- Database in Active Directory contains groups per user. Group membership gets put into tickets.

218

### APIs from RSA Data Security

- BSAFE: low level access to crypto functions
- RSAREF similar, free for non-commercial use.
- PKCS-11, interface to smart cards, or "software smart card" so applications can be smart card enabled without requiring them

219

### Other APIs

- GSSAPI
  - IETF standard
  - High level interface
  - Too high level for TLS, IPSEC, or PEM
- CAPI (Microsoft), CDSA (common data security architecture, Intel et al), ICF (HP)
  - API on top for applications. On bottom for crypto-service providers.

220

### Clipper

- Allow encryption without impeding govt ability to wiretap
- Each chip has key  $K$  and Serial #  $N$ . Split  $K$  into  $K_1$  and  $K_2$  such that  $K_1 \text{ XOR } K_2 = K$ . Give one agency  $N/K_1$ , another  $N/K_2$
- LEAF field contains  $N$  and session key encrypted with  $K$
- With court order, govt can get  $K$

221

### Why would anyone buy Clipper?

- Backdoor makes it more expensive
- Other forms might become illegal
- US govt will use it, so needed for secure comm to them. Might result in de facto standard, alternate mechanisms might not gain market share
- ATT has a warehouse full of them...

222

## Big Brother Laws

- US (and others) didn't allow export of strong crypto, usually more than 40 bits
- Some countries (France, and US would love to join them) didn't allow their citizens to use crypto without govt access
- Usual workaround: good domestic crypto, 40 bit exportable, no crypto for France, etc.

223

## Crypto-With-A-Hole

- intent: prohibit adding strong crypto to an exported application
- Reality: vague and impossible
- Result: another vague club that can be used selectively against uncooperative software vendors
- Irony: US govt want to be able to buy commercial stuff and add their own crypto

224

## Crypto-With-A-Hole

- Companies scramble to provide algorithm independent crypto
  - Microsoft CAPI, HP ICF, Intel CDSA
- Idea: digitally sign plug-in crypto modules that agree to abide by US export rules
- Having lots of algorithms makes interoperability difficult

225

## Escrow for Careless Users

- Recover encryption key, and signature key
- Easy to do. Encrypt user's key with escrow agent's PK, and store redundantly (for paranoia, not with escrow agent)
- Can do k out of n scheme, too.

226

## Key Recovery: Law vs Business

- |   |   |
|---|---|
| • stored and transmitted                      | • stored data only                        |
| • enc key only. Bad for them to know sig key! | • maybe both encryption and signature key |
| • "noncircumventable"                         | • simple, cheap, fast                     |
| • all-but-40 makes sense                      | • want full key                           |
| • Without owner's knowledge or coop           | • There are some Big Brother companies    |
| • k of n agents useful                        | • k of n agents useful                    |

227

## Conclusions

- The technology for cryptographically securing the Internet is well understood
- Products are beginning to be deployed
- "This stuff isn't that hard. How come nobody gets it right?"... Al Eldridge

228