Principles of Network Security Protocols

Radia Perlman
radia.perlman@sun.com

Charlie Kaufman
charliek@microsoft.com

1

What is this tutorial about?

- What network protocol designers need to know about security protocols
- Putting some "hot buzzwords" in perspective
- Going beyond details of standards and math to what "really matters"

2

First, what do we mean by "security"?

3

First, what do we mean by "security"?

- Limit data disclosure just to intended recipients?

4

First, what do we mean by "security"?

- Limit data disclosure just to intended recipients?
  - Monitor traffic to find terrorists?
  - Cut down on spam and malware?

5

First, what do we mean by "security"?

- Authenticate the source?

6

First, what do we mean by "security"?

- Authenticate the source?
  - allow anonymity for whistleblowers, spies?

7

First, what do we mean by "security"?

- DRM?

8

What are the threats?

- Environment-dependent
- Read data?
- Modify data?
- Generate data?
- Infect machines?

9

Some basic terms

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

10

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)

11

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

12

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 to 3rd party that I placed the order

13

Cryptography

- Crypto
  - secret key
  - public key
  - cryptographic hashes
- Used for
  - authentication, integrity protection, encryption

14

What you really need to know

- The math (of some algorithms) is cool, for its own sake
- But you need to know the functionality

15

What you really need to know

- The math (of some algorithms) is cool, for its own sake
- But you need to know the functionality
  - e.g., implications of basing integrity checks, or authentication, on public vs secret keys

16

What you really need to know

- The math (of some algorithms) is cool, for its own sake
- But you need to know the functionality
  - e.g., implications of basing integrity checks, or authentication, on public vs secret keys
- And the "hard stuff" not in the standards

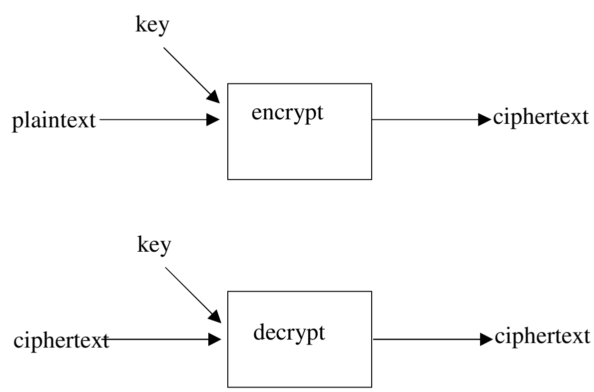
17

Secret Key/Symmetric Crypto

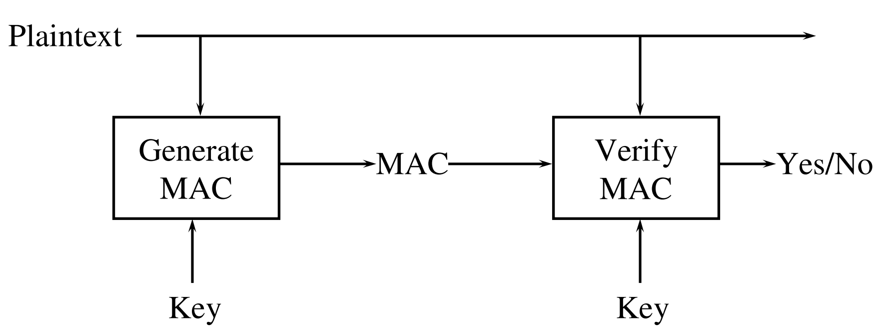
- Two operations ("encrypt", "decrypt") which are inverses of each other. Like multiplication/division
- One parameter ("the key")
- Even the person who designed the algorithm can't break it without the key (unless they diabolically designed it with a trap door)
- Ideally, a different key for each pair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

18

Secret key encryption



Secret Key Integrity Protection



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 →

Secret key crypto, Alice and Bob share secret S

- encrypt=f(S, plaintext)=ciphertext
- decrypt=f(S, ciphertext)=plaintext
- authentication: send f(S, challenge)
- integrity check: f(S, msg)=X
- verify integrity check: f(S, X, 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”
- Even 64 bits is daunting to search through
- Faster computers work to the benefit of the good guys!

23

Token Cards

- should be 2-factor (card+PIN)
- challenge/response (need keyboard)
- time-based
  - time skew (can adjust time and rate each time)
  - if no keyboard, PIN can be sent with value
- sequence based
  - issue if mess up several times
  - same PIN issues if no keyboard

24

Public Key/Asymmetric 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?

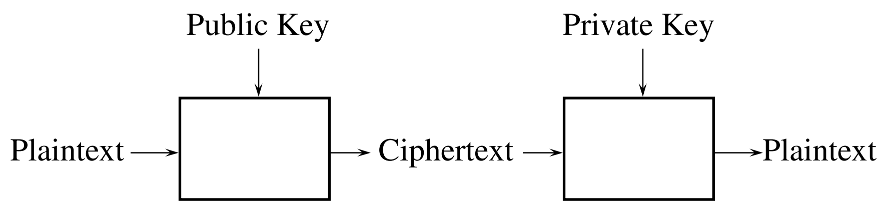
25

Digital Signatures

- One of the best features of public key
- An integrity check
  - calculated as f(priv key, data)
  - verified as f(public key, data, 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

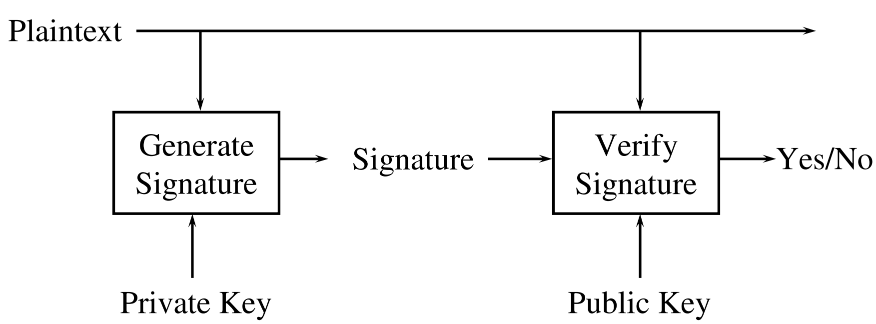
26

Public Key Encryption for Privacy



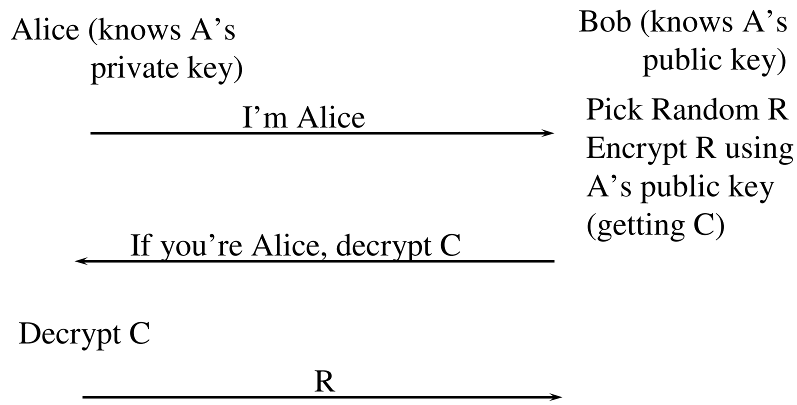
27

Public Key Integrity Protection



28

Public Key Authentication



29

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!

30

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

31

Uses of hashes

- Public key signature of msg is actually signature on h(msg)
- Shorthand for a public key when doing configuration
- Keyed hash: h(msg, key) = integrity check on msg

32

Aren’t hashes “broken”?

- Terminology for 3 attacks (in order of difficulty, from easiest to hardest)
  - collision: finding two things with same hash
  - 2nd preimage: finding something with the same hash as a given thing
  - 1st preimage: given a hash value, finding something with that hash

33

What’s been done

- Collisions with all of the older standards
- 2nd preimage with some
- 1st preimage with MD4
- Collisions should take $2^{n/2}$, and preimages should be 2^n: hashes considered “weak” if attacks in fewer than that (e.g., SHA-1 in 2^{51})
- Note: these attacks (mostly) don’t affect “keyed hash”, certainly not in practice (today)

34

Current hashes

- SHA-256, SHA-384, and SHA-512 still OK, but “suspect”
- NIST is running a competition
- Real lesson: crypto-agility

35

Real vulnerabilities tend to be more mundane

36

Real vulnerabilities tend to be more mundane

- C language: null terminated strings
- Other languages: counted strings

37

Real vulnerabilities tend to be more mundane

- C language: null terminated strings
- Other languages: counted strings

- CA issues a cert to badguys.com for any name that ends in .badguys.com
- Browser alerts if cert does not not match the URL (say bigbank.com)

38

Real vulnerabilities tend to be more mundane

- C language: null terminated strings
- Other languages: counted strings
- CA issues a cert to badguys.com for any name that ends in .badguys.com
- Browser alerts if cert does not not match the URL (say bigbank.com)
- certificate for bigbank.com\0.badguys.com

39

Popular Secret Key Algorithms

- DES (old standard, 56-bit key, slow)
- 3DES: fix key size but 3 times as slow
- RC4: variable length key, “stream cipher” (generate stream from key, XOR with data)
- AES: NIST-driven standard

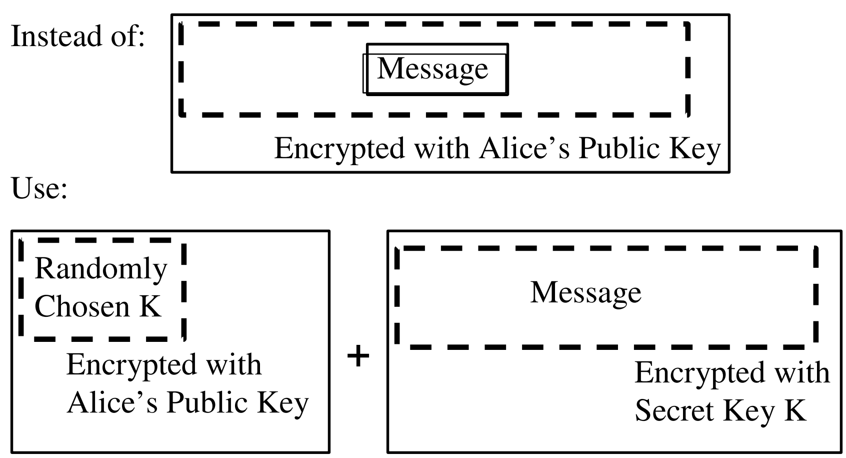
40

Popular Public Key Algorithms

- RSA: nice feature: public key operations can be made very fast, but private key operations will be slow. Patent expired.
- ECC (elliptic curve crypto): smaller keys, so faster than RSA (but not for public key ops). Some worried about patents

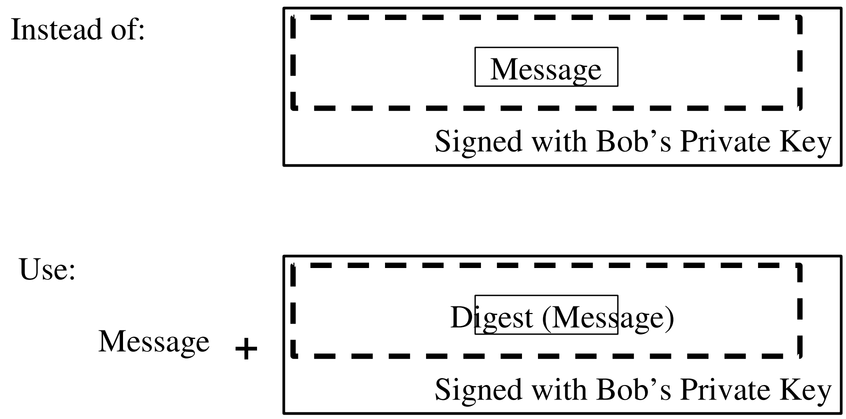
41

Hybrid Encryption



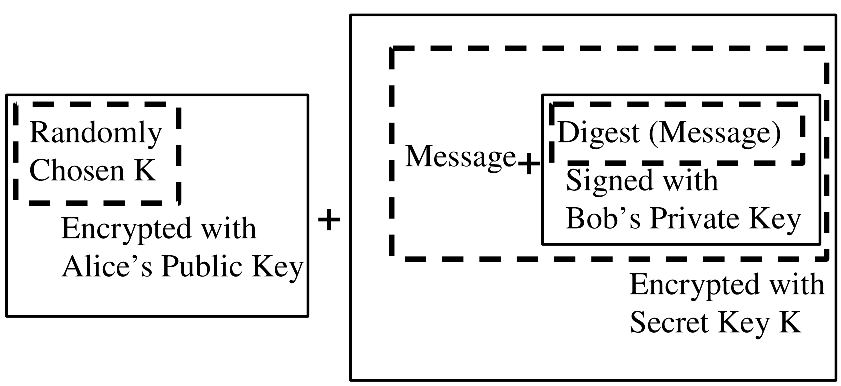
42

Hybrid Signatures



43

Signed and Encrypted Message



44

Our Notation

Encryption: Curly brackets followed by name of key

{data}K_{AB}

Signed info: Square brackets followed by name of key

[information]Alice

45

An elegant public key algorithm

46

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

47

Why is D-H Secure?

- We assume the following is hard:
- Given g, p, and g^X mod 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

48

Diffie-Hellman

Alice agree on g,p Bob

choose random A choose random B

g^A mod p

g^B mod p

compute $(g^B \bmod p)^A$ compute $(g^A \bmod p)^B$

agree on g^{AB} mod p

49

Man in the Middle

Alice Trudy Bob

g^A mod p g^T mod p

g^T mod p g^B mod p

agree on g^{AT} mod p agree on g^{TB} mod p

{data}g^{AT} mod p {data}g^{TB} mod p

{data}g^{AT} mod p {data}g^{TB} mod p

50

Signed Diffie-Hellman (Avoiding Man in the Middle)

Alice | Bob

choose random A | choose random B

[g^A mod p] signed with Alice's Private Key →

← [g^B mod p] signed with Bob's Private Key

verify Bob's signature | verify Alice's signature

agree on g^{AB} mod p

If you have keys, why do D-H?

- "Perfect Forward Secrecy" (PFS)
- Prevents me from decrypting a conversation even if I break into both parties after it ends (or if private key is escrowed)

Example non-PFS (like SSL)

Alice → Bob: {K}Bob

Alice ↔ Bob: protect conversation using K

53

PFS without Diffie-Hellman

Bob: invent new RSA pair for this conversation

Bob → Alice: [Use public key P]Bob

Alice → Bob: {K}P

Alice ↔ Bob: protect conversation using K

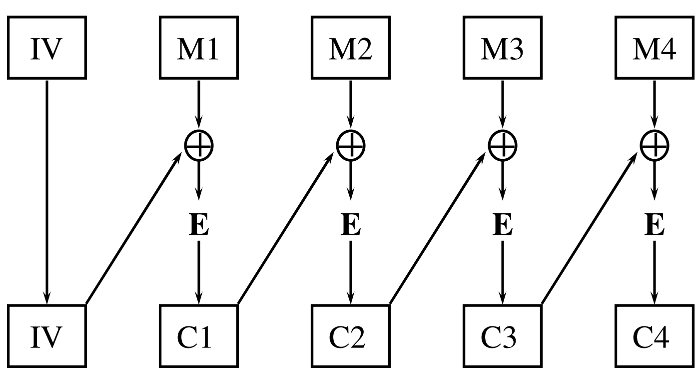
54

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!

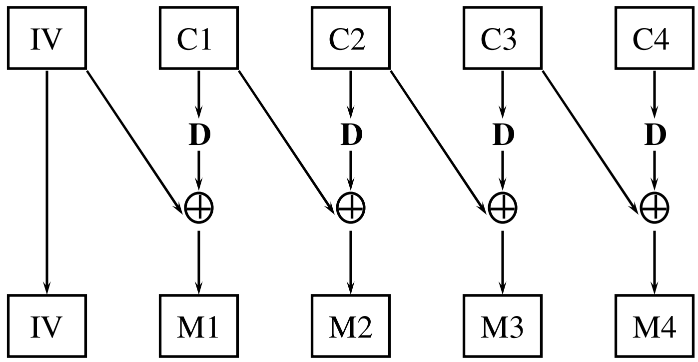
55

CBC (Cipher Block Chaining)



56

CBC Decryption



57

New topic: Key Distribution

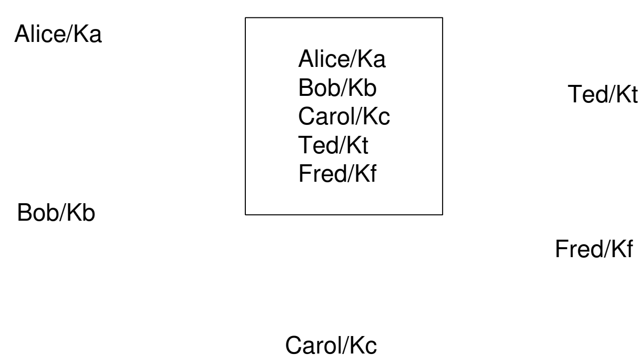
58

Key Distribution - Secret Keys

- 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
- This is basically Kerberos

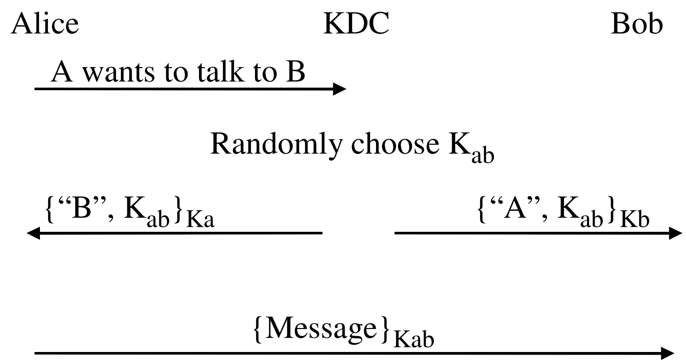
59

KDC



60

Key Distribution - Secret Keys



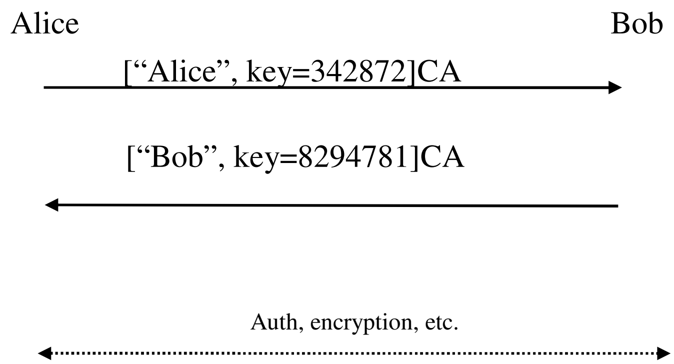
61

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

62

Key Distribution - Public Keys



63

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

64

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

65

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

66

Revocation

- What if someone steals your credit card?
  - depend on expiration date?
  - publish book of bad credit cards (like CRL mechanism …cert revocation list)
  - have on-line trusted server (like OCSP … online certificate status protocol)

67

Another approach: Identity Providers

- Evolved from “Passport”
- For authenticating users to web sites
- Instead of zillions of username/pwd at each site, authenticate to one: your “identity provider”
- With the magic of cookies and URL rewriting, it vouches for you at affiliated sites

68

Pluses and Minuses

- Plus:
  - Works with “existing” browsers
- Minus:
  - identity provider vs peer-to-peer authentication
    - availability
    - performance
    - security
    - privacy
  - What if zillions of identity providers?

69

New topic: Key hierarchies

70

Strategies for Hierarchies

- Monopoly
- Oligarchy
- Anarchy
- Bottom-up

71

Monopoly

- 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

72

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 that one organization, forever

73

Oligarchy of CAs

- Come configured with 80 or so trusted CA public keys (in form of "self-signed" certificates!)
- Usually, can add or delete from that set
- Eliminates monopoly pricing

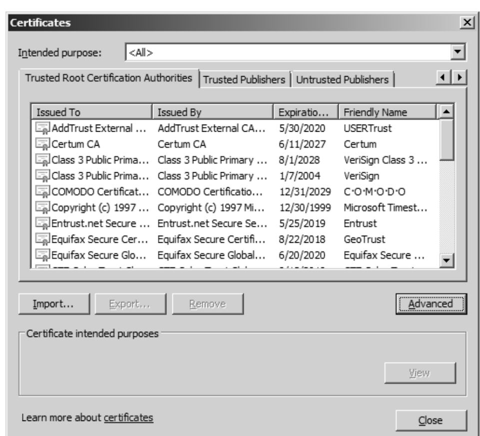
74

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)
  - impractical for anyone to check trust anchors
- Although not monopoly, still favor certain organizations. Why should these be trusted?

75

Default Windows Oligarchy



76

Windows Idiosyncrasies

- The list you see is a subset of the real list
- New CAs are downloaded from WindowsUpdate “on demand”
- You can turn this feature off
- Some companies manage the trusted list for their enrolled systems including adding their own CAs

77

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

78

Important idea

- CA trust shouldn't be binary: "is this CA trusted?"
- Instead, a CA should only be trusted for certain certificates
- Name-based seems to make sense (e.g., trusted for *.citibank.com)

79

Top Down with Name-based policies

- Assumes hierarchical names
- Each CA only trusted for the part of the namespace rooted at its name
- Easy to find appropriate chain
- This is a sensible policy that users don't have to think about
- But: Still monopoly or oligopoly at top, since every chain starts somewhere

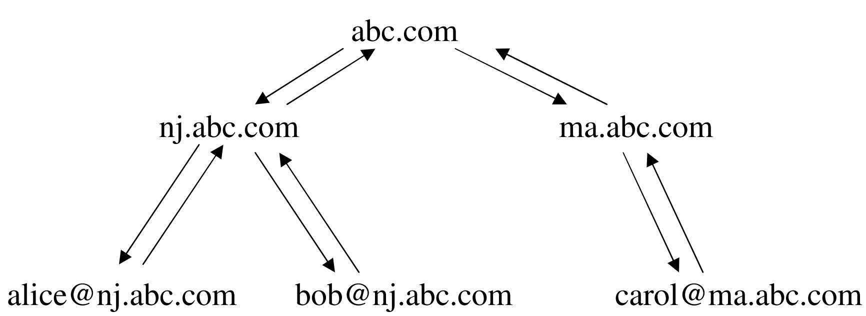
80

Bottom-Up Model

- Each arc in name tree has parent certificate (up) and child certificate (down)
- Name space has CA for each node
- Cross Links to connect Intranets, or to increase security
- Start with your public key, navigate up, cross, and down
- When a key changes, only the nodes that certify it have to do anything

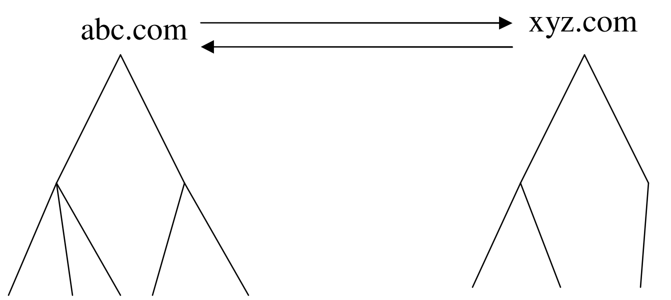
81

Company designs its PKI trusting no one

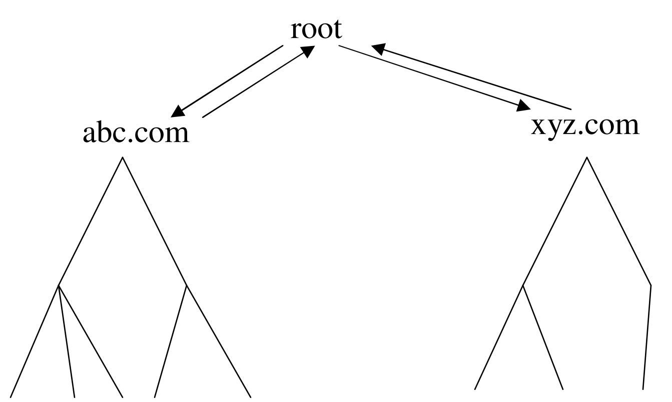


82

Cross-Certify Partner Companies



Root Service can simplify things



Notes

- You don't have to start at your own key, e.g., federal PKI with "bridge" CA
- Cross certification can bypass hierarchy, for trust issues
- Can have multiple competing CAs at various points in the hierarchy
  - If too many, might negotiate "which roots do I trust" with other side

85

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

86

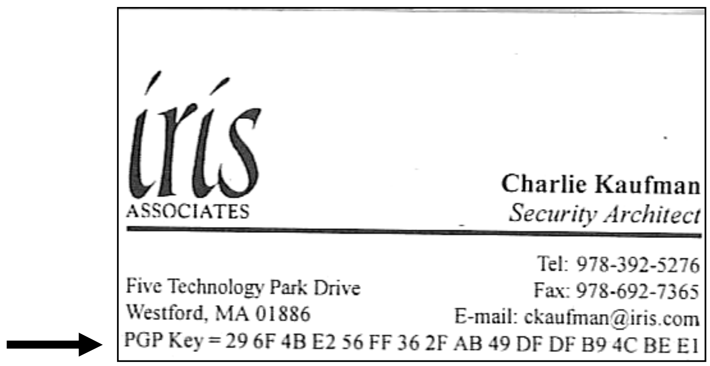
Public Keys without a PKI

87

How do we deal with people?

88

Some of us tried…



89

eMail

- I could include my public key with every message (and sign the message)
- Your mailer reports it to you only if someone's public key ever changes
- I encrypt messages if I know the recipient's public key

90

Web Accounts

- When I set up an account on a web site, instead of supplying a username/password I provide a public key
- Future visits use the public key to identify and authenticate me

91

Revocation

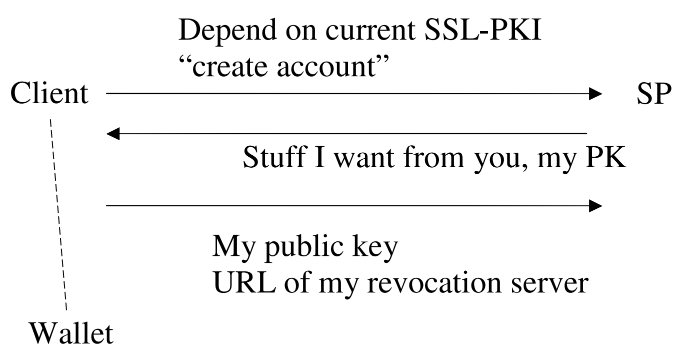
- What if you lose your key, or it is stolen?
- Now hundreds of merchants have stored your old key

92

Revocation

- What if you lose your key, or it is stolen?
- Now hundreds of merchants have stored your old key
- Idea: Revocation service

Enrolling



{addresses}
{credit cards}
{telephone numbers}
Passport number
Per site info (its public key, your key pair for that site)

Revocation service

- SP learns user’s revocation server along with the user’s public key
- SP can “enroll” with that revocation service, to be notified in case of revocation
- Or SP can check periodically
- User has to have some sort of out-of-band mechanism to authenticate to revocation service and revoke the key
- Perhaps allow revocation service to inform of the new key as well, though that makes revocation service more trusted

95

Other potential issues

- Authenticated attributes (e.g., qualified for AAA discount, age > 21, member of group allowed to do free IEEE library searches...)
- Web of trust between organizations
- Neither of those are harder with IDPs than with certificates

96

So, two examples of easy PKI

- Within an enterprise: one CA
- On the web: “key continuity”; don’t need certificates or CAs at all

97

Cryptographic Handshakes

98

Cryptographic Handshakes

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

99

Challenge/Response vs Timestamp

Alice → Bob

compare:

Alice → Bob: I'm Alice

Bob → Alice: R

Alice → Bob: $\{R\}_K$

---

vs:

Alice → Bob: I'm Alice, $\{\text{timestamp}\}_K$

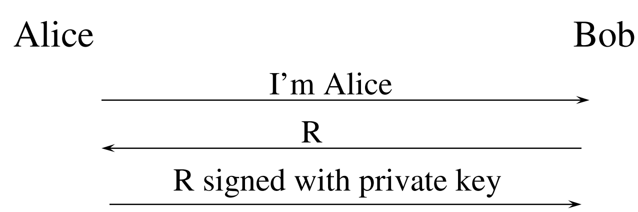
100

Challenge/Response vs Timestamp

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

101

Pitfalls with Public Key



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

102

Eavesdropping/Server Database Stealing

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

103

Salt

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

104

Lamport’s Hash (S/Key)

Bob’s database holds:
n, salt, $hash^{n+1}$(pwd | salt)

Alice → Bob: I’m Alice

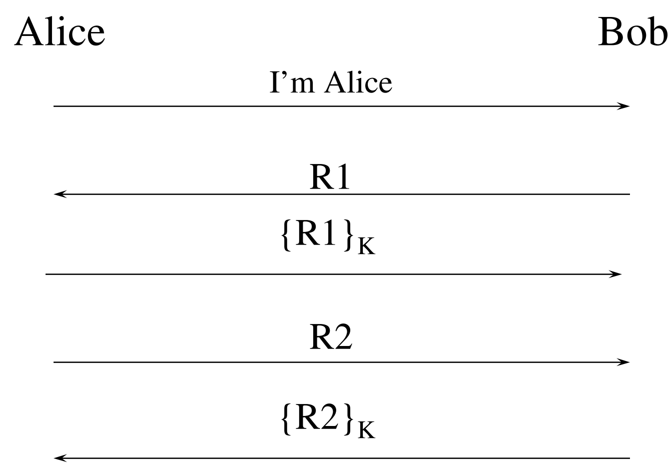
Bob → Alice: n, salt

Alice → Bob: $hash^{n}$ (pwd | salt)

Lamport’s Hash (S/Key)

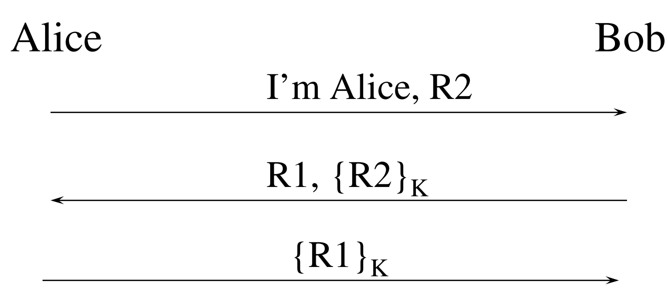
- Offers protection from eavesdropping and server database reading without public key cryptography
- No mutual authentication
- Only finitely many logins
- Small n attack: someone impersonates Bob

Mutual Authentication



107

More Efficient Mutual Authentication



108

Reflection Attack

Trudy Bob

I'm Alice, R2

R1, $\{R2\}_K$

start a second parallel connection

I'm Alice, R1

R3, $\{R1\}_K$

complete the first

$\{R1\}_K$

109

Timestamp Based Mutual Authentication

Alice Bob

I'm Alice, $\{timestamp\}_K$

I'm Bob, $\{timestamp\}_K$

Two messages instead of three

Must assure Bob's timestamp is different

110

Crypto negotiation

- These days protocols need to be "crypto-agile" (negotiate which algorithms to use for encryption, hash, etc.)
- Usually one side says "here is a list, in preference order, of what I can do"
- Other side chooses

111

Security issue

- By definition, negotiation messages can't be integrity-protected
- So an attacker can remove the secure choices; force them to talk with insecure algorithms

112

Handshake techniques

- Extended sequence numbers
- Use Diffie-Hellman, or some other technique, to ensure unique key per session

113

Some specific protocols

114

SSL/TLS

Client — Initiate Request → Server

Server Certificate

$\{\text{Session key}\}_{\text{Server's public key}}$

$\{\text{Data}\}_{\text{Session key}}$

115

Exportable Crypto

Client — Initiate Request → Server

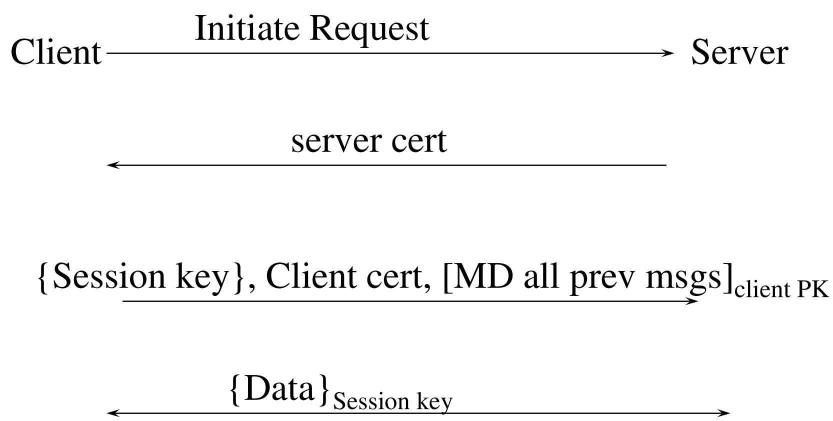
$\text{server cert, [E=ephemeral PK]}_{\text{server's private key}}$

$\{\text{Session key}\}_{E}$

$\{\text{Data}\}_{\text{Session key}}$

116

Client Auth



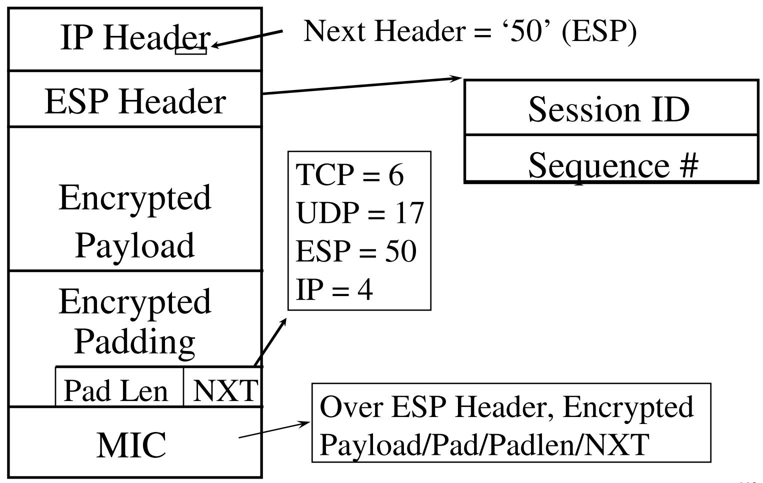
117

IPsec

- Packet format for data (AH/ESP)
- Authentication handshake/establish key (IKE)

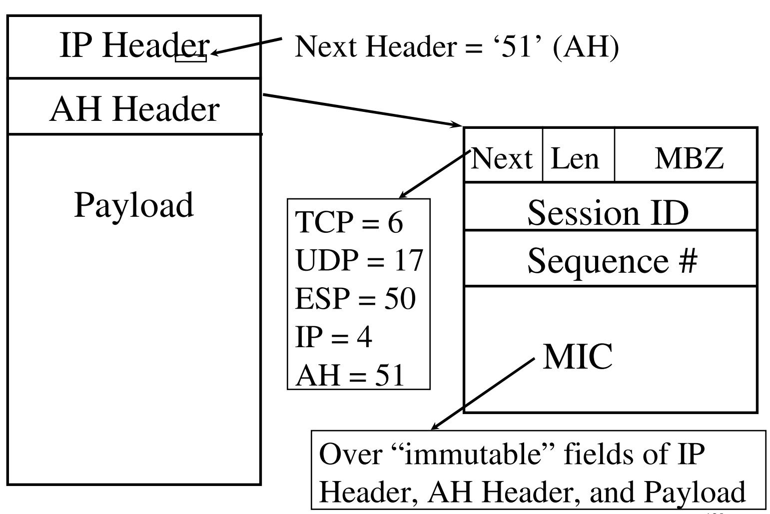
118

ESP
Encapsulating Security Payload



119

AH (Authentication Header)



120

General idea of IKEv2

Alice Bob

g^A mod p, $nonce_A$

g^B mod p, $nonce_B$

{“Alice”, proof I’m Alice}g^{AB} mod p

{“Bob”, proof I’m Bob}g^{AB} mod p

121

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

122

New topic: Electronic Mail

123

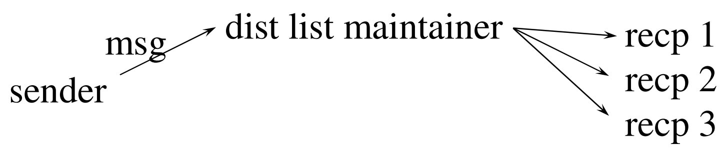
Electronic Mail Security: What might you want?

- Privacy
- Authentication
- Integrity
- Non-repudiation
- Spam/malware defense

124

Distribution Lists

- Simple multi-recipient messages
- Named distribution lists



125

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

126

Privacy with Dist. Exploders

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

127

Source Authentication/Integrity

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

128

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

129

Data at Rest issues

- Granularity? Whole disk? Files? Records?
- Layer of encryption: OS? Application?
- Encryption modes so can read/write chunks
- Room to put integrity check/IV for each chunk
- Ability to prevent overwriting chunk with old chunk
- Key recovery
- Assured delete

130

Assured Delete

Assured Delete

- If you make copies, too hard to ensure all copies are gone
- So we'll invent an "ephemerizer" with time-based public keys
- Add that lock to already-encrypted data
- But have to minimize overhead

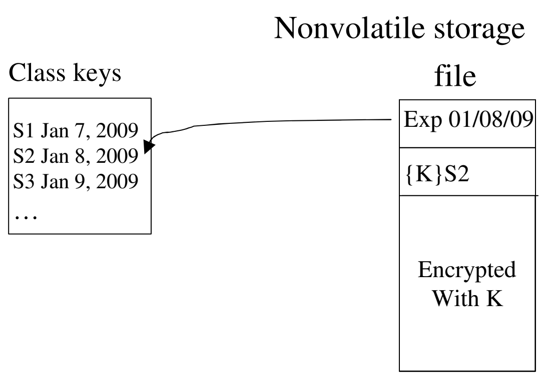
What if ephemerizer faulty?

- Rather than relying on it to be super-robust, use multiple flaky ephemerizers
- If worried about it not forgetting keys, use k out of n scheme

133

File system with Master class keys

Class keys: Secret keys
Generated by file system

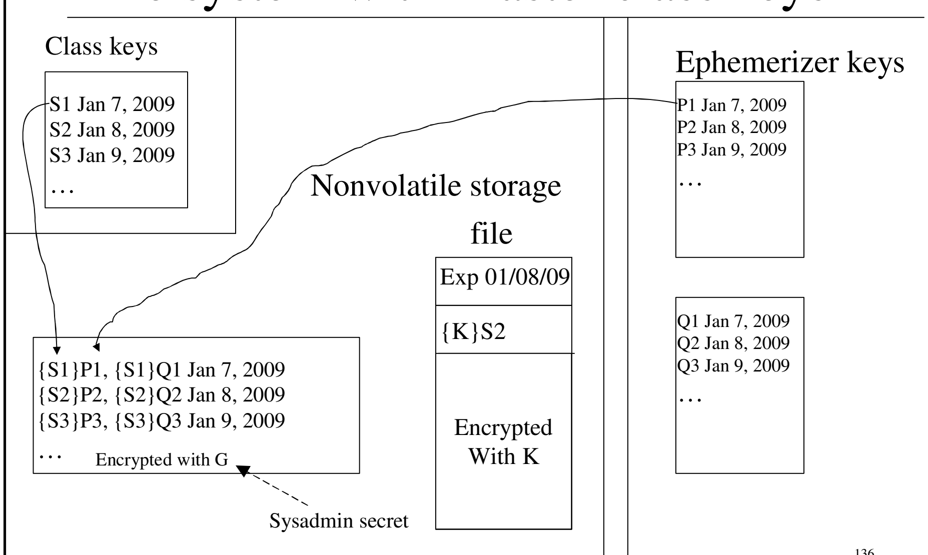


134

Use ephemerizer to back up class keys

135

File system with Master class keys



136

Protocol for requesting decryption from Ephemerizer

137

What we want to accomplish

Bob — Ephemerizer

Has $\{M\}P_i$

Please decrypt {M}Pi with key ID i →

← M

138

What we want to accomplish

Bob

Has $\{M\}P_i$

Ephemerizer

Please decrypt {M}Pi with key ID i →

← M

But we don't want the Ephemerizer to see M

139

We'll create "blind decryption"

- Same basic idea as blind signatures
- Bob wants Ephemerizer to decrypt {M}Pi with its private key #i
- … Without seeing what it is decrypting
- Bob encrypts (blinds) with another function, which commutes with Ephemerizer's crypto
- Then unblinds

140

Blind Decryption

Bob

Has $\{M\}P_i$

Invents (B,U)

Ephemerizer

Please decrypt { {M}Pi } B with key ID i →

← {M}B

File system applies U to get M

Ephemerizer only sees {M}B

141

Non-math fans can take a nap

142

For you math fans…

143

Blind Decryption with RSA, BD's RSA PK=(e,n), msg=M

Alice — BD

wants to decrypt M^e mod n
chooses R, computes R^e

$M^e\ R^e$ →

applies (d,n)
$M^{ed}R^{ed}$

← M R

divides by R to get plaintext M

144

Blind encryption with D-H public key g^x mod p

- BD has Diffie-Hellman key (g^x mod p)
- Alice encrypts M with BD's public key:
  - Alice chooses y, raises g and BD's PK to y
    - (g^y) and $(g^x)^y$
  - Encrypts M with g^{xy} mod p: $\{M\}g^{xy}$
  - Saves $\{M\}g^{xy}$ and (g^y)
  - Discards y and g^{xy}

145

Blind Decryption: BD's PK=(g^x mod p)

Alice — BD

knows: g^y mod p, $\{M\}g^{xy}$ mod p
chooses z, z^{-1}, computes $(g^y)^z$

g^{yz} mod p → (Alice to BD)

g^{xyz} mod p (BD applies priv x) ← (BD to Alice)

$(g^{xyz} \text{ mod } p)^{z^{-1}}$
decrypts $\{M\}g^{xy}$ mod p with g^{xy} mod p

146

Properties of our protocol

- Ephemerizer gains no knowledge when it is asked to do a decryption
- Protocol is really efficient: one IP packet request, one IP packet response
- No need to authenticate either side
- Decryption can even be done anonymously

147

OK, non-math fans can wake up now

148

Quick things if time

- IBE
- quantum
- DRM
- Why does software suck?
- TPM

149

Plug for our book

- Kaufman, Perlman, Speciner, “Network Security: Private Communication in a Public World”

150

SI SPY NET WORK, BIG FEDJAW IOG LINK KYXOGY

151

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

152