An Introduction to Cryptography

Steven M. Bellovin
smb@research.att.com
http://www.research.att.com/~smb
AT&T Labs Research

Outline

- What is Cryptography?
- Cryptographic Primitives
- Cryptographic Combinations and Protocols
- Cryptography and the Internet
- Threats
- References

What is Cryptography?

What is a Cryptosystem?

- $K = \{0,1\}^l$
- $P = \{0,1\}^m$
- $C' = \{0,1\}^n, C \subseteq C'$
- $E : P \times K \rightarrow C$
- $D : C \times K \rightarrow P$
- $\forall p \in P, k \in K : D(E(p,k),k) = p$
- It is infeasible to find $F : P \times C \rightarrow K$

Let's start again, in English…

What is a Cryptosystem?

A cryptosystem is pair of algorithms that take a *key* and convert *plaintext* to *ciphertext* and back.

Plaintext is what you want to protect; ciphertext should appear to be random gibberish.

The design and analysis of today's cryptographic algorithms is highly mathematical. Do *not* try to design your own algorithms.

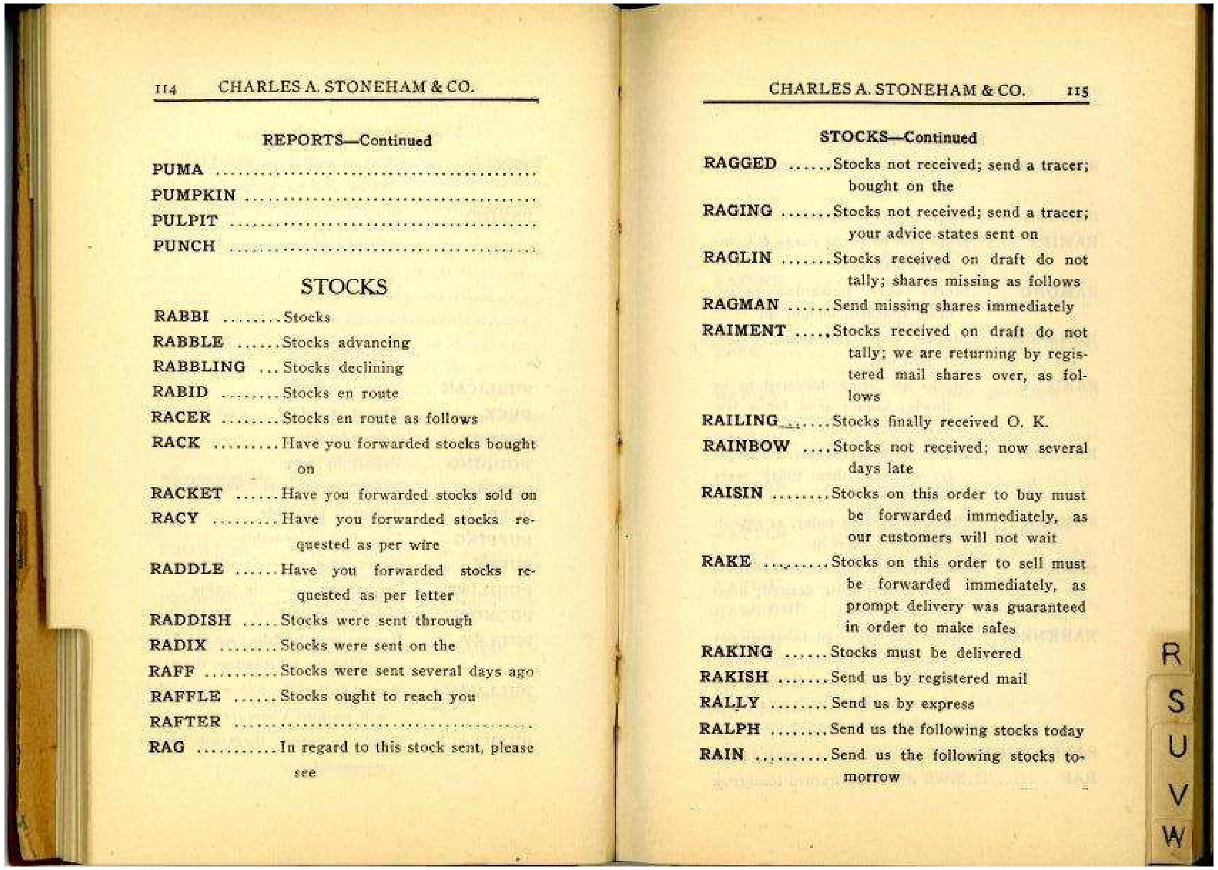
A Tiny Bit of History

- Encryption goes back thousands of years
- Classical ciphers encrypted letters (and perhaps digits), and yielded all sorts of bizarre outputs.
- The advent of military telegraphy led to ciphers that produced only letters.

Codes vs. Ciphers

- Ciphers operate *syntactically*, on letters or groups of letters: $A \rightarrow D$, $B \rightarrow E$, etc.
- Codes operate semantically, on words, phrases, or sentences, per this 1910 codebook

A 1910 Commercial Codebook



114 CHARLES A. STONEHAM & CO.

REPORTS—Continued

PUMA ....
PUMPKIN ....
PULPIT ....
PUNCH ....

STOCKS

RABBI ....Stocks
RABBLE ......Stocks advancing
RABBLING ...Stocks declining
RABID ....Stocks en route
RACER ....Stocks en route as follows
RACK ....Have you forwarded stocks bought on
RACKET ......Have you forwarded stocks sold on
RACY ....Have you forwarded stocks requested as per wire
RADDLE ......Have you forwarded stocks requested as per letter
RADDISH .....Stocks were sent through
RADIX ....Stocks were sent on the
RAFF ....Stocks were sent several days ago
RAFFLE ......Stocks ought to reach you
RAFTER ....
RAG ....In regard to this stock sent, please see

CHARLES A. STONEHAM & CO. 115

STOCKS—Continued

RAGGED ......Stocks not received; send a tracer; bought on the
RAGING ....Stocks not received; send a tracer; your advice states sent on
RAGLIN ....Stocks received on draft do not tally; shares missing as follows
RAGMAN ......Send missing shares immediately
RAIMENT .....Stocks received on draft do not tally; we are returning by registered mail shares over, as follows
RAILING.......Stocks finally received O. K.
RAINBOW ....Stocks not received; now several days late
RAISIN ....Stocks on this order to buy must be forwarded immediately, as our customers will not wait
RAKE ....Stocks on this order to sell must be forwarded immediately, as prompt delivery was guaranteed in order to make sales
RAKING ......Stocks must be delivered
RAKISH ....Send us by registered mail
RALLY ....Send us by express
RALPH ....Send us the following stocks today
RAIN ....Send us the following stocks tomorrow

R S U V W

Properties of a Good Cryptosystem

- There should be no way short of enumerating all possible keys to find the key from any reasonable amount of ciphertext and plaintext, nor any way to produce plaintext from ciphertext without the key.
- Enumerating all possible keys must be infeasible.
- The ciphertext must be indistinguishable from true random values.

Milestones in Modern Cryptography

1883 Kerckhoffs' Principles

1917-1918 Vernam/Mauborgne cipher (one-time pad)

1920s-1940s Mathematicization and mechanization of cryptography and cryptanalysis

1973 U.S. National Bureau of Standards issues a public call for a standard cipher.

1976 Diffie and Hellman describe public key cryptography

Kerckhoffs' Law

> The system must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.

In other words, the security of the system must rest entirely on the secrecy of the key.

Vernam/Mauborgne Cipher

- Exclusive-OR a key stream tape with the plaintext
- Online encryption of teletype traffic, combined with transmission
- For a one-time pad — which is provably secure — use true-random keying tapes and *never* reuse the keying material.
- If keying material is reusable, it's called a *stream cipher*

☞ Snake oil alert! *If the key stream is algorithmically generated, it's not a one-time pad!*

Mathematicization and Mechanization

- Mechanical encryptors (Vernam, Enigma, Hagelin, Scherbius)
- Mathematical cryptanalysis (Friedman, Rejewski et al, Bletchley Park)
- Machine-aided cryptanalysis (Friedman, Turing et al.)

Standardized Ciphers

- Until the 1970s, most strong ciphers were government secrets
- The spread of computers created a new threat
- Reportedly, the Soviets eavesdropped on U.S. grain negotiators' conversations
- NBS (now called NIST) issued a public call for a cipher; eventually, IBM responded
- The eventual result — via a secret process — was DES

Public Key Cryptography

- Merkle invents a public key distribution scheme
- Diffie and Hellman invent public key encryption and digital signatures, but do not devise a suitable algorithm with all of the desired properties. Rivest, Shamir, and Adelman invent their algorithm soon thereafter
- In fact, the British GCHQ had invented “non-secret encryption” a few years earlier.
- There have been claims, but no evidence, that the American NSA invented it even earlier

What We Have Today

- Encryption is completely computerized, and operates on bits
- The basic primitives of encryption are combined to produce very powerful results
- Encryption is by far the strongest weapon in the computer security arsenal; host and operating system software is by far the weakest link
- *Bad software trumps good crypto*

Cryptographic Primitives

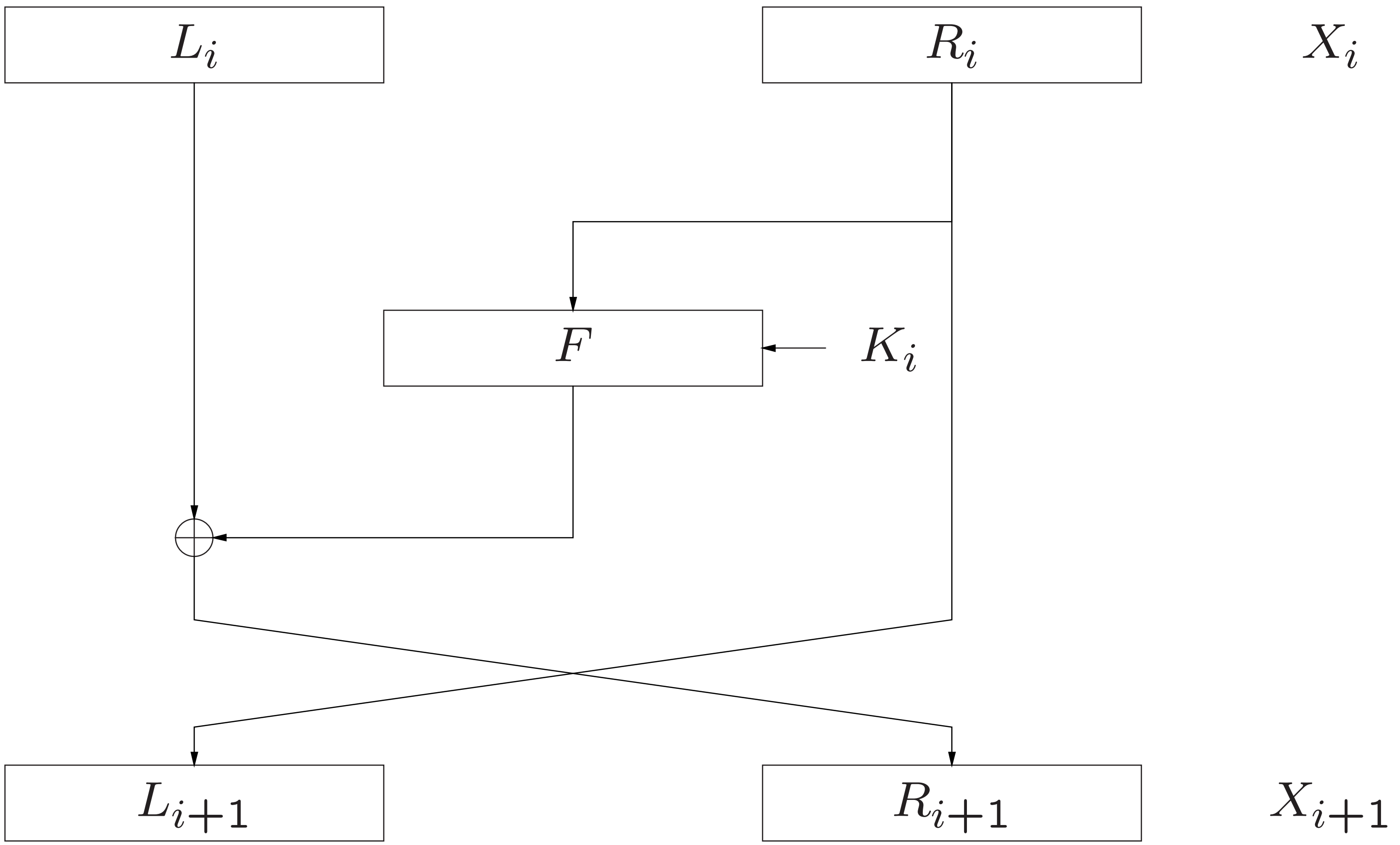
Block Ciphers

- Operate on a fixed-length set of bits
- Output blocksize generally the same as input blocksize
- Well-known examples: DES (56-bit keys; 64-bit blocksize); AES (128-, 192-, and 256-bit keys; 128-bit blocksize)

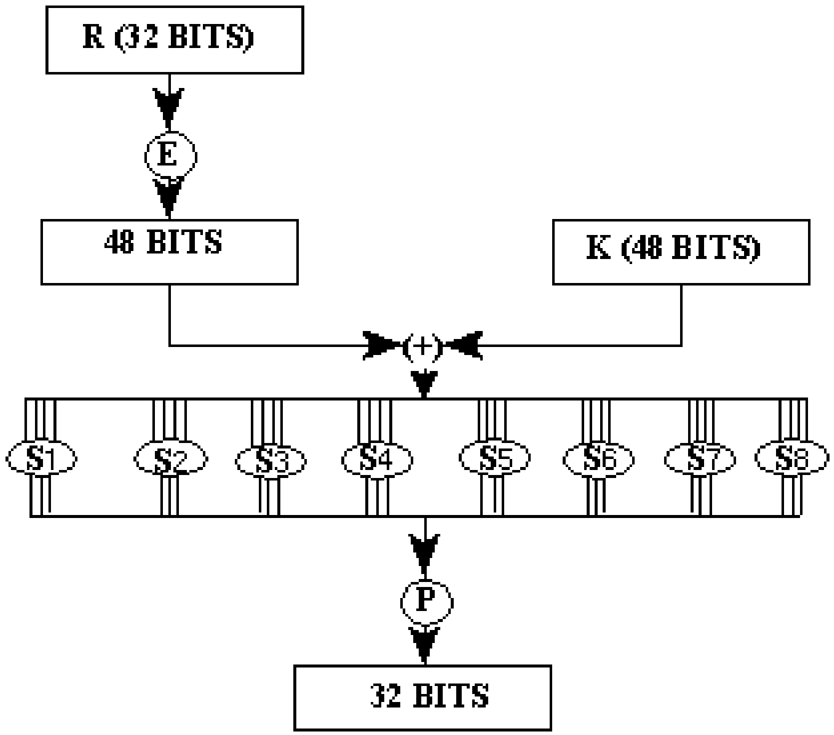
Basic Structure of (Most) Block Ciphers

- Optional key scheduling — convert supplied key to internal form
- Multiple *rounds* of combining the plaintext with the key.
- DES has 16 rounds; AES has 9-13 rounds, depending on key length

DES Round Structure



DES ”f” Funciton



How DES Works

For each round:

1. Divide the input block in half. The right half of each round becomes the left half of the next round's input.
2. Take the right half, pass it through a non-linear function of data and key, and exclusive-OR the result with the current input's left half.
3. The output of that function becomes the right half of the next round's input.

What's Wrong with DES?

- The key size is too short — a machine to crack DES was built in 1998.
- (Charges that NSA could crack DES were leveled in 1979. But the claim that NSA designed in a back door are false.)
- The blocksize is too short.
- It depends on bit-manipulation, and is too slow in software

Selecting the Advanced Encryption Standard

- NIST issued an open call for submissions
- 15 ciphers were submitted, from all over the world
- Several open conferences were held (and the NSA did its own private evaluations)
- 5 ciphers were eliminated as not secure enough
- 5 more were dropped for inefficiency or low security margin
- Of the 5 finalists, Rijndael — a Belgian submission — was chosen because of good security and very high efficiency across a wide range of platforms

How Does Rijndael Work?

- Input block viewed as a byte array; key viewed as a two-dimensional matrix
- Each round consists of a series of simple, byte-oriented operations: ByteSubstitution, ShiftRow, MixColumn, AddRoundKey.
- The key is mixed with the entire block in each round
- The basic operations are individually reasonably tractable mathematically, but are combined in a hard-to-invert fashion.

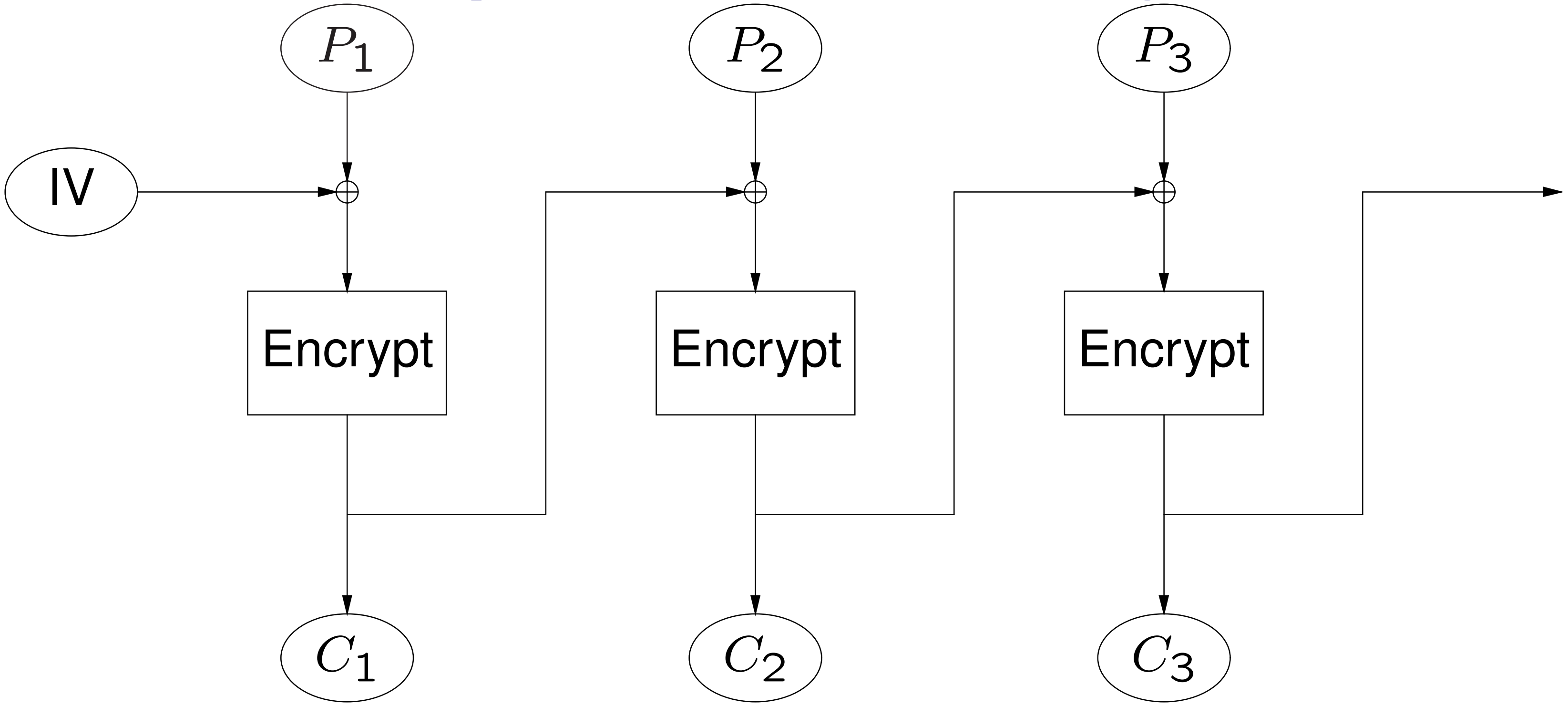
Modes of Operation

- Direct use of a block cipher is inadvisable
- Enemy can build up “code book” of plaintext/ciphertext equivalents
- Beyond that, direct use only works on messages that are a multiple of the cipher block size in length
- Solution: five standard *Modes of Operation*: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR).

Electronic Code Book

- Direct use of the block cipher
- Used primarily to transmit encrypted keys
- Very weak if used for general-purpose encryption; never use it for a file or a message.
- We write $\{P\}_k \rightarrow C$ to denote "encryption of plaintext P with key k to produce ciphertext C"

Cipher Block Chaining



$$\{P_i \oplus C_{i-1}\}_k \rightarrow C_i$$
$$\{C_i\}_{k^{-1}} \oplus C_{i-1} \rightarrow P_i$$

Properties of CBC

- The ciphertext of each encrypted block depends on the plaintext of all preceeding blocks.
- There is a dummy initial ciphertext block C_0 known as the *Initialization Vector* (IV); the receiver must know this value.
- Consider a 4-block message:

$$
\begin{aligned}
C_1 &= \{P_1 \oplus IV\}_k \\
C_2 &= \{P_2 \oplus C_1\}_k \\
C_3 &= \{P_3 \oplus C_2\}_k \\
C_4 &= \{P_4 \oplus C_3\}_k
\end{aligned}
$$

If C_2 is damaged during transmission, what happens to the plaintext?

Error Propagation in CBC Mode

- Look at the decryption process, where C' is a garbled version of C:

$$\begin{aligned} P_1 &= \{C_1\}_{k^{-1}} \oplus IV \\ P_2 &= \{C'_2\}_{k^{-1}} \oplus C_1 \\ P_3 &= \{C_3\}_{k^{-1}} \oplus C'_2 \\ P_4 &= \{C_4\}_{k^{-1}} \oplus C_3 \end{aligned}$$

- P_1 depends only on C_1 and IV, and is unaffected
- P_2 depends on C_2 and C_1, and hence is garbled
- P_3 depends on C_3 and C_2, and is also garbled. The enemy can control the change to P_3.
- P_4 depends on C_4 and C_3, and not C_2; it thus isn't affected.
- Conclusion: Two blocks change, one of them predicatably

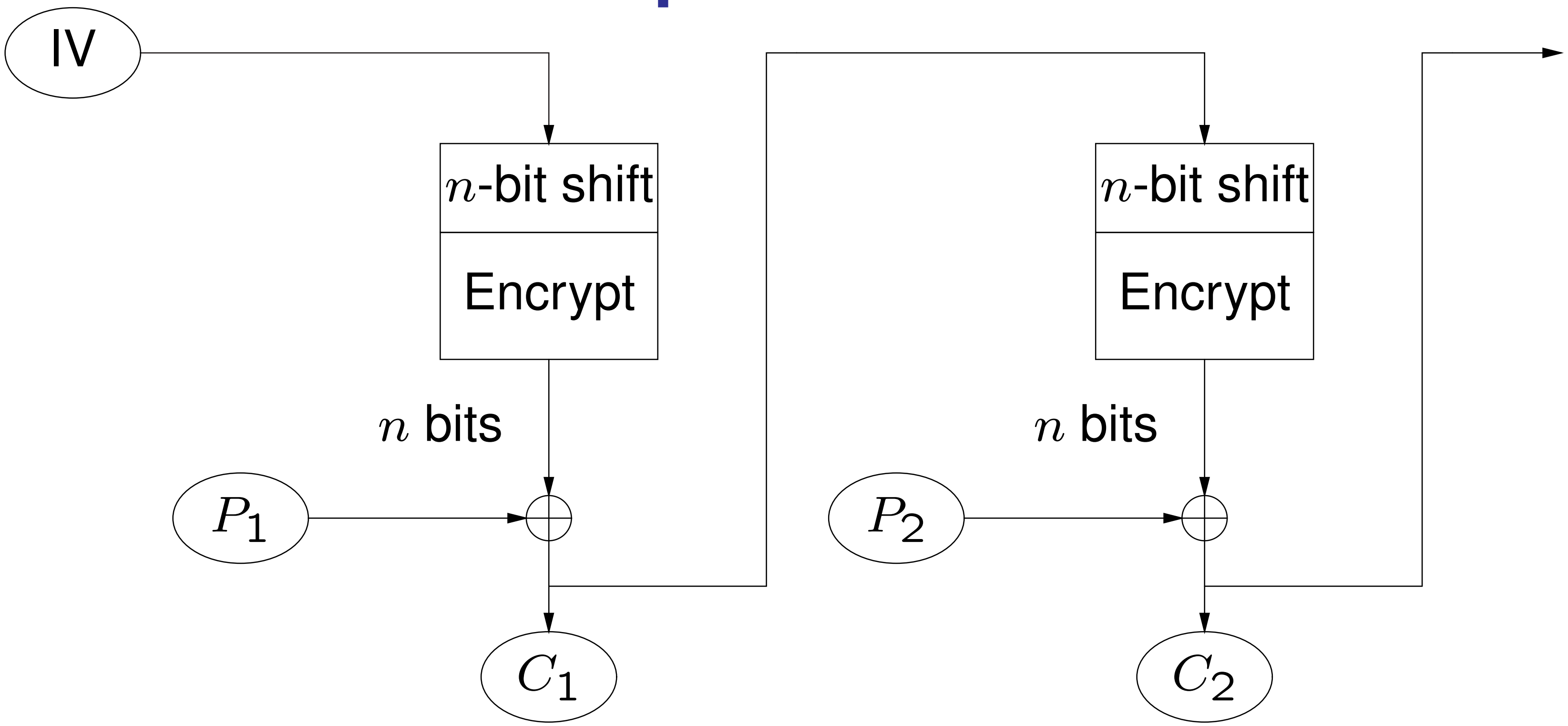
Cutting and Pasting CBC Messages

- Consider the encrypted message

$$IV, C_1, C_2, C_3, C_4, C_5$$

- The shortened message IV, C_1, C_2, C_3, C_4 appears valid
- The truncated message C_2, C_3, C_4, C_5 is valid: C_2 acts as the IV.
- Even C_2, C_3, C_4 is valid, and will decrypt properly.
- Any subset of a CBC message will decrypt cleanly.
- If we snip out blocks, leaving IV, C_1, C_4, C_5, we only garble one block of plaintext.
- Conclusion: if you want message integrity, you have to do it yourself.

n-bit Cipher Feedback

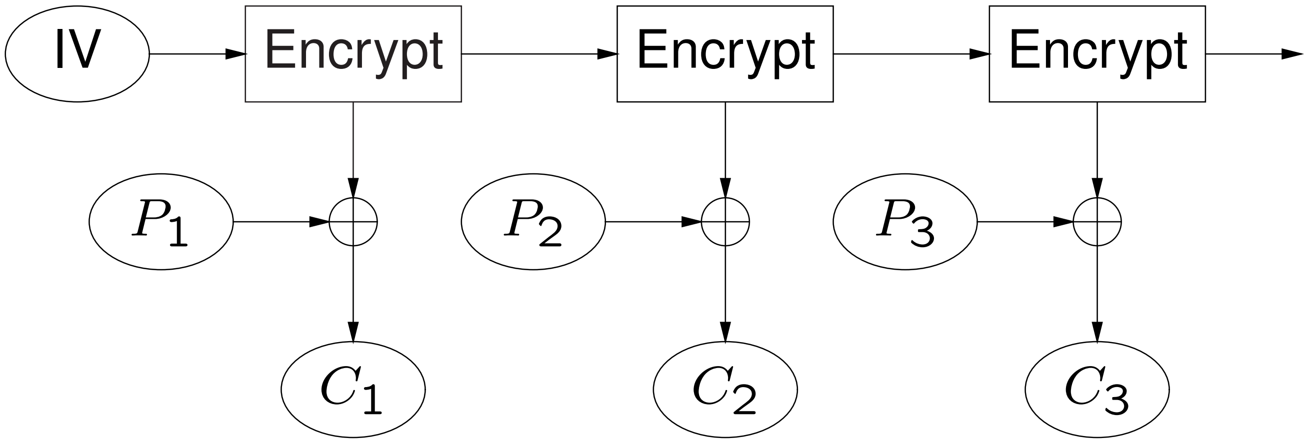


$$P_i \oplus \{C_{i-1}\}_k \rightarrow C_i$$
$$\{C_{i-1}\}_k \oplus C_i \rightarrow P_i$$

Properties of Cipher Feedback Mode

- Underlying block cipher used only in encryption mode
- Feedback path actually incorporates a shift register; some of the previous cycle's ciphertext can be retained.
- 8-bit CFB is good for asynchronous terminal traffic.
- Errors propagate while bad data is in the shift register — 17 bytes for CFB_8 when using AES.
- Copes gracefully with deletion of n-bit unit

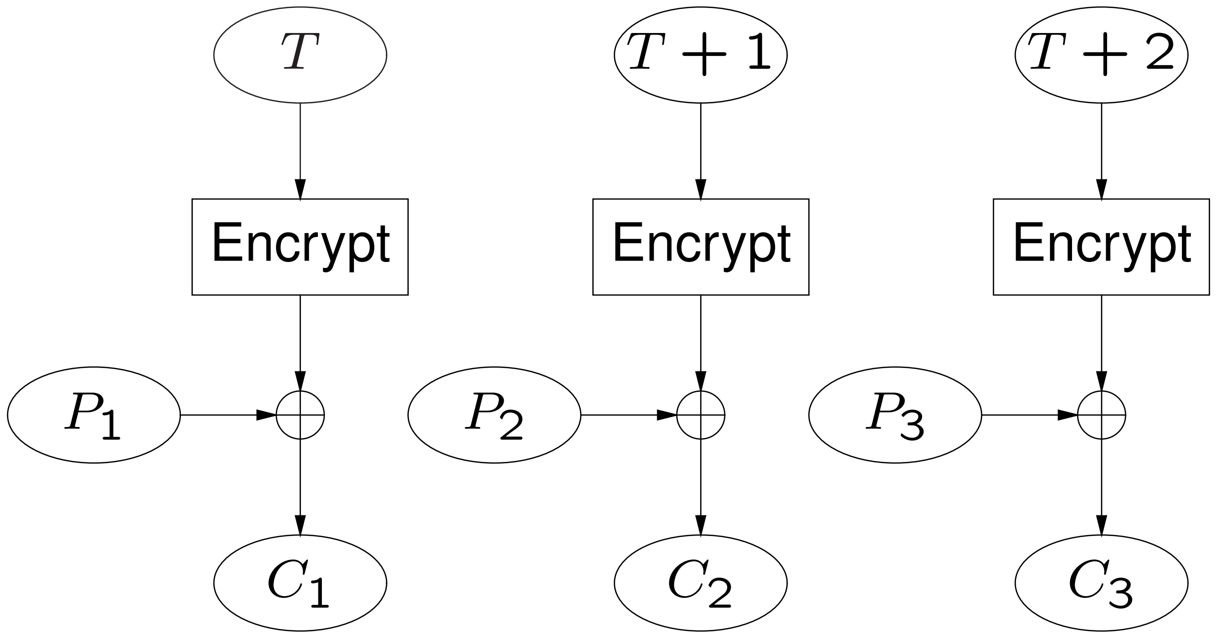
n-bit Output Feedback



Properties of Output Feedback Mode

- No error propagation
- Active attacker can make controlled changes to plaintext
- OFB is a form of *stream cipher*

Counter Mode



Properties of Counter Mode

- Another form of stream cipher
- Active attacker can make controlled changes to plaintext
- Highly parallelizable; no linkage between stages
- Vital that counter never repeat for any given key

Which Mode for What Task?

- General file or packet encryption: CBC.
  ☞Input must be padded to multiple of cipher block size
- Risk of byte or bit deletion: CFB_8 or CFB_1
- Bit stream; noisy line and error propagation is undesirable: OFB
- Very high-speed data: CTR
- In most situations, an integrity check is needed

Stream Ciphers

- Key stream generator produces a sequence S of pseudo-random bytes; key stream bytes are combined (generally via XOR) with plaintext bytes
- $P_i \oplus S_i \rightarrow C_i$
- Stream ciphers are very good for asynchronous traffic
- Best-known stream cipher is RC4; commonly used with SSL
- Key stream S must *never* be reused for different plaintexts:

$$\begin{aligned} C &= A \oplus K \\ C' &= B \oplus K \\ C \oplus C' &= A \oplus K \oplus B \oplus K \\ &= A \oplus B \end{aligned}$$

- Guess at A and see if B makes sense; repeat for subsequent bytes

RC4

- Extremely efficient
- After key setup, it just produces a key stream
- No way to resynchronize except by rekeying and starting over
- Internal state is a 256-byte array plus two integers
- Note: weaknesses if used in ways other than as a stream cipher.

The Guts of RC4

```
for(counter = 0; counter < buffer_len; counter ++)
{
     x = (x + 1) % 256;
     y = (state[x] + y) % 256;
     swap_byte(&state[x], &state[y]);
     xorIndex = (state[x] + state[y]) % 256;
     buffer_ptr[counter] ^= state[xorIndex];
 }
```

Cipher Strengths

- A cipher is no stronger than its key length: if there are too few keys, an attacker can enumerate all possible keys
- DES has 56 bits — arguably too few in 1976; far too few today.
- Strength of cipher depends on how long it needs to resist attack.
- No good reason to use less than 128 bits
- NSA rates 128-bit AES as good enough for SECRET traffic; 256-bit AES is good enough for TOP-SECRET traffic.
- But a cipher can be considerably weaker! (A monoalphabetic cipher over all bytes has a 1684-bit key, but is trivially solvable.)

Public Key Cryptography

- Separate keys for encryption and decryption
- Not possible to derive decryption key from encryption key
- Permissible to publish encryption key, so that anyone can send you secret messages
- All known public key systems are very expensive to use, in CPU time and bandwidth.
- Most public systems are based on mathematical problems.

RSA

- The best-known public key system is RSA.
- Generate two large (at least 512 bit) primes p and q; let $n = pq$
- Pick two integers e and d such that $ed \equiv 1 \bmod (p-1)(q-1)$. Often, $e = 3$, since that simplifies encryption calculations.
- The public key is $\langle e, n \rangle$; the private key is $\langle d, n \rangle$.
- To encrypt m, calculate $c = m^e \bmod n$; to decrypt c, calculate $m = c^d \bmod n$.
- The security of the system relies on the difficulty of factoring n.
- Finding such primes is relatively easy; factoring n is believed to be extremely hard.

Classical Public Key Usage

- Alice publishes her public key in the phone book.
- Bob prepares a message and encrypts it with that key by doing a large exponentiation.
- Alice uses her private key to do a different large exponentiation.
- It's not that simple...

Complexities

- RSA calculations are *very* expensive; neither Bob nor Alice can afford to do many.
- RSA is too amenable to mathematical attacks; encrypting the wrong numbers is a bad idea.
- Example: "yes"^{3} is only 69 bits, and won't be reduced by the modulus operation; finding $\sqrt[3]{503565527901556194283}$ is easy.
- We need a better solution

A More Realistic Scenario

- Bob generates a random key k for a conventional cipher.
- Bob encrypts the message: $c = \{m\}_k$.
- Bob pads k with a known amount of padding, to make it at least 512 bits long; call this k'.
- k' is encrypted with Alice's public key $\langle e, n \rangle$.
- Bob transmits $\{c, (k')^e \bmod n\}$ to Alice.
- Alice uses $\langle d, n \rangle$ to recover k', removes the padding, and uses k to decrypt ciphertext c.
- In reality, it's even more complex than that...

Perfect Forward Secrecyy

- If an endpoint is compromised (i.e., captured or hacked), can an enemy read old conversations?
- Example: if an attacker has recorded $\{c, (k')^e \bmod n\}$ and then recovers Alice's private key, he can read c.
- Solution: use schemes that provide *perfect forward secrecy*, such as Diffie-Hellman key exchange.

Diffie-Hellman Key Exchange

- Agree on a large (at least 1024-bit) prime p, usually of the form $2q + 1$ where q is also prime.
- Find a generator g of the group “integers modulo p”. (This is easy to do if p is prime.)
- Alice picks a large random number x and sends Bob $g^x \bmod p$. Bob picks a large random number y and sends Alice $g^y \bmod p$.
- Alice calculates $k = (g^y)^x \equiv g^{xy} \bmod p$; Bob does a similar calculation.
- If x and y are really random, they can't be recovered if Alice or Bob's machine is hacked.
- Eavesdroppers can't calculate x from $g^x \bmod p$, and hence can't get the shared key. This is called the *discrete logarithm* problem.

Random Numbers

- Random numbers are very important in cryptography.
- They need to be as random as possible — an attacker who can guess these numbers can break the cryptosystem. (This is a a common attack!) To the extent possible, use true-random numbers, not pseudo-random numbers.
- Where do true-random numbers come from?
- Physical processes are best — radioactive decay, thermal noise in amplifiers, oscillator jitter, etc.
- Often, a true-random number is used to seed a cipher — modern cryptographic functions are very good pseudo-random numbers.

Digital Signatures

- RSA can be used backwards: you can encrypt with the private key, and decrypt with the public key.
- This is a *digital signature*: only Alice can sign her messages, but anyone can verify that the message came from Alice.
- Again, it's too expensive to sign the whole message. Instead, Alice calculates a *cryptographic hash* of the message and signs the hash value.
- If you sign the plaintext and encrypt the signature, the signer's identity is concealed; if you sign the ciphertext, a gateway can verify the signature without having to decrypt the message.

Cryptographic Hash Functions

- Produce relatively-short, fixed-length output string from arbitrarily long input.
- Computationally infeasible to find two different input strings that hash to the same value
- Computationally infeasible to find any input string that hashes to a given value
- Strength roughly equal to half the output length
- Best-known cryptographic hash functions: MD5 (128 bits), SHA-1 (160 bits), SHA-256 (256 bits)
- 128 bits and shorter are not very secure for general usage

Late-Breaking News

- At CRYPTO '04, several hash functions were cracked.
- More precisely, collisions were found: $H(M) = H(M'), M \neq M'$
- Cracked functions include MD4, MD5, HAVAL-128, RIPEMD, and SHA-0.
- But SHA-0 was known to be flawed; NSA replaced it with SHA-1 in 1994

☞ No significant weaknesses were found in SHA-1.

- MD5 is still commonly used, though weaknesses have long been suspected.

Abusing a Weak Hash Function

- Alice prepares two contracts, m and m', such that $H(m) = H(m')$
- Contract m is favorable to Bob; contract m' is favorable to Alice

☞ The exact terms aren't important; Alice can prepare many different contracts while searching for two suitable ones.

- Alice sends m to Bob; he signs it, producing $\{H(m)\}_{K_B{}^{-1}}$.
- Alice shows m' and $\{H(m)\}_{K_B{}^{-1}}$ to the judge and asks that m' be enforced
- Note that the signature matches...

Elliptic Curve Cryptography

- Public key and D-H algorithms, but based on more complex math
- Considerably more security per key bit; allows for shorter keys
- More importantly, allows for much more efficient computation
- Many patents in this space

Rough Table of Key Length Equivalences

Symmetric Key Size (bits)	*RSA or DH Modulus Size (bits)*
70	947
80	1228
90	1553
100	1926
150	4575
200	8719
250	14596

(Numbers by Orman and Hoffman)

Cryptographic Combinations and Protocols

Building Blocks

- Conventional (symmetric) ciphers, plus modes of operation
- Cryptographic hash functions
- Public key (asymmetric) ciphers
- Using them properly is complex (and not for amateurs)

Problems and Threats

- Confidentiality
- Message integrity
- Watch out for replayed messages
- Attacker can cut-and-paste message pieces
- Make sure message goes to right party

What are the Attacker's Powers?

- Eavesdropping
- Message insertion
- Message deletion
- Message modification
- Suitably paranoid attitude: you give your packets to the enemy to deliver

Message Integrity

- We can use a key and a cryptographic hash to generate a *Message Authentication Code* (MAC).
- Best-known construct is HMAC — provably secure under minimal assumptions
- HMAC$(m, k) = H(k, H(k, m))$ where H is a cryptographic hash function
- Can also do a CBC encryption and retain only the last ciphertext block
- Note: authentication key, for either scheme, *must* be distinct from the confidentiality key

What are Certificates

- How does Alice get Bob's public key?
- What if the enemy tampers with the phone book? Sends the phone company a false change-of-key notice? Interferes with Alice's query to the phone book server?
- A certificate is a digitally-signed message containing an identity and a public key — prevents tampering.

Why Trust a Certificate?

- Who signed it? Why do you trust them?
- How do you know the public key of the *Certificate Authority* (CA)?
- Some public key (known as the *trust anchor*) must be provided out-of-band — trust has to start somewhere.

Certificate Authorities

- Who picks CAs? No one and every one.
- Your browser has some CAs built-in — because the CA paid the browser vendor enough money. Is that grounds for trust?
- Matt Blaze: “A commercial certificate authority can be trusted to protect you from anyone from whom they won’t take money.”

Who Gets Certificates?

- How do you prove your identity to a CA?
- How good a job do they do verifying it?
- What warranties does the CA give if someone is fooled? (Most disclaim all liability. . .)

Another Trust Model

- Get certificates from parties whom you know.
- Issue certificates to your own users: *authorization certificates*
- Don’t rely on commercial identity-based CAs.

Certificate Hierarchy versus Web of Trust

- Most CAs are tree-structured
- Top-level CAs can use *bridge CAs* to cross-certify each other
- PGP uses a different style: a *web of trust*.
- Certificate signings can form an arbitrarily-complex graph — users can verify path to as many trust anchors as they wish.

Styles of Certification

- At least 3 major styles
- X.509/PKIX — traditional hierarchical CA (but can have “pki” instead of “PKI”)
- SPKI/SDSI — authorization certificates
- PGP web of trust (primarily for email)

What Else is in a Certificate?

- Technical information, such as algorithm identifiers
- More identification information — company, location, etc.
- Expiration date
- Logos
- Certificate role

Not All Certificates are Alike

- An email certificate isn't the same as an ecommerce certificate.
- A CA certificate is even more different — can I use my att.com email certificate to issue more att.com certificates?
- What about a code-signing certificate for ActiveX?
- Usage-specific information, such as IP address range

Revoking Certificates

- Keys associated with certificates can be compromised
- One choice – *certificate revocation list* (CRL)
- Can get large, which is one reason why certificates expire
- For connected hosts, possible to do online certificate status checking
- Can the attacker block connectivity to the status server?
- CRLs are the weak link of public key cryptography.

Key Management for Symmetric Ciphers

- Simplest case: each pair of communicators has a shared key
- Doesn't scale.
- Besides, cryptographically unwise — each key is used too much
- Need a *Key Distribution Center* (KDC)

Desired Properties

- Alice and Bob want to end up with a shared session key K, with the help of a key server S.
- They each want proof of the other's identity
- They want to be sure the key is fresh

Needham-Schroeder Protocol (1978)

$$A \rightarrow S: \quad A, B, N_A \tag{1}$$

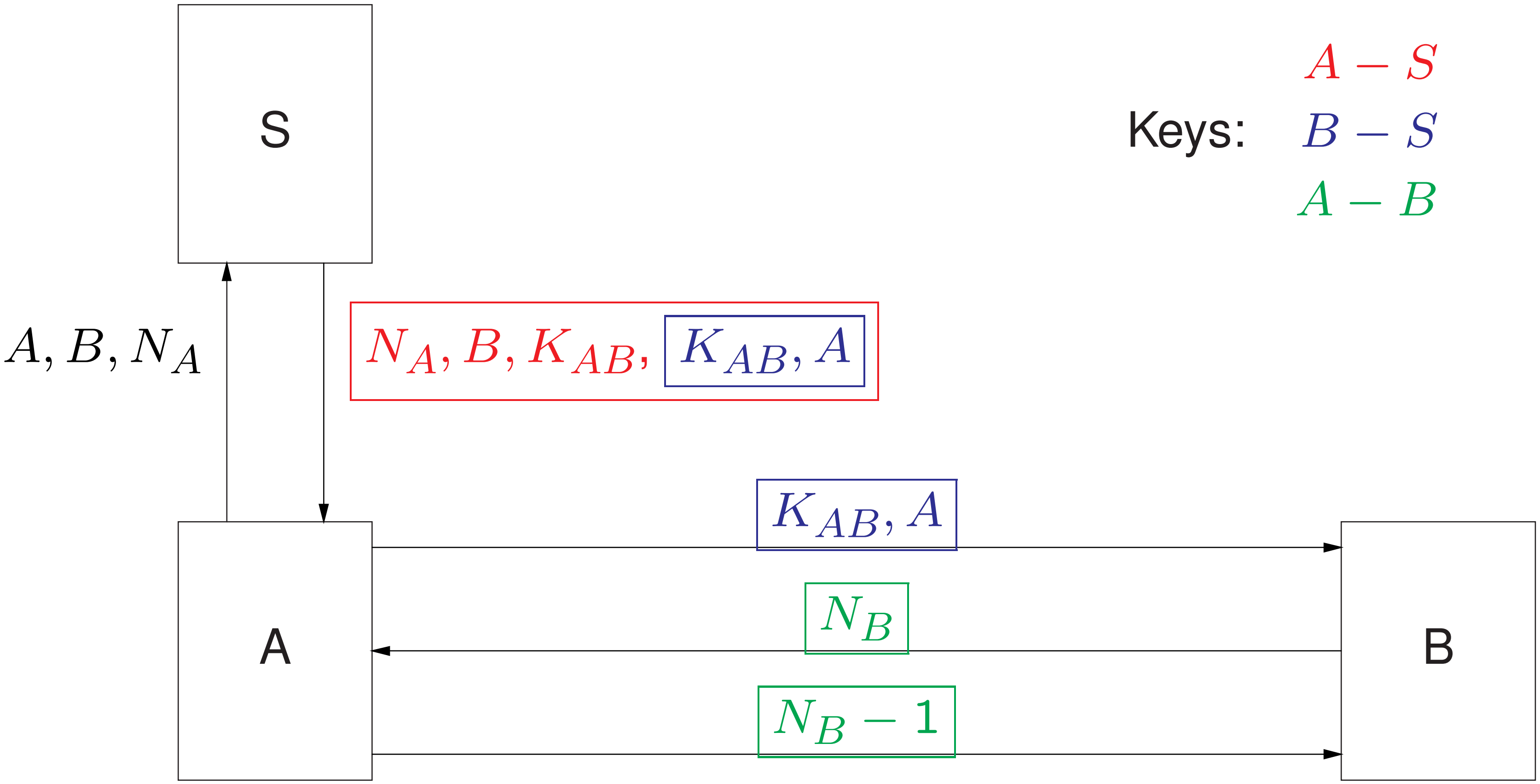
$$S \rightarrow A: \quad \{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \tag{2}$$

$$A \rightarrow B: \quad \{K_{AB}, A\}_{K_{BS}} \tag{3}$$

$$B \rightarrow A: \quad \{N_B\}_{K_{AB}} \tag{4}$$

$$A \rightarrow B: \quad \{N_B - 1\}_{K_{AB}} \tag{5}$$

Needham-Schroeder Protocol



Explaining Needham-Schroeder

(1) Alice sends S her identity, plus a random *nonce*

(2) S's response is encrypted in K_{AS}, which guarantees its authenticity. It includes a new random session key K_{AB}, plus a sealed package for Bob

(3) Alice sends the sealed package to Bob. Bob knows it's authentic, because it's encrypted with K_{BS}

(4) Bob sends his own random nonce to Alice, encrypted with the session key

(5) Alice proves that she could read the nonce

Cryptographic Protocol Design is Hard

- Bob never proved his identity to Alice
- If K_{AB} is ever compromised, the attacker can impersonate Alice forever
- Denning and Sacco proposed a fix for this problem in 1981.
- In 1994, Needham found a flaw in their fix.
- In 1995, a new flaw was found in the public key version of the original Needham-Schroeder protocol — in modern notation, that protocol is only 3 messages.
- Cryptographic protocol design is hard...

Kerberos

- Originally developed at MIT; now an essential part of Windows authentication infrastructure.
- Designed to authenticate users to servers
- Users must use their password as their initial key — and must not be forced to retype it constantly
- Based on Needham-Schroeder, with timestamps to limit key lifetime

How Kerberos Works

- Users present *tickets* — cryptographically sealed messages with session keys and identities — to obtain a service.
- Use Needham-Schroeder (with password as Alice's key) to get a *Ticket-Granting Ticket* (TGT); this ticket (and the associated key) are retained for future use during its lifetime.
- Use the TGT (and TGT's key) in a Needham-Schroeder dialog to obtain keys for each actual service
- (Actual details are more complex, and are left as an exercise.)

Attacking DH Exponential Key Exchange

Suppose we have a man-in-the-middle between Alice and Bob...

$$\begin{aligned} A \rightarrow M &: \quad g^x \bmod p \\ M \rightarrow B &: \quad g^z \bmod p \\ B \rightarrow M &: \quad g^y \bmod p \\ M \rightarrow A &: \quad g^{z'} \bmod p \end{aligned}$$

Alice and M share a key $g^{xz} \bmod p$; Bob and M share a key $g^{yz'} \bmod p$.

When Alice sends a message towards Bob, M decrypts it, reads it and perhaps modifies it, re-encrypts it, and sends it to Bob.

Diffie-Hellman key exchange provides no authentication — and if Alice or Bob sent a password, M would read that, too.

Authenticating Diffie-Hellman

- Alice and Bob — and perhaps M — engage in a Diffie-Hellman exchange.
- Bob digitally signs a hash of the exchanged exponentials, and transmits it; Alice does the same.
- M can't tamper with digitally-signed messages.
- If there's an attacker, Alice and Bob realize that the signed key doesn't match their own key, so they know there's something wrong.
- (Station-to-station protocol)

Coin Flips

- How do you flip a coin on the Internet, without a trusted third party?
- Alice picks a random number x, and sends $H(x)$ to Bob.
- Bob guesses if x is even or odd, and sends his guess to Alice.
- If Bob's guess is right, the result is heads; if he's wrong, the result is tails.
- Alice discloses x. Both sides can verify the result. Alice can't cheat, because she can't find an x' such that $H(x) = H(x')$.
- Note: this protocol is crucially dependent on the lack of correlation between the parity of x and the values of $H(x)$, or Bob can cheat.

Cryptography and the Internet

Cryptographic Niches

- Cryptography used in may different places
- The most successful forms are the ones that take no user effort
- But — not as much crypto as there should be

Where to Encrypt?

- Link layer? Only protects that hop, but guards against traffic analysis. Example: WEP.
- Network layer? Protects all applications, and protects transport layer headers, but requires kernel or hardware changes. Example: IPsec.
- Transport layer? Easy to install in applications, but requires changes to the application. Example: SSL.
- Application layer? Can be intrusive, but can provide a close match to the application's semantics. Example: S/MIME.

What to Encrypt?

- Conversations
  - Can use KDC or public key
  - Primary place where Diffie-Hellman is used
  - In case of failure, negotiate a new key
- Objects
  - Email, files, non-end-to-end messages
  - Public key or external key input often used
  - Key loss may be asynchronous — hard to recover

Secure Socket Layer (SSL)

- Most common form of cryptography on the Internet
- (IETF-standardized version of SSL is called TLS — Transport Layer Security.)
- Used by Web browsers for ecommerce.
- No setup required by users; some effort required by Web servers.
- In practice, not as secure as in theory, because users don't validate the results

How Does SSL Work?

- Client and server do a Diffie-Hellman exchange to generate a session key.
- Server signs the session key, and sends the signature and its certificate to the client.
- The client (usually) doesn't have a certificate: the server doesn't care who the client is
- The client knows there's no MITM attack. The client also verifies that the name in the certificate matches the desired Web site's name.
- Both sides cache the session key, and attempt to reuse it if possible.

Is SSL Secure?

- No known flaws in the protocol — but usage is a different matter
- Very, very few users validate the certificate chain to some CA that they trust. (Most don't know what a certificate chain or a trust anchor are.)
- Are you connecting to the right site? Maybe the attacker changed the last unencrypted HTTP message to point to an evil Web site instead. You have a secure connection — to the wrong place.
- You don't go through the crypto; you go around it.

Status

- Implemented in all web browsers and servers
- Open source implementation available
- Installed in many email clients and servers

Other Uses of SSL

- SSL is actually a general-purpose secure transport layer.
- Other protocols can be carried over SSL, including (especially) email uploads and downloads.
- Easiest way to bolt crypto on top of any TCP application.
- The difficulty is application-specific: how do you negotiate the use of SSL?

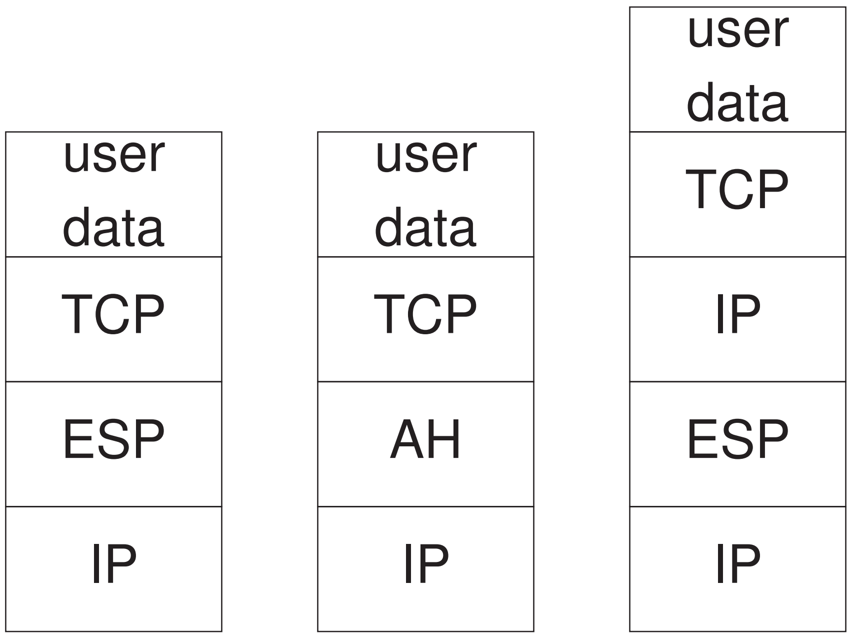
IPsec and IKE

- IPsec is network-layer encryption for the Internet
- IKE is the key management layer
- Conceptually simple, but the devil is in the details

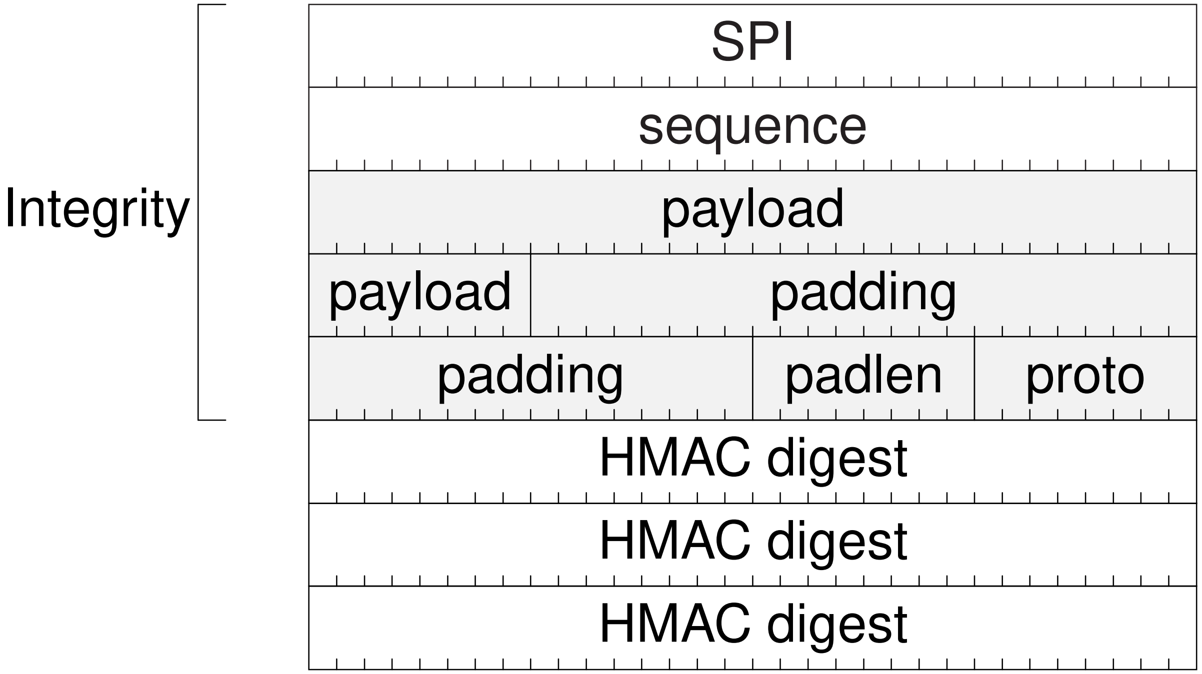
What is Network Layer Security?

- Adds an encryption layer just above IP
- Protects everything above it, including the TCP or UDP header
- Transparent to applications; can protect all of them without any changes
- Can provide replay protection, packet authentication only (AH, sometimes ESP) or authentication and confidentiality (ESP)
- But — requires kernel or hardware changes

Possible IPsec Packet Layouts



Encapsulating Security Protocol (ESP)



The shaded portion of the packet is encrypted. Null confidentiality algorithm available; replay-checking and authentication optional.

How ESP Works

- Security Parameter Index (SPI) identifies *Security Association* (SA) — what packets should be encrypted, and with what key and algorithm
- Sequence number used to prevent replay attacks. Packets need not be strictly ordered, but duplicate packets are dropped
- Payload contains more headers, plus user data
- Padding used to round up to CBC blocksize, plus foil traffic analysis.
- “Proto” is the protocol number of the payload
- The HMAC digest is the (truncated) authentication code

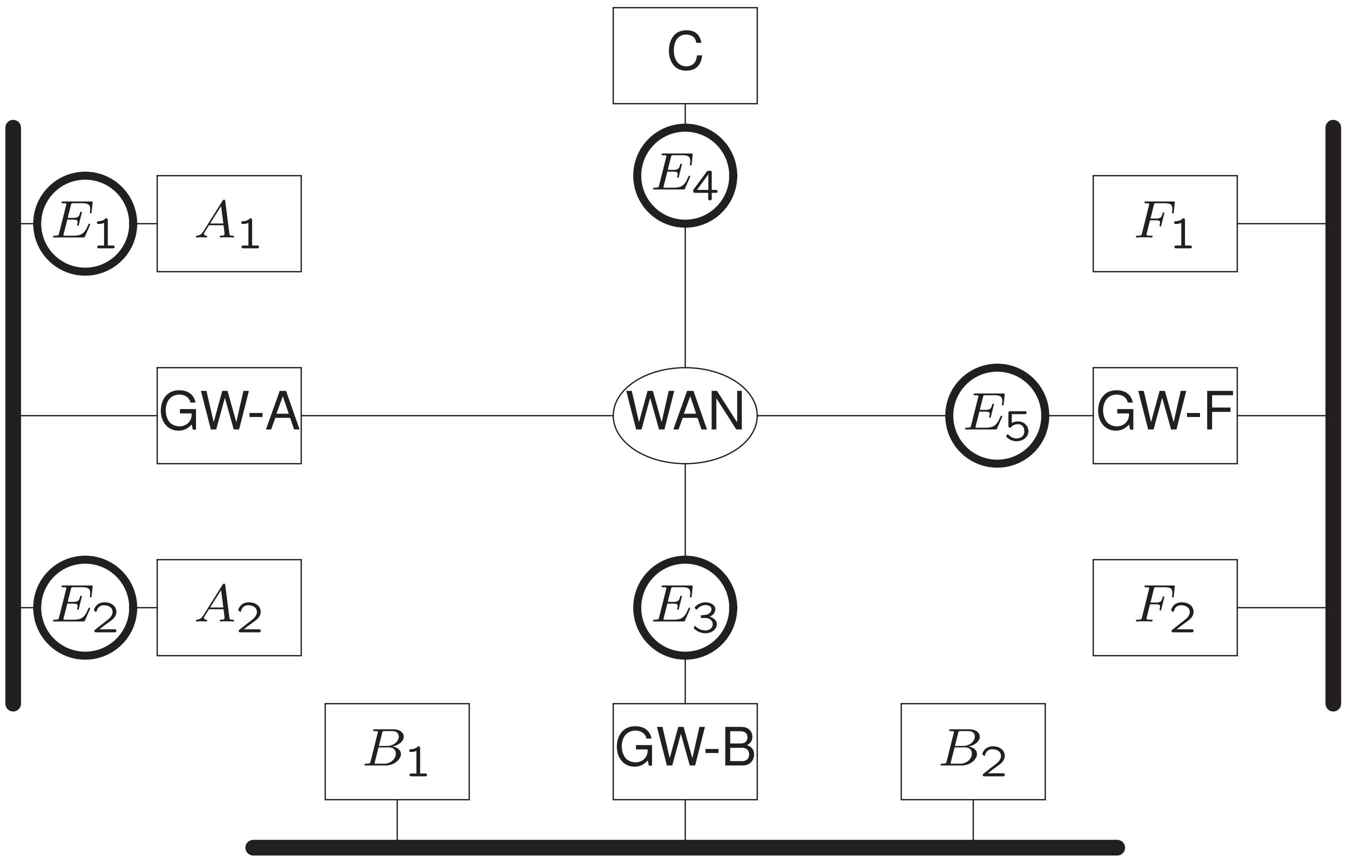
Authentication Header (AH)

next proto	length	reserved
SPI		
sequence		
HMAC (variable length)		

How AH Works

- SPI the same as for ESP
- Optional sequence number checking
- The HMAC digest covers portions of the *preceeding* IP header, as well as whatever follows the AH header

IPsec Topologies



Using IPsec

- IPsec can be host-to-host, host-to-gateway, or gateway-to-gateway
- Traffic from A_1 to C is protected by IPsec encryptors E_1 and E_4. (Encryptors can be in the stack, in the NIC, or a "bump in the wire".)
- This is *transport mode* IPsec; the header following th IPsec header is the TCP or UDP header.
- Traffic from C to anywhere on Network B is protected by E_4 and E_3.
- Traffic from anywhere on Net F to anywhere on Net B is protected by E_5 and E_3.
- The latter two cases use *tunnel mode*, and have an IP header following the IPsec header. This is the Virtual Private Network (VPN) usage.

Internet Key Exchange (IKE)

- Key exchange and management layer for IPsec
- *Very* complex — many different options, variants, etc.
- Basic cryptographic scheme is (more or less) the station-to-station protocol, but with lots of extra features
- Also creates and destroys security associations: exactly what IP addresses and port numbers are to be protected.
- (Note: IPsec receivers check incoming packets to be sure they were protected with the proper algorithms and keys.)

Status of IPsec

- All major platforms have IPsec either in the base system or as a readily-available add-on
- Implementations interoperate less than they should — there are too many options and variations, especially in IKE and in certificate semantics
- Primarily used for VPNs; host-to-use use is rare

S/MIME — Secure Email

- Provides mechanism for secure attachments
- Attachments can be encrypted, digitally signed, or both
- Provision for multiple recipients, which implies multiple public key-encrypted message keys
- Uses X.509 certificates
- Must also adapt to email oddities — actual over-the-wire format must be immune to havoc wreaked by some email systems

Status of S/MIME

- Many Mail User Agents (MUA) implement S/MIME, but it is little-used
- Many hard-core geeks prefer PGP
- The crucial difference between the two is the certificate trust model; the (many) syntactic incompatibilities are minor by comparison

WEP — Using a Flawed Cipher in a Bad Way for the Wrong Application

- WEP — *Wireline Equivalent Privacy* for 802.11 netorks
- Many different mistakes
- Case study in bad crypto design

Datagrams and Stream Ciphers

- WEP uses RC4 because of its efficiency
- But 802.11 is datagram-oriented; there's no inter-packet byte stream to use

☞ Must rekey for every packet

- But you can't reuse a stream cipher key on different packets. . .

Key Setup for WEP

- Each WEP node keeps a 24-bit packet counter
- Actual cipher key is configured key concatenated with counter
- Two different flaws. . .
- 2^{24} packets isn't that many — you still get key reuse when the packet counter overflows
- RC4 has a cryptanalytic flaw — the key values for different packets don't differ in many bit positions, and there's a feasible *related key* attack
- But it's worse than that

The Biggest Flaw in WEP

- There's no key management; all users at a site always share the same WEP key.
- ☞ You can't rekey when the counter overflows
- ☞ Everyone shares the same key; if it's cryptanalzed or stolen or betrayed, everyone is at risk
- ☞ It's all but impossible to rekey a site of any size, since everyone has to change their keys simultaneously and you don't have a secure way to provide the new keys

What WEP Should Have Been

- Use a block cipher in CBC mode
- Use a separate key per user, plus a key identifier like the SPI
- Provide dynamic key management
- WPA — WiFi Protected Access — is better than WEP; forthcoming wireless security standards will use AES.

Threats to Cryptographic Systems

Cryptography Has its Limits

- Cryptanalysis
- Broken protocols
- Operational errors
- *The endpoints!*

Endpoint Security

- On the Internet, most security problems are caused by buggy software or misconfigured systems
- Cryptography doesn't help here!
- Attackers don't have to go through the crypto, they go around it.
- Real world example: don't crack SSL; hack into the merchant's Web site and steal lots of credit card numbers at once.

Cryptanalysis

- Cryptanalysis is not a major threat today — there's a lot of math behind today's algorithms. There are also academic cryptographers who look at many new algorithms.
- The most common reason for rejecting a proposed cipher is a *certificational weakness* — a minor hint that something isn't quite as good as it should be.
- Of course, historically there were lots of people who believed — erroneously — that their ciphers were secure...
- Still, most plausible cryptanalytic attacks on modern ciphers will require a great deal of effort per key or message recovered.
- In other words, your enemy needs to have lots of resources — and frequent key changes will make the attacker's job harder.

Recent Cryptanalytic Successes

- Related key attack against RC4
- Million-message attack against SSL — the format used to pad a key before RSA encryption wasn't good enough.
- Deep Crack — privately-funded brute force DES cracker

Broken Protocols

- Cryptographic protocol design is hard
- Original version of IPsec had many different flaws
- Many deployed protocols have been erroneous

A Flaw in Public Key Needham-Schroeder

CA is Alice's certificate; CB is Bob's certificate. Alice's private key is $K_{a^{-1}}$; Bob's public key is K_b. T_a is a timestamp.

$$
\begin{aligned}
A \rightarrow S: &\quad A, B \\
S \rightarrow A: &\quad CA, CB \\
A \rightarrow B: &\quad CA, CB, \{\{K_{ab}, Ta\}_{K_a{}^{-1}}\}_{K_b}
\end{aligned}
$$

Where's the flaw?

Some Hints

- It's a cut-and-paste attack
- It allows Bob to impersonate Alice
- The flaw went undetected from 1978 to 1995...

A Cut-and-Paste Attack

Bob can transform this

$$A \rightarrow B: \quad CA, CB, \{\{K_{ab}, Ta\}_{K_a{}^{-1}}\}_{K_b}$$

into this:

$$B \rightarrow C: \quad CB, CC, \{\{K_{ab}, Tc\}_{K_a{}^{-1}}\}_{K_b}$$

and send the message to Carol.

Bob can read that segment (including the session key K_{ab}) because it's encrypted with his public key. Carol does nothing to ensure that it's fresh.

Simplest fix: Alice can sign the ciphertext of a message encrypted to Bob:

$$A \rightarrow B: \quad CA, CB, \{\{K_{ab}, Ta\}_{K_b}\}_{K_a{}^{-1}}$$

Operational Errors

- Operational errors can weaken even a strong cipher.
- If the Germans had used the ENIGMA properly, it's doubtful that the Allies could have cracked it.
- Using a password as a cryptographic key is a modern example — and in fact is quite close to one of the common German errors.
- A major design goal for crypto gear is to minimize the chances of user operational error.

Common Operational Problems

- Bad random number generators or bad seeds
- Leaving keys or plaintext on compromised or seized machines
- Swap files often have copies of keys
- Uncleared buffers saved and transmitted by some applications (i.e., Microsoft Word)
- Applications that can silently disable crypto

Weird Attacks

- Timing attacks — guess at secret key by how long calculations take.

☞ Demonstrated on LANs

- Power attacks — guess at bit values by seeing the instantaneous power consumption
- Serious threat to smart cards
- Spurious RF emissions — so-called *TEMPEST* issue

References

General Cryptography

- *Handbook of Applied Cryptography*, Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone, CRC Press, 1996.
- *Applied Cryptography*, Bruce Schneier, second edition, Wiley, 1995.
- *The Codebreakers*, David Kahn, second edition, Scribner, 1996.
- *Seizing the Enigma: The Race to Break the German U-Boat Codes, 1939–1943*, David Kahn, Houghton Mifflin, 1991.

Classic Papers

- "Using Encryption for Authentication in Large Networks of Computers", R. M. Needham and M. Schroeder, *Comm. ACM* 21:12 (1978), pp. 993-999.
- "New Directions in Cryptography", Whitfield Diffie and Martin E. Hellman, *IEEE Transactions on Information Theory* IT-11 (1976), pp. 644-654.
- "A Method of Obtaining Digital Signatures and Public-Key Cryptosystems", Ronald L. Rivest, Adi Shamir, and Leonard Adleman, *Comm. ACM* 21:2 (1978), pp. 120–126.

IETF Standards

IPsec RFC 2401–2409. (Note: these are being revised.)

SSL/TLS RFC 2246

SSL and TLS: Designing and Building Secure Systems, Eric Rescorla, Addison-Wesley, 2000.

S/MIME RFC 3370, 3850–3852.

PKIX RFC 2510–2511, 2560, 2797, 3161, 3279–3280.