CIS 6930/4930 Computer and Network Security

Final exam review

About the Test

- This is an open book and open note exam.
  - You are allowed to read your textbook and notes during the exam;
  - You may bring your laptop to the exam but you are not allowed to access to internet during the exam.
  - Before midterm 30%, after midterm 70%

Introduction to Cryptography

- Basic Security Concepts
  - Confidentiality, integrity, availability
- Introduction to Cryptography
  - Secret key cryptography
    - Sender and receiver share the same key
    - Applications
      - Communication over insecure channel, Secure storage, Authentication, Integrity check

Introduction to Cryptography

- Introduction to Cryptography
  - Public key cryptography
    - Public key: publicly known
    - Private key: kept secret by owner
    - Encryption/decryption mode
      - How the keys are used?
    - Digital signature mode
      - How the keys are used?
    - Application: Secure communication, secure storage, authentication, digital signature, key exchange

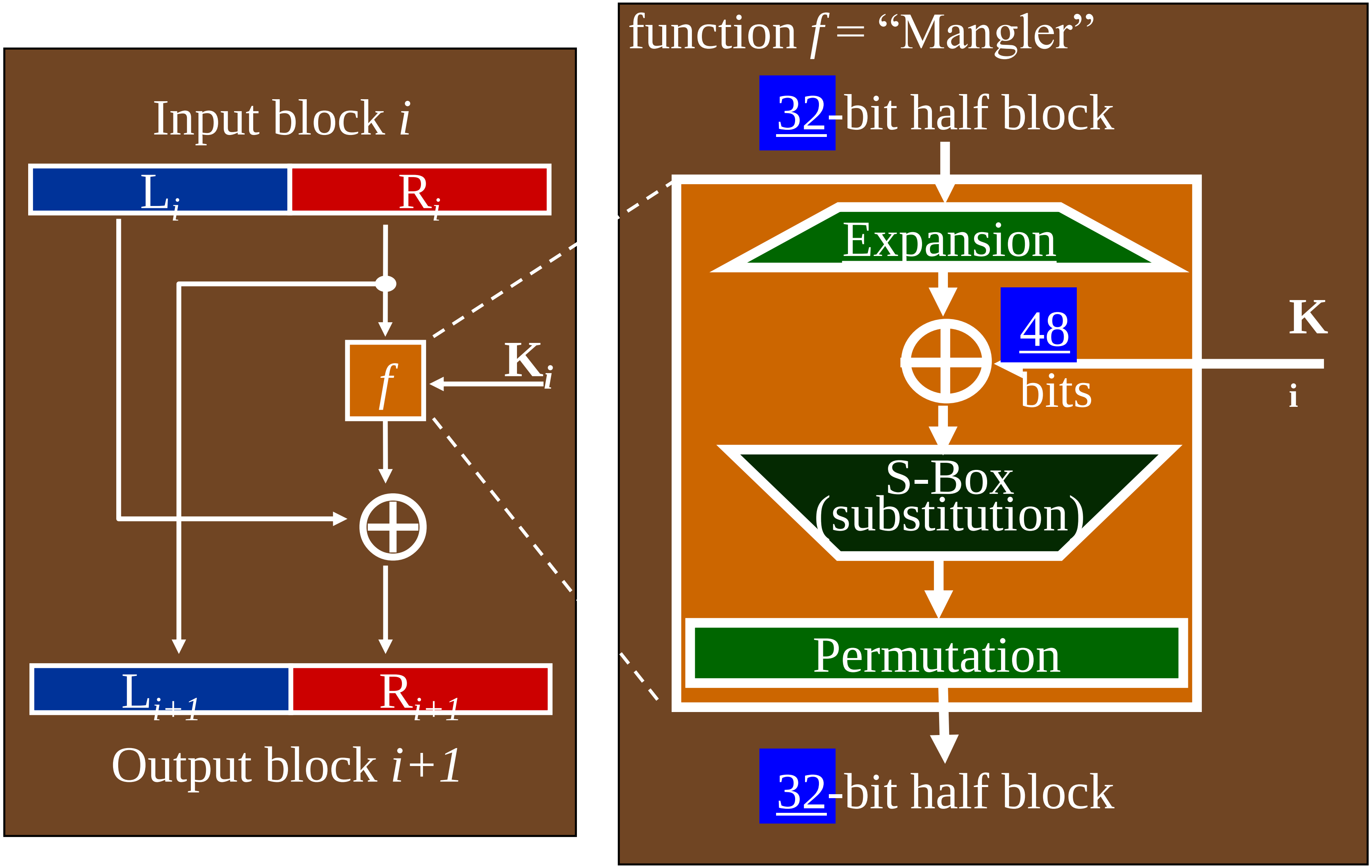
Introduction to Cryptography

- Introduction to Cryptography
  - Hash function
    - Map a message of arbitrary length to a fixed-length short message
  - Desirable properties
    - Performance, one-way, weak collision free, strong collision free

DES

- DES
  - Parameters
    - Block size (input/output 64 bits)
    - key size (56 bits)
    - number of rounds (16 rounds)
    - subkey generalization algorithm
    - round function

DES Round: *f* (Mangler) Function



Modes of Block Cipher Operations

- ECB (Electronic Code Book)
- CBC (Cipher Block Chaining Mode)
- OFB (Output Feedback Mode)
- CFB (Cipher Feedback Mode)

Modes of Block Cipher Operations

- Properties of Each Mode
  - Chaining dependencies
  - Error propagation
  - Error recovery

Double DES and Triple DES

- You need to understand how double and triple DES works
  - Double DES C=Ek2(Ek1(P))
  - Triple DES C = Ek1(Dk2(Ek1(P))
  - Meet-in-the-middle attacks
  - Operation modes using Triple DES

The Meet-in-the-Middle Attack

1. Choose a plaintext P and generate ciphertext C, using double-DES with $\mathcal{K}1+\mathcal{K}2$
2. Then…
   a. encrypt P using single-DES for all possible 2^{56} values K_1 to generate all possible single-DES ciphertexts for P: $X_1, X_2, \ldots, X_{2^{56}}$;
      store these in a table indexed by ciphertex values
   b. decrypt C using single-DES for all possible 2^{56} values K_2 to generate all possible single-DES plaintexts for C: $Y_1, Y_2, \ldots, Y_{2^{56}}$;
      for each value, check the table

Steps … (Cont'd)

3. Meet-in-the-middle:
   - Each match ($X_i = Y_j$) reveals a *candidate key pair* K_i+K_j
   - There are 2^{112} pairs but there are only 2^{64} X's
4. On average, how many pairs have identical X and Y?
   - For any pair (X, Y), the probability that X = Y is $1/2^{64}$
   - There are 2^{112} pairs.
   - The average number of pairs that result in identical X and Y is $2^{112} / 2^{64} = 2^{48}$

Steps … (Cont'd)

5. The attacker uses a second pair of plaintext and ciphertext to try the 2^{48} Key pairs

– There are 2^{48} pairs and there are 2^{64} X's (Y's)

— The average number of pairs that result in identical X and Y is $2^{48} / 2^{64} = 2^{-16}$

— The expected number of survived candidate key pairs is less than 1. After examine two pairs of plaintext and ciphtertext, the attacker identifies the key

Number Theory Summary

- Fermat: If *p* is prime and *a* is positive integer not divisible by *p*, then $a^{p-1} \equiv 1 \pmod{p}$

Example: 11 is prime, 3 not divisible by 11, so $3^{11-1} = 59049 \equiv 1 \pmod{11}$

Euler: For every *a* and *n* that are relatively prime, then $a^{ø(n)} \equiv 1 \bmod n$

Example: For a = 3, n = 10, which relatively prime: $\phi(10) = 4$, $3^{\phi(10)} = 3^4 = 81 \equiv 1 \bmod 10$

Variant: for all a in $\mathcal{Z}_n^*$, and all non-negative *k*, $a^{k\phi(n)+1} \equiv a \bmod n$

Example: for n = 20, a = 7, $\phi(n) = 8$, and k = 3: $7^{3*8+1} \equiv 7 \bmod 20$

Generalized Euler's Theorem: for $n = pq$ (*p* and *q* are distinct primes), all *a* in $\mathcal{Z}_n$, and all non-negative *k*, $a^{k\phi(n)+1} \equiv a \bmod n$

Example: for n = 15, a = 6, $\phi(n) = 8$, and k = 3: $6^{3*8+1} \equiv 6 \bmod 15$

$x^y \bmod n = x^{y \bmod \phi(n)} \bmod n$ (foundation for RSA public key cryptographic)

Example: x = 5, y = 7, n = 6, $\phi(6) = 2$, $5^7 \bmod 6 = 5^{7 \bmod 2} \bmod 6 = 5 \bmod 6$

Public Key Cryptography

- RSA Algorithm
  - Basis: factorization of large numbers is hard
  - Variable key length (1024 bits or greater)
  - Variable plaintext block size
    - plaintext block size must be smaller than key size
    - ciphertext block size is same as key size

Generating a Public/Private Key Pair

- Find large primes p and q
- Let $n = p*q$
  - do not disclose p and q!
  - $\phi(n) = (p-1)*(q-1)$
- Choose an e that is relatively prime to $\phi(n)$
  - **public** key = <e,n>
- Find d = multiplicative inverse of e mod $\phi(n)$ (i.e., $e*d = 1 \bmod \phi(n)$)
  - **private** key = <d,n>

RSA Operations

- For plaintext message ***m*** and ciphertext ***c***

Encryption: $\boldsymbol{c = m^{e} \bmod n}$, $m < n$

Decryption: $\boldsymbol{m = c^{d} \bmod n}$

Signing: $\boldsymbol{s = m^{d} \bmod n}$, $m < n$

Verification: $\boldsymbol{m = s^{e} \bmod n}$

Diffie-Hellman Protocol

- For negotiating a shared secret key using only public communication
- Does not provide authentication of communicating parties
- What's involved?
  - *p* is a large prime number (about 512 bits)
  - *g* is a primitive root of *p*, and $g < p$
  - *p* and *g* are publicly known

D-H Key Exchange Protocol

Alice	Bob
Publishes g and p	Reads g and p
Picks random number S_A *(and keeps private)*	Picks random number S_B *(and keeps private)*
Computes $T_A = g^{S_A} \bmod p$	Computes $T_B = g^{S_B} \bmod p$
Sends T_A to Bob,	Sends T_B to Alice,
Computes $T_B^{S_A} \bmod p$ =	Computes $T_A^{S_B} \bmod p$

Key Exchange (Cont'd)

Alice and Bob have now both computed the same secret $g^{S_A S_B}$ mod p, which can then be used as the shared secret key K

S_A is the discrete logarithm of g^{S_A} mod p and

S_B is the discrete logarithm of g^{S_B} mod p

Why is This Secure?

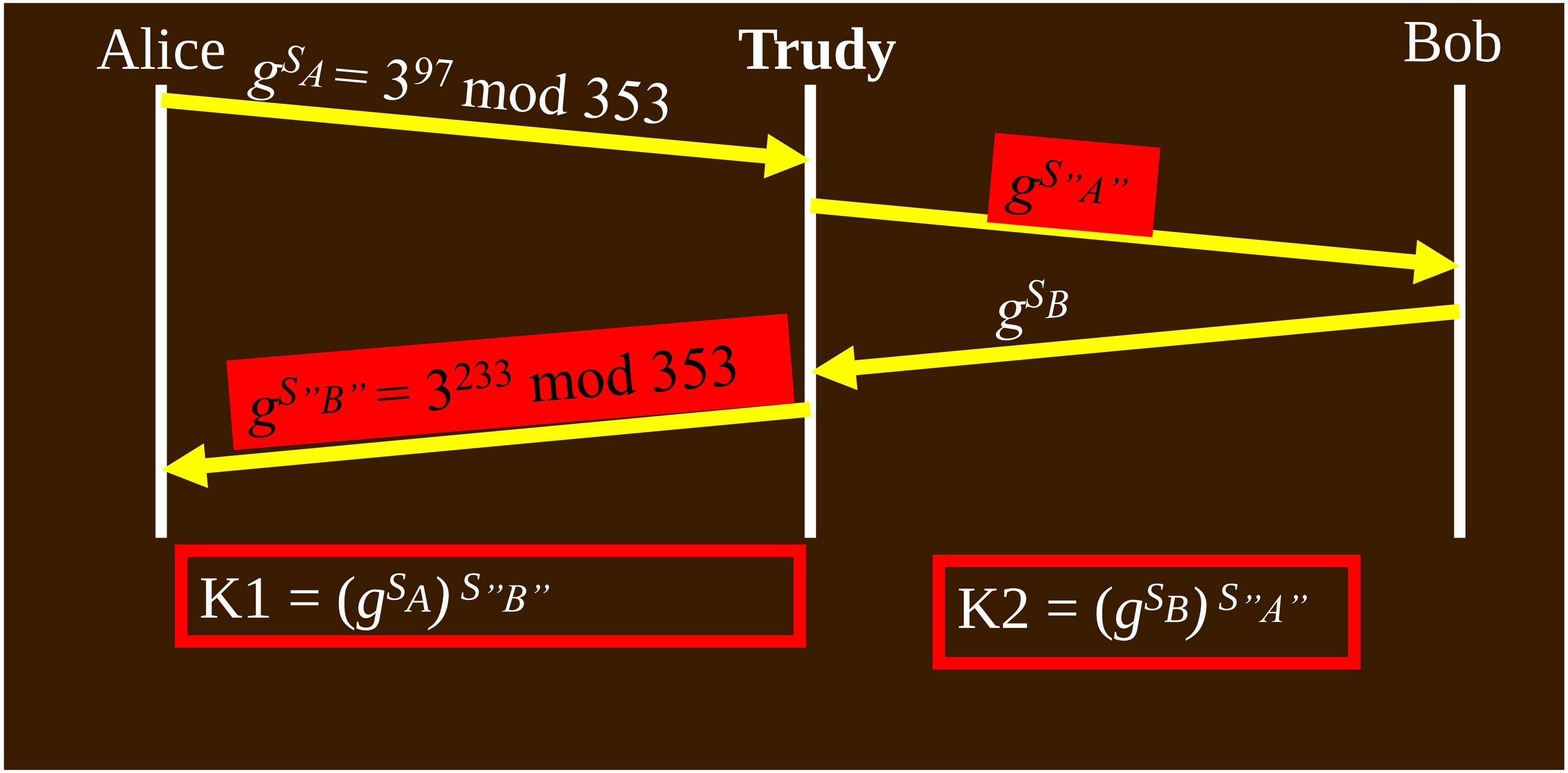
- Discrete log problem is hard:
  - given $a^x \bmod b$, a, and b, it is computationally infeasible to compute x

D-H Limitations

- Expensive exponential operation is required
  - possible timing attacks??
- Algorithm is useful for key negotiation only
  - i.e., not for public key encryption/verification
- Not for user authentication
  - In fact, you can negotiate a key with a complete stranger!

Man-In-The-Middle Attack

- Trudy impersonates as Alice to Bob, and also impersonates as Bob to Alice



Authenticating D-H Messages

- That is, you know who you're negotiating with, and that the messages haven't been modified
- Requires that communicating parties already share something
- Then use shared information to enable authentication

Using D-H in "Phone Book" Mode

1. Alice and Bob each chooses a secret number, generate T_A and T_B
2. Alice and Bob *publish* T_A, $T_{B,}$ i.e., Alice can get Bob's T_B at any time, Bob can get Alice's T_A at any time
3. Alice and Bob can then generate a shared key without communicating
   – but, they must be using the same *p* and *g*

- Essential requirement: reliability of the published values (no one can substitute false values)

Digital Signature Standard (DSS)

- Useful only for digital signing (no encryption or key exchange)
- Components
  - SHA-1 to generate a hash value (some other hash functions also allowed now)
  - Digital Signature Algorithm (DSA) to generate the digital signature from this hash value
- Designed to be fast for the signer rather than verifier

Digital Signature Algorithm (DSA)

1. Announce public parameters used for signing
   - pick p (a prime with >= 1024 bits)
     ex.: $p = 103$
   - pick q (a 160 bit prime) such that $q \mid (p-1)$
     ex.: $q = 17$ (divides 102)
   - choose $g \equiv h^{(p-1)/q} \bmod p$, where $1 < h < (p-1)$, such that $g > 1$
     ex.: if $h = 2$, $g = 2^6 \bmod 103 = 64$
   - note: g is of order q mod p
     ex.: powers of 64 mod 103 =
     64 79 9 61 93 81 34 13 8 100 14 72 76 23 30 66 1

     17 values

DSA (Cont'd)

2. User Alice generates a long-term private key x
   - random integer with $0 < x < q$

ex.: $x = 13$

3. Alice generates a long-term public key y
   - $y = g^x \bmod p$

ex.: $y = 64^{13} \bmod 103 = 76$

DSA (Cont'd)

4. Alice randomly picks a per message secret number k such that $0 < k < q$, and generates $k^{-1} \bmod q$

ex.: k = 12, $12^{-1} \bmod 17 = 10$

5. Signing message M

ex.: H(M) = 75

   – $r = (g^k \bmod p) \bmod q$

ex.: $r = (64^{12} \bmod 103) \bmod 17 = 4$

   – $s = [k^{-1} * (H(M) + x * r)] \bmod q$

ex.: s = [10 * (75 + 13*4)] mod 17 = 12

   – transmitted info = M, r, s

ex.: M, 4, 12

Verifying a DSA Signature

- Known : g, p, q, y

ex.: p = 103, q = 17, g = 64, y = 76, H(M) = 75

- Received from signer: M, r, s

ex.: M, **4**, 12

1. $w = (s)^{-1} \bmod q$

ex.: $w = 12^{-1} \bmod 17 = 10$

2. $u_1 = [H(M) * w] \bmod q$

ex.: $u_1 = 75*10 \bmod 17 = 2$

3. $u_2 = (r*w) \bmod q$

ex.: $u_2 = 4*10 \bmod 17 = 6$

4. $v = [(g^{u1} * y^{u2}) \bmod p] \bmod q$

ex.: $v = [(64^2 * 76^6) \bmod 103] \bmod 17 = \mathbf{4}$

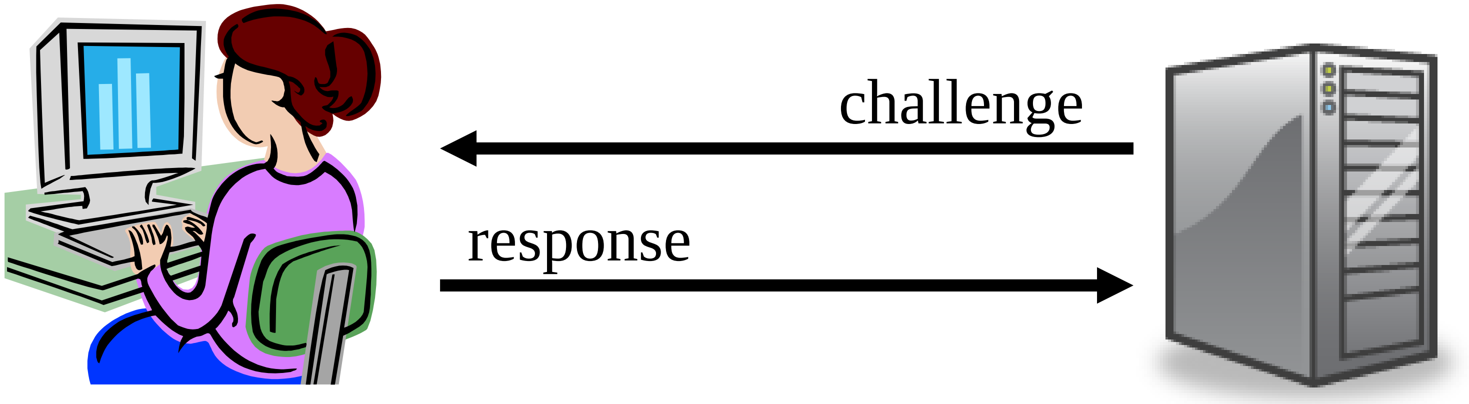
5. If $v = r$, then the signature is verified

Authentication

- Authentication is the process of reliably verifying certain information.
- Examples
  – User authentication
    - Allow a user to prove his/her identity to another entity (e.g., a system, a device).
  – Message authentication
    - Verify that a message has not been altered without proper authorization.

Password-Based User Authentication

- User demonstrates knowledge of a secret value to authenticate
  - most common method of user authentication



Password Storage

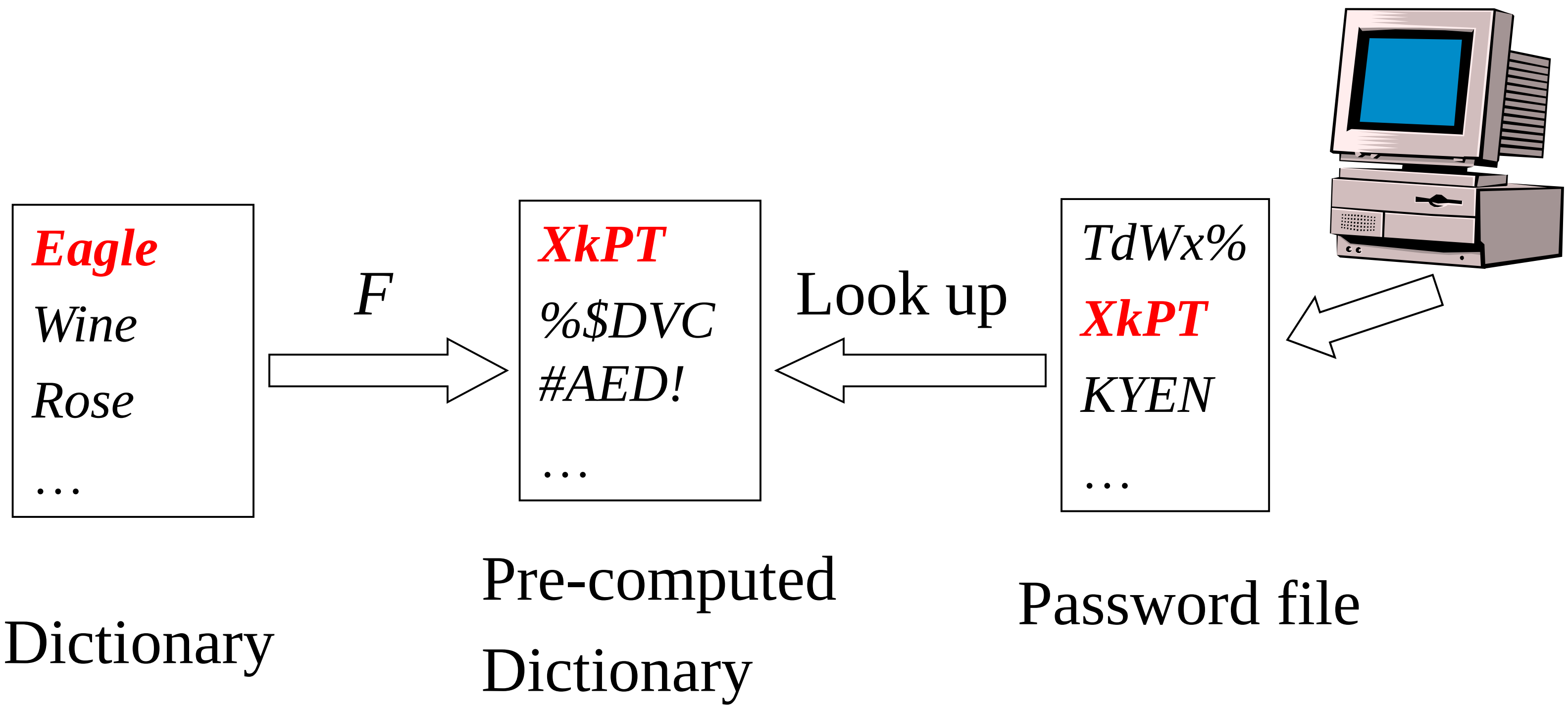
- Storing unencrypted passwords in a file is high risk
  - compromising the file system compromises all the stored passwords
- Better idea: use the password to compute a one-way function (e.g., a hash, an encryption), and store the output of the one-way function
- When a user inputs the requested password…
  1. compute its one-way function
  2. compare with the stored value

Common Password Choices

- Pet names
- Common names
- Common words
- Dates
- Variations of above (backwards, append a few digits, etc.)

Dictionary Attacks (Cont'd)

- Attack 3 (offline):
  - To speed up search, pre-compute *F*(dictionary)
  - A simple look up gives the password

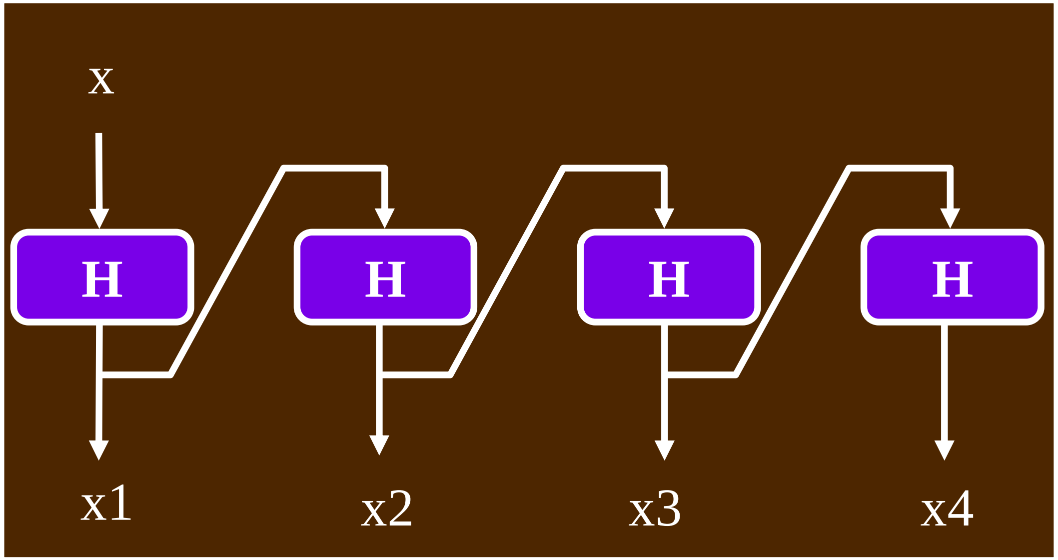


Password Salt

- To make the dictionary attack a bit more difficult
- Salt is a n-bit number between 0 and 2^n
- Derived from, for example, the system clock and the process identifier

S/Key Password Generation

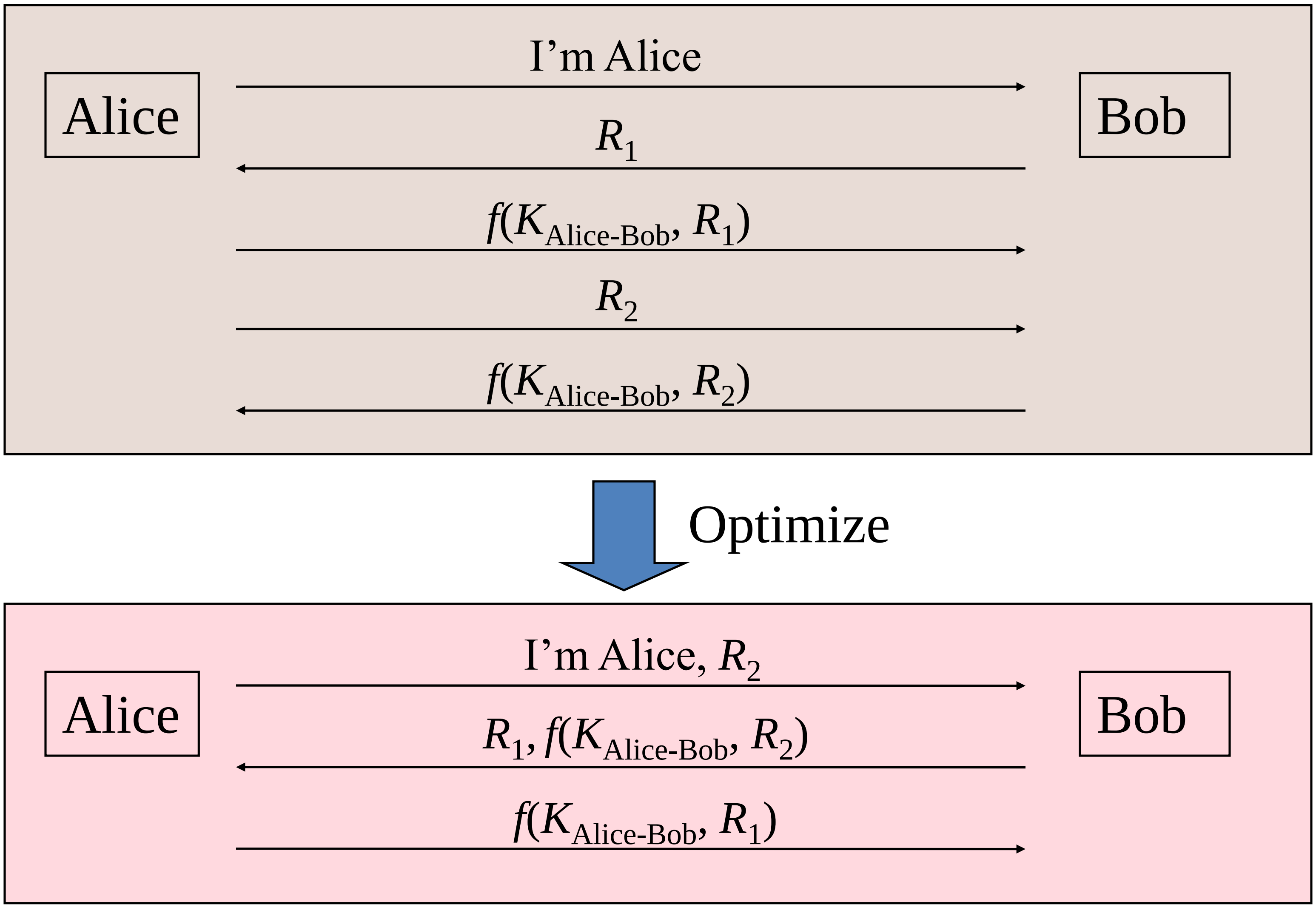
1. Alice selects a password **x**
2. Alice specifies *n*, the number of passwords to generate
3. Alice's computer then generates a sequence of passwords
   - $x_1 = H(\mathbf{x})$
   - $x_2 = H(x_1)$
   - …
   - $x_n = H(x_{n-1})$



Authentication Handshakes

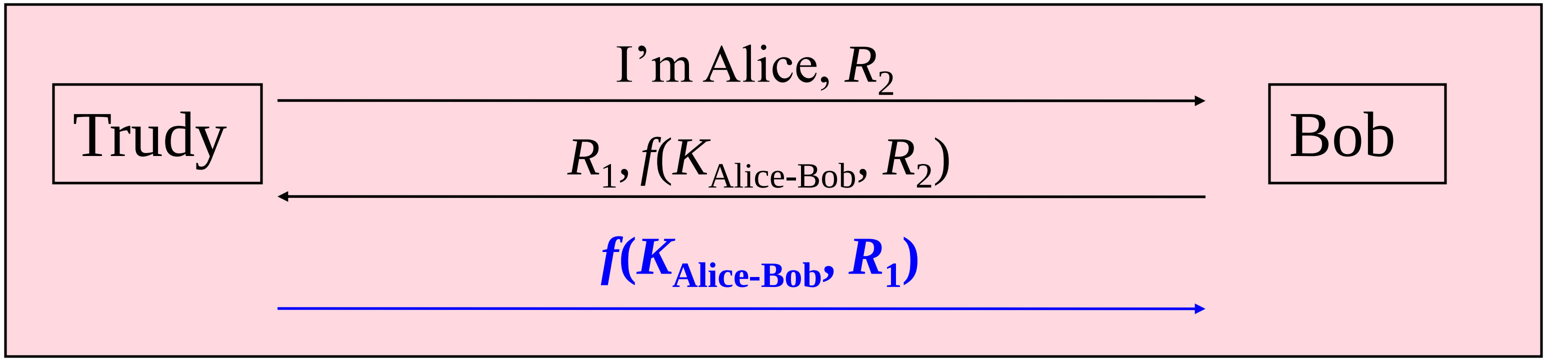
- Secure communication almost always includes an initial authentication handshake.
  - Authenticate each other
  - Establish session keys
  - *This process is not trivial; flaws in this process undermine secure communication*

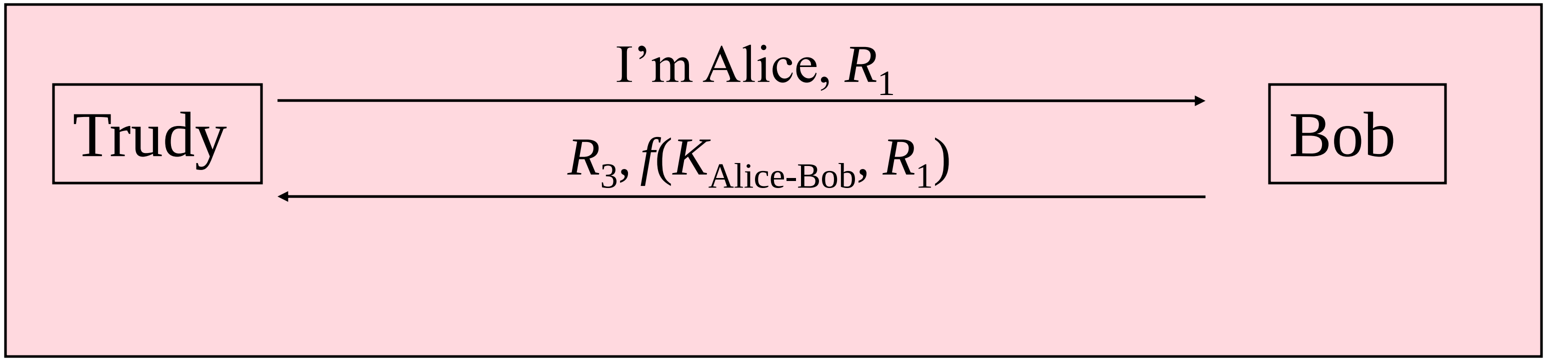
Mutual Authentication



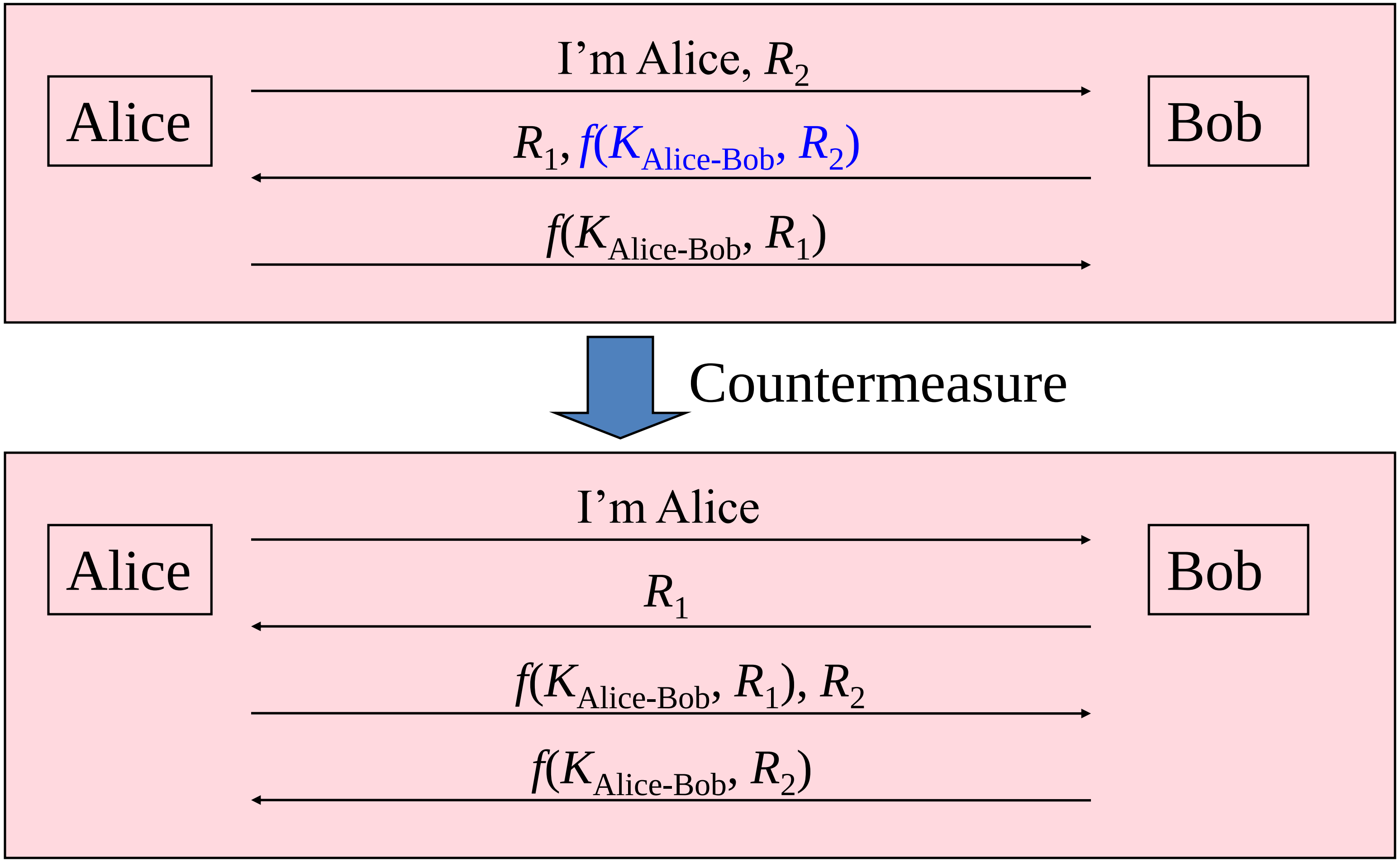
Mutual Authentication (Cont'd)

- Reflection attack





Mutual Authentication (Cont'd)

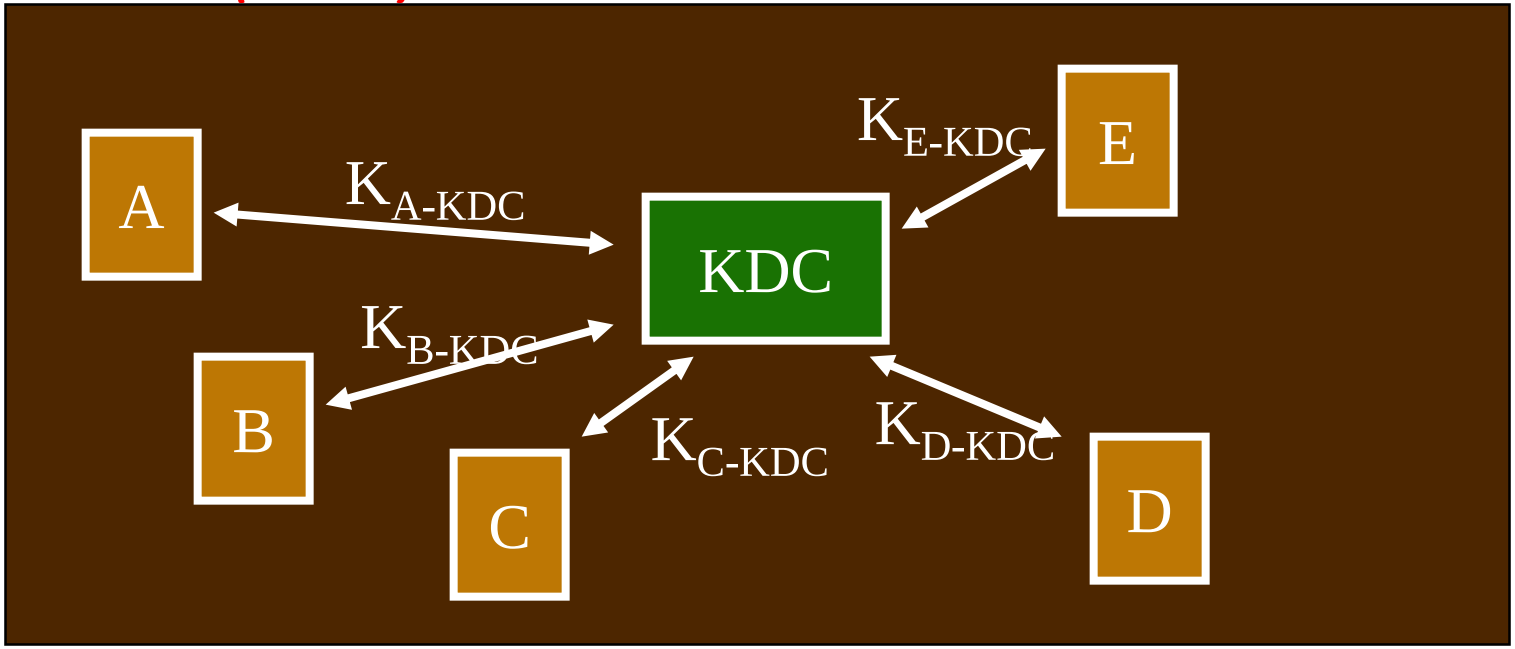


Trusted Key Servers

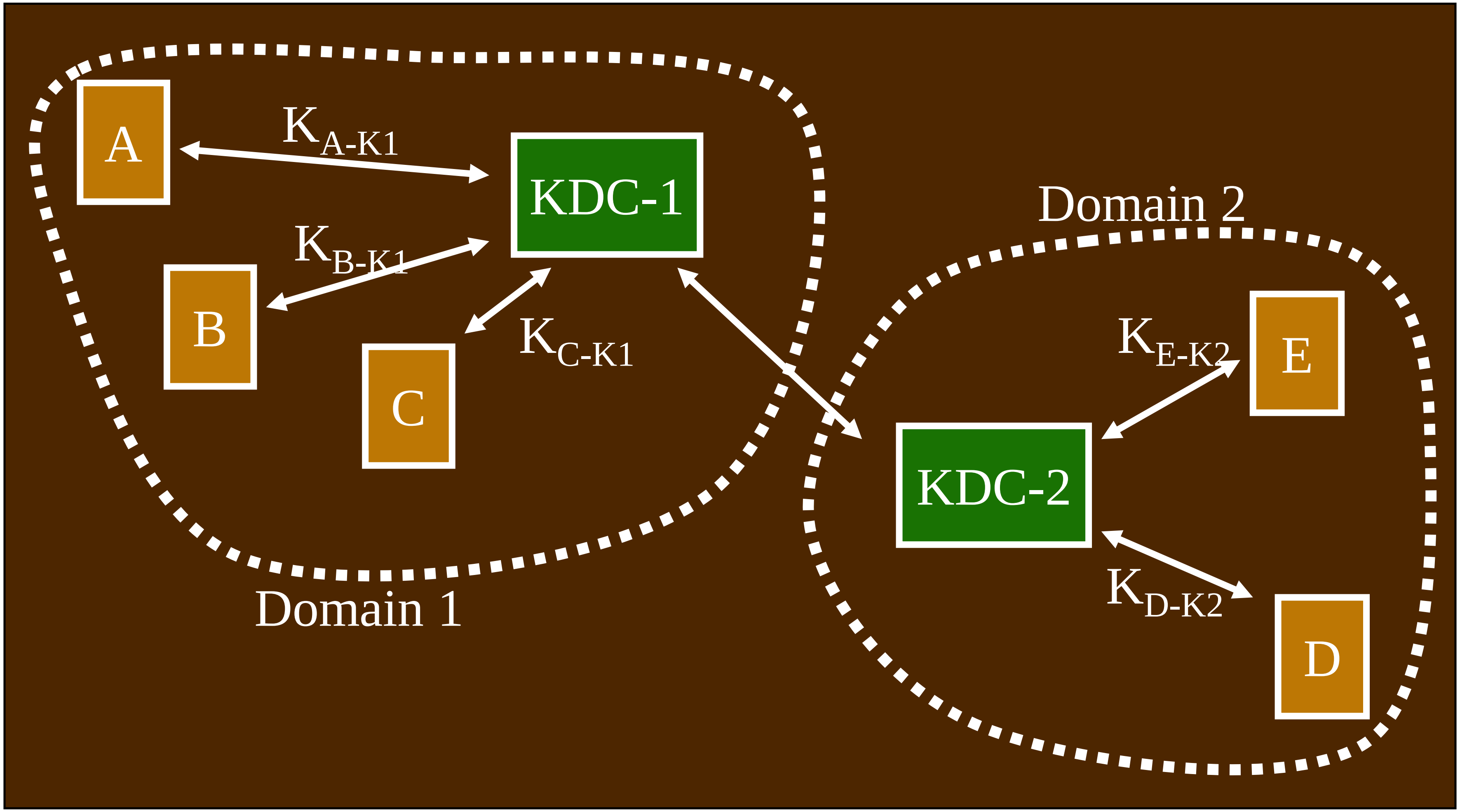
- How do a large number of users authenticate each other?
  - inefficient / impractical for every pair of users to negotiate a secret key or share passwords
- Alternative: everybody shares a key with (and authenticates to) a single trusted third party
- Assumes there is a way to negotiate a key with the *third party*

Trusted… (cont’d)

- Shared keys between the *Key Distribution Center (KDC)* and users

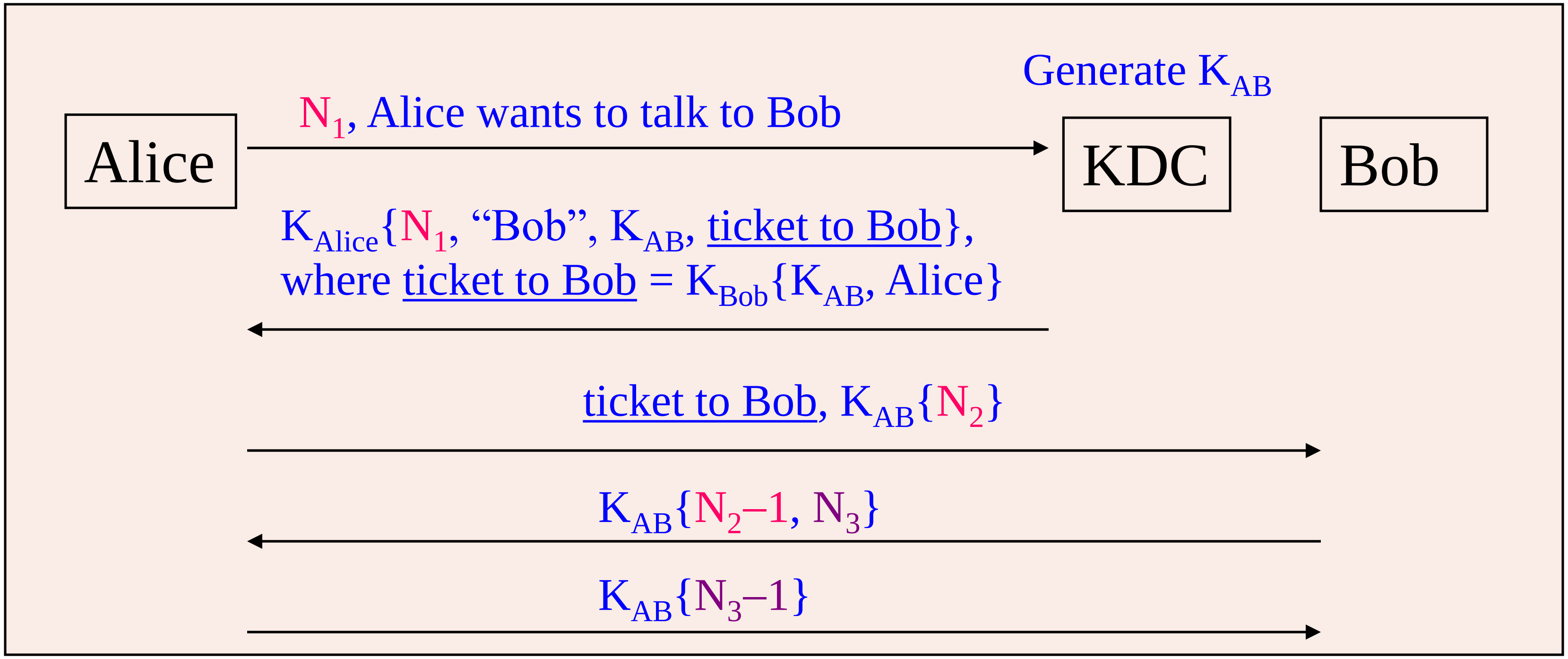


Hierarchy… (cont'd)



Needham-Schroeder Protocol

- Classic protocol for authentication with KDC
  - Many others have been modeled after it (e.g., Kerberos)

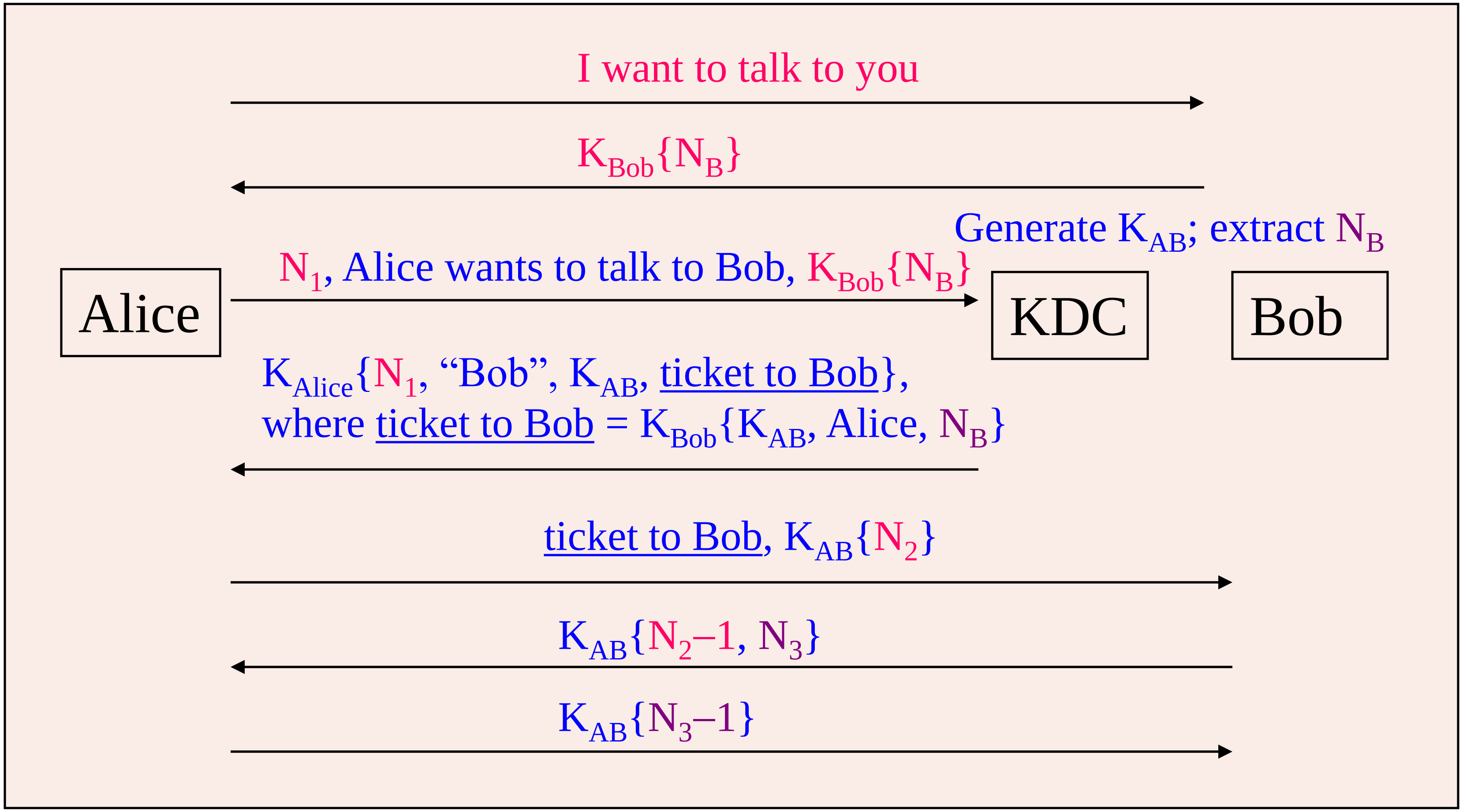


How is Bob authenticated? How is Alice authenticated? How is KDC authenticated? What are the N's used for? Why is N-1 needed?

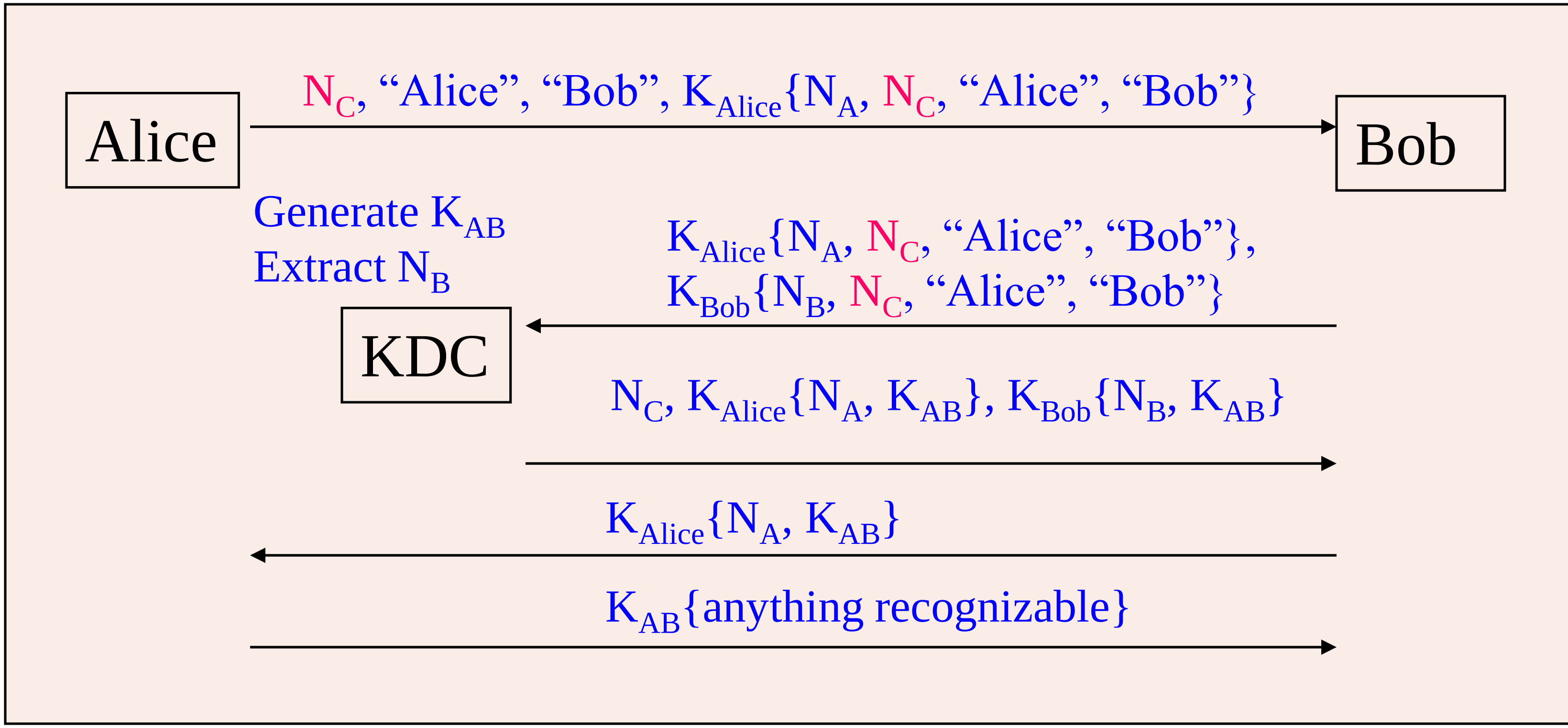
Needham-Schroeder Protocol (Cont'd)

- A vulnerability
  - When Trudy gets a previous key K_{AB} used by Alice, Trudy may reuse a previous ticket issued to Bob for Alice
  - Essential reason
    - The ticket to Bob stays valid even if Alice changes her key

Expanded Needham-Schroeder Protocol



Otway-Rees Protocol



- Only has five messages
- KDC checks if N_C matches in both cipher-texts
  - Make sure that Bob is really Bob

Trusted Intermediaries

- Problem: authentication for large networks
- Solution #1
  - Key Distribution Center (KDC)
    - Representative solution: Kerberos
  - Based on secret key cryptography
- Solution #2
  - Public Key Infrastructure (PKI)
  - Based on public key cryptography

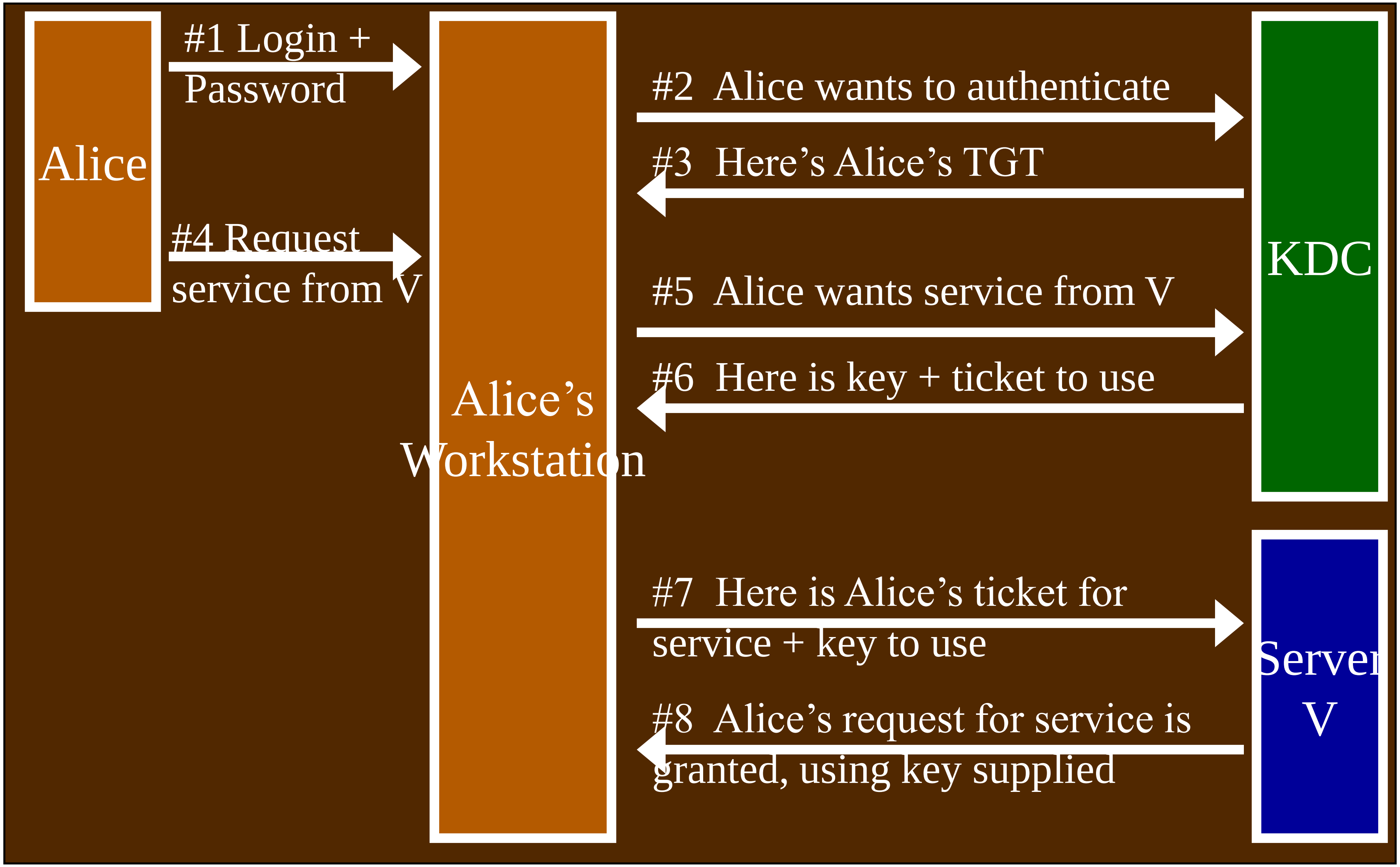
Goals of Kerberos

1. User $\leftrightarrow$ server mutual authentication
2. Users should only need to authenticate once to obtain services from multiple servers
3. Should scale to large numbers of users and servers
   - makes use of a Key Distribution Center so servers don’t need to store information about users

Some Properties

- Kerberos uses only secret key (symmetric) encryption
  - originally, only DES, but now 3DES and AES as well
- A *stateless* protocol
  - KDCs do not need to remember what messages have previously been generated or exchanged
  - the state of the protocol negotiation is contained in the message contents

Protocol Sketch (Common Case)



Some Differences with v4

1. v5 uses ASN.1 syntax to represent messages
   - a standardized syntax, not particularly easy to read
   - but, very flexible (optional fields, variable field lengths, extensible value sets, …)
2. v5 extends the set of encryption algorithms
3. v5 supports much longer ticket lifetimes
4. v5 allows "Pre-authentication" to thwart password attacks
5. v5 allows delegation of user access / rights

Delegation

- Giving someone else the right to access your services
- Some not-so-good ways to implement
  - give someone else your password / key
  - give someone else your tickets (TKT_V's)
- Kerberos v5 provides 3 better choices

Pre-Authentication

#3. KDC→W: $K_{A\text{-}KDC}$(ID_A | TS_1 | $Lifetime_1$ | $\mathcal{K}_{A\text{-}KDC}$ | ID_{KDC} | TGT)

- Reminder: Msg #3 is encrypted by the KDC with $\mathbf{K_{A\text{-}KDC}}$
  - An adversary may send many authentication requests to cause the Denial-of-Service.
- Solution: before Msg #3, require Alice to send *pre-authentication data* to the KDC
  - i.e., a timestamp encrypted with the shared master key
  - this proves Alice knows the key

Pre-Authentication (Cont'd)

$K_{V\text{-}KDC}(ID_A \mid Addr_A \mid \mathcal{K}_{\mathcal{A}\text{-}\mathcal{V}} \mid Lifetime_5 \mid TS_5 \mid ID_V)$

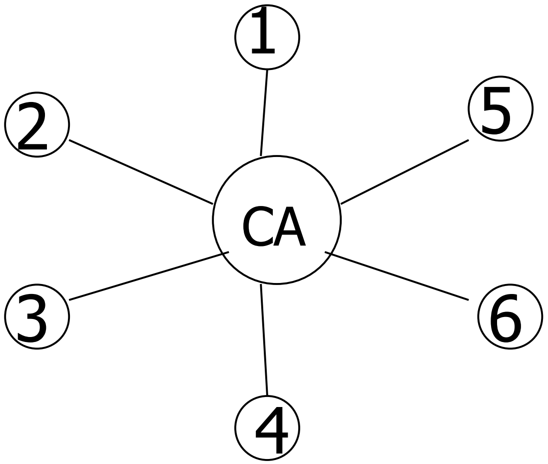
- Msg#6 provides an opportunity for Alice to mount a password-guessing attack against the server key $\mathbf{K_{V\text{-}KDC}}$
  - solution: servers are not allowed to generate keys based on (weak) passwords

What Is PKI

- Informally, the infrastructure supporting the use of public key cryptography.
- A PKI consists of
  - Certificate Authority (CA)
  - Certificates
  - A repository for retrieving certificates
  - A method of revoking/updating certificates

Certification Authorities (CA)

- A CA is a trusted node that maintains the public keys for all nodes (Each node maintains its own private key)



If a new node is inserted in the network, only that new node and the CA need to be configured with the public key for that node

Certificates

- A CA is involved in authenticating users' public keys by generating certificates
- A certificate is a signed message vouching that a particular name goes with a particular public key
- Example:
  1. [Alice's public key is 876234]$_{carol}$
  2. [Carol's public key is 676554]$_{Ted}$ & [Alice's public key is 876234]$_{carol}$
- Knowing the CA's public key, users can verify the certificate and authenticate Alice's public key

Certificates

- Certificates can hold expiration date and time
- Alice keeps the same certificate as long as she has the same public key and the certificate does not expire
- Alice can append the certificate to her messages so that others know for sure her public key

CA Advantages

1. The CA does not need to be online. [Why?]

2. If a CA crashes, then nodes that already have their certificates can still operate.

3. Certificates are not security sensitive (in terms of confidentiality).
   - Can a compromised CA decrypt a conversation between two parties?
   - Can a compromised CA fool Alice into accepting an incorrect public key for Bob, and then impersonate Bob to Alice?

PKI Models

1. Monopoly model
2. Monopoly + RA
3. Delegated CAs
4. Oligarchy model
5. Anarchy model
6. Name constraints
7. Top-down with name constraints
8. Bottom-up with name constraints

Certificate Revocation

- Certificates for public keys (Campus IDs) might need to be revoked from the system
  - Someone is fired
  - Someone is graduated
  - Someone’s certificate (card) is stolen

Certificate Revocation

- Certificates typically have an associated expiration time
  - Typically in the order of months (too long to wait if it needs to be revoked)

- Solutions:
  - Maintain a Certificate Revocation List (CRL)
  - A CRL is issued periodically by the CA and contains all the revoked certificates
  - Each transaction is checked against the CRL

CRLs

1. Why are CRLs issued periodically even if no certificates are revoked?

2. How frequent should CRLs be issued?

3. If a CRL is maintained, why associate an expiration time with certificates?

Delta CRL

- A Delta CRL includes lists changes from the last complete CRL
- Delta CRLs may be issued periodically (frequently) and full CRLs are issued less frequently

Good-lists vs. Bad-lists

- How about maintaining a list of valid certificates in the CRL instead of the revoked certificates?
- Is this more secure? Why?

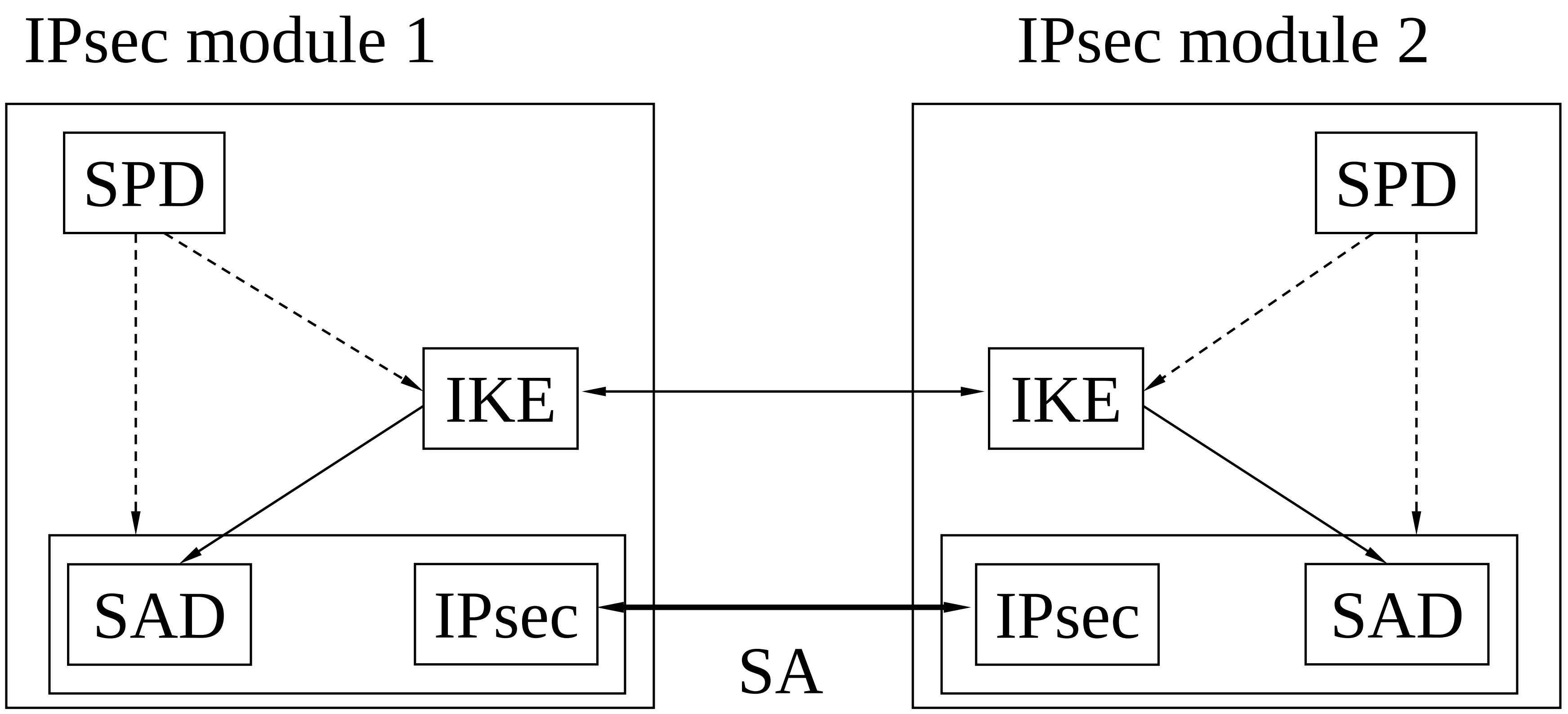
- Problems:
  1. A good list is likely to be much larger than the bad list (worse performance)
  2. Organizations might not want to maintain its list of valid certificates public.

  Solution: The good-list can maintain only hashes of the valid certificates

IPsec Objectives (Cont'd)

- IP layer security mechanism for IPv4 and IPv6
  - Not all applications need to be security aware
  - Can be transparent to users
  - Provide authentication and confidentiality mechanisms.

IPsec Architecture

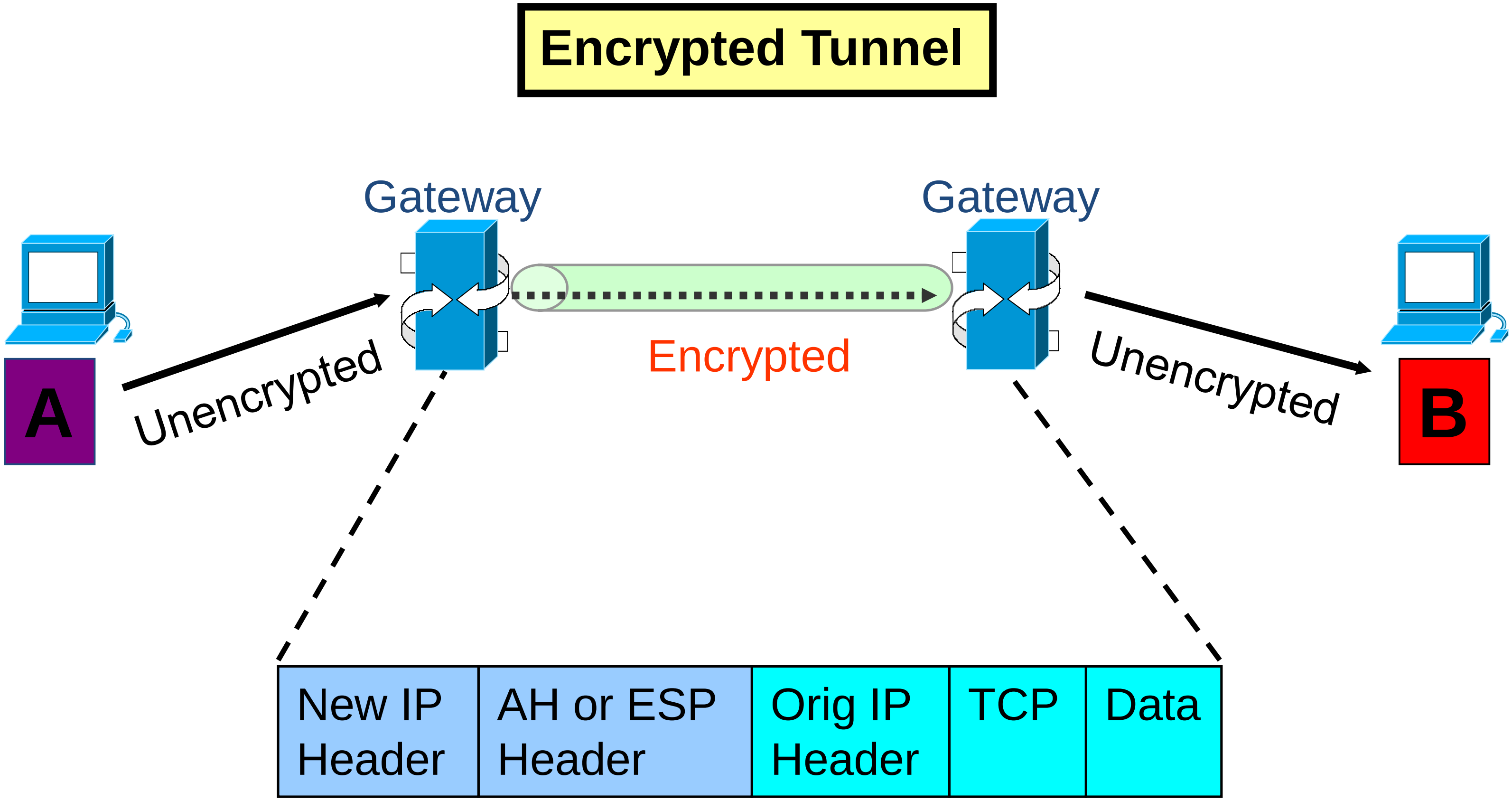


SPD: *Security Policy Database*; IKE: *Internet Key Exchange*; SA: *Security Association*; SAD: *Security Association Database*.

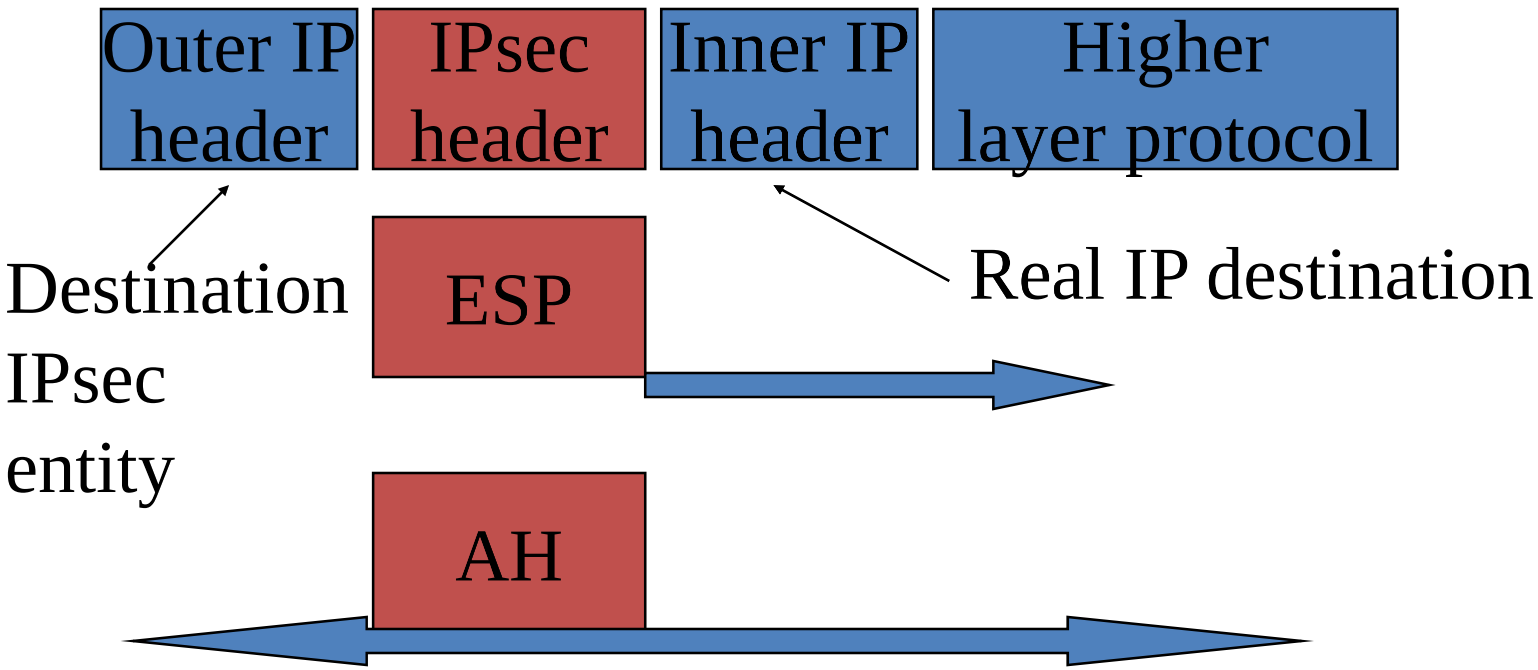
IPsec Architecture (Cont'd)

- Two Protocols (Mechanisms)
  - Authentication Header (AH)
  - Encapsulating Security Payload (ESP)
- IKE Protocol
  - Internet Key Management

Tunnel Mode

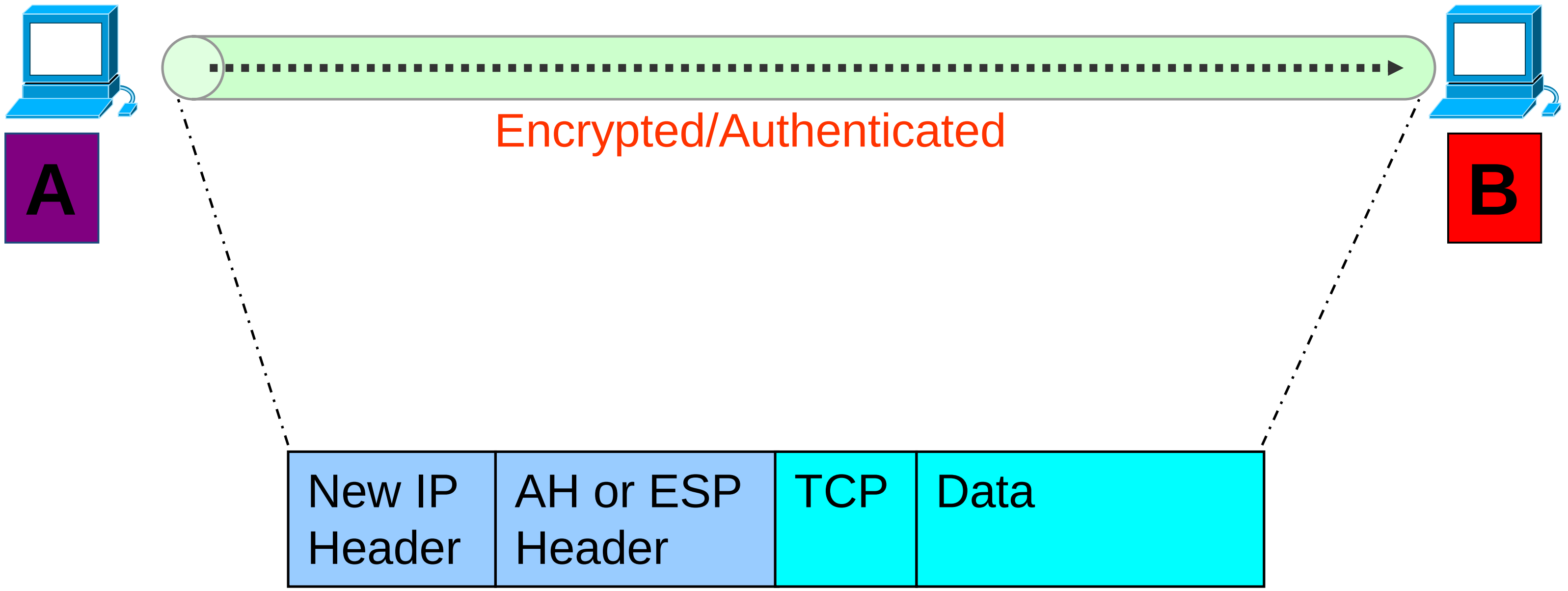

Encrypted Tunnel
Gateway
Gateway
A
Unencrypted
Encrypted
Unencrypted
B
New IP Header
AH or ESP Header
Orig IP Header
TCP
Data

Tunnel Mode (Cont'd)

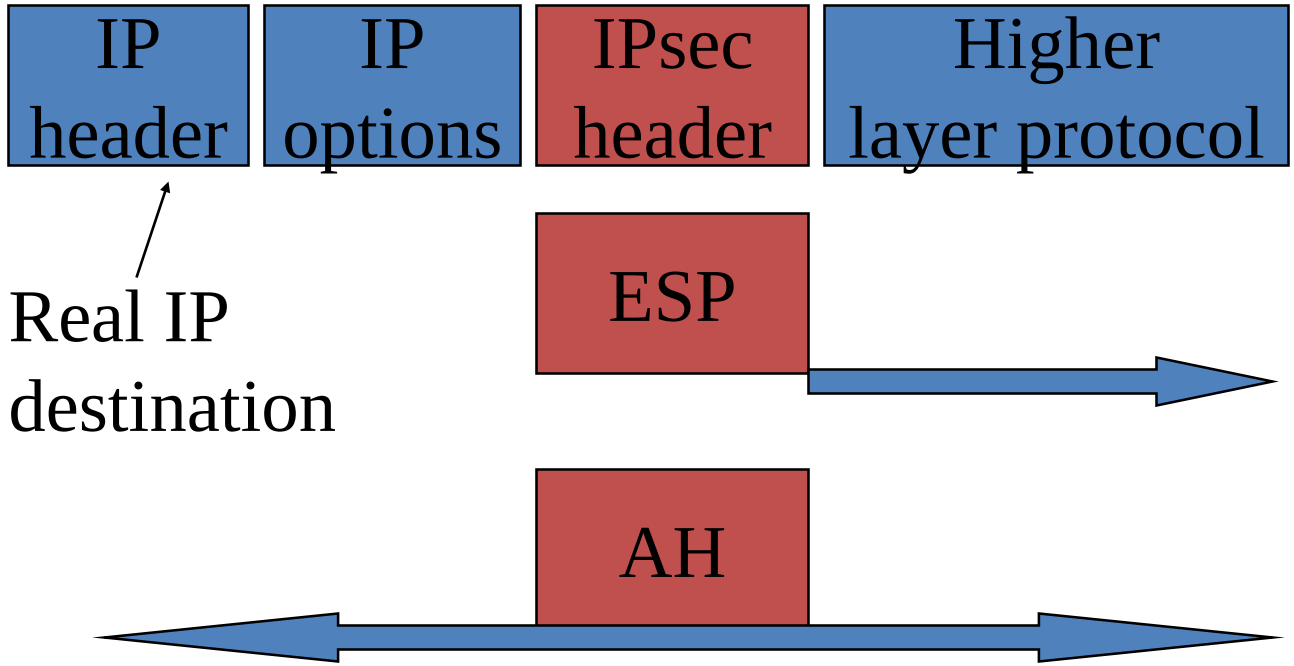


- ESP applies only to the tunneled packet
- AH can be applied to portions of the outer header

Transport Mode



Transport Mode (Cont’d)



- ESP protects higher layer payload only
- AH can protect IP headers as well as higher layer payload

Security Association (SA)

- An association between a sender and a receiver
  - Consists of a set of security related parameters
  - E.g., sequence number, encryption key
- Determine IPsec processing for senders
- Determine IPsec decoding for destination
- SAs are not fixed! Generated and customized per traffic flows

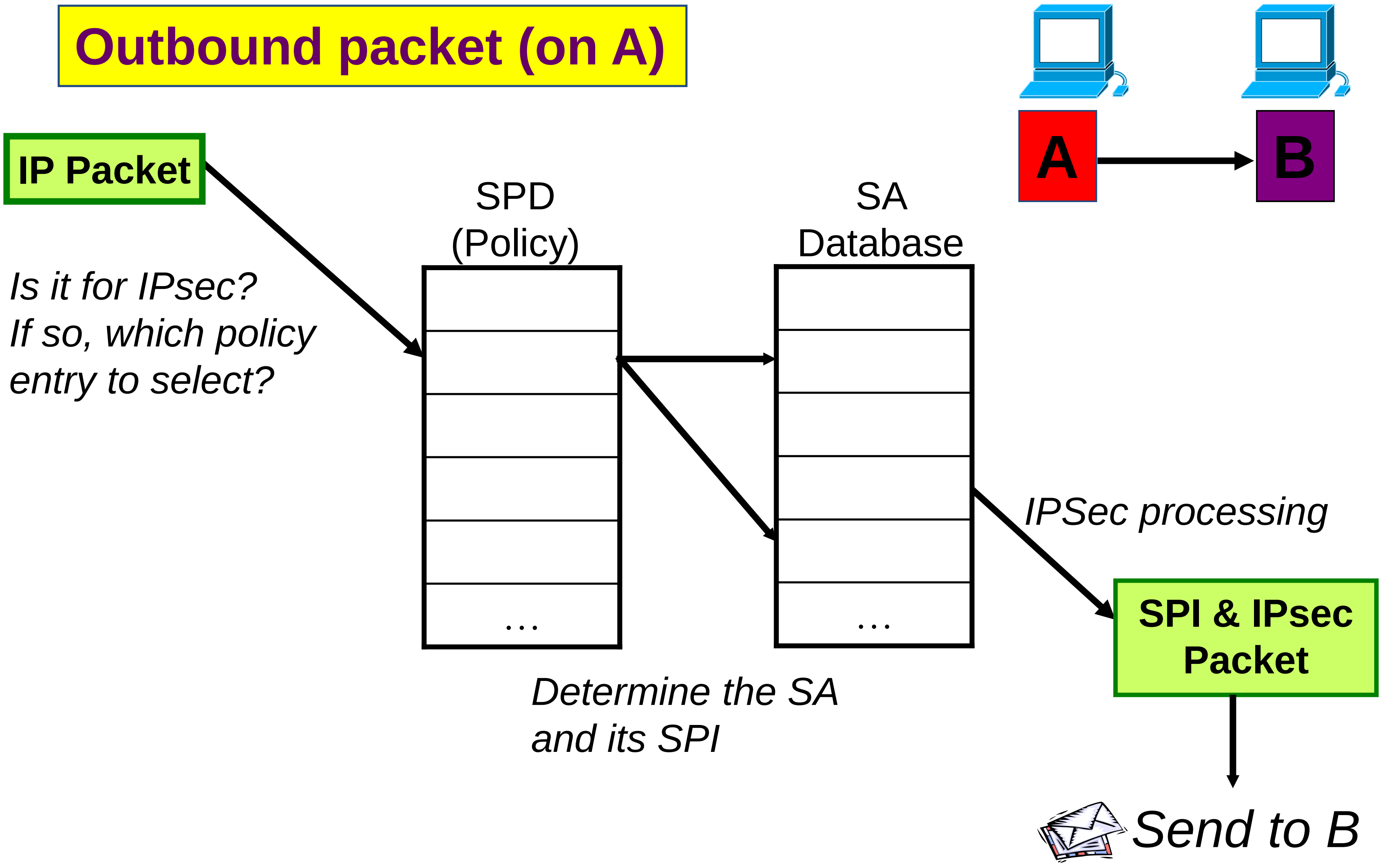
Security Parameters Index (SPI)

- A bit string assigned to an SA.
- Carried in AH and ESP headers to enable the receiving system to select the SA under which the packet will be processed.
- 32 bits
- SPI + Dest IP address + IPsec Protocol
  - Uniquely identifies each SA in SA Database (SAD)

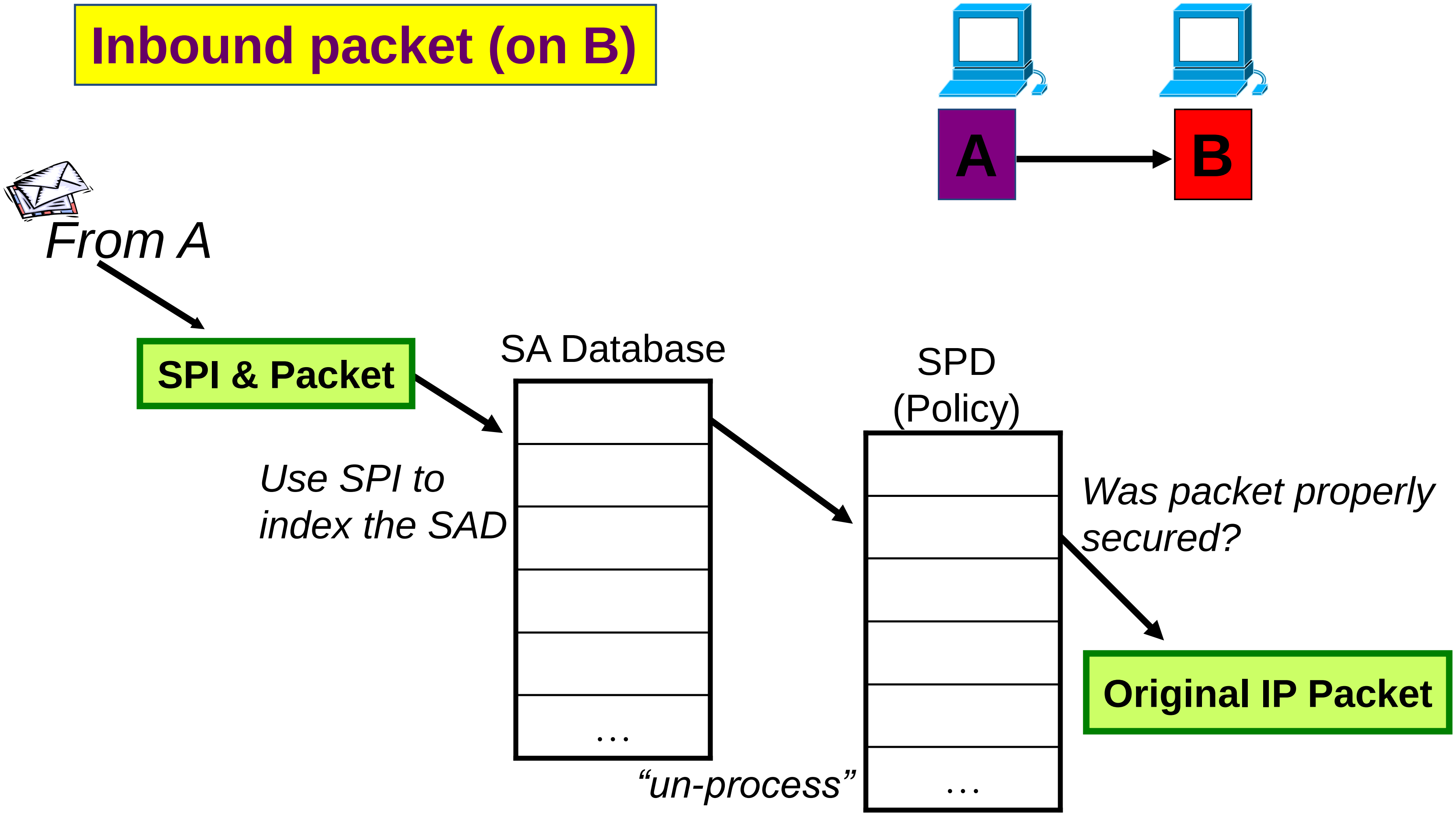
Security Policy Database (SPD)

- Policy entries define which SA or SA Bundles to use on IP traffic
- Each host or gateway has their own SPD
- Index into SPD by Selector fields
  - Selectors: IP and upper-layer protocol field values.
  - Examples: Dest IP, Source IP, Transport Protocol, IPSec Protocol, Source & Dest Ports, …

Outbound Processing



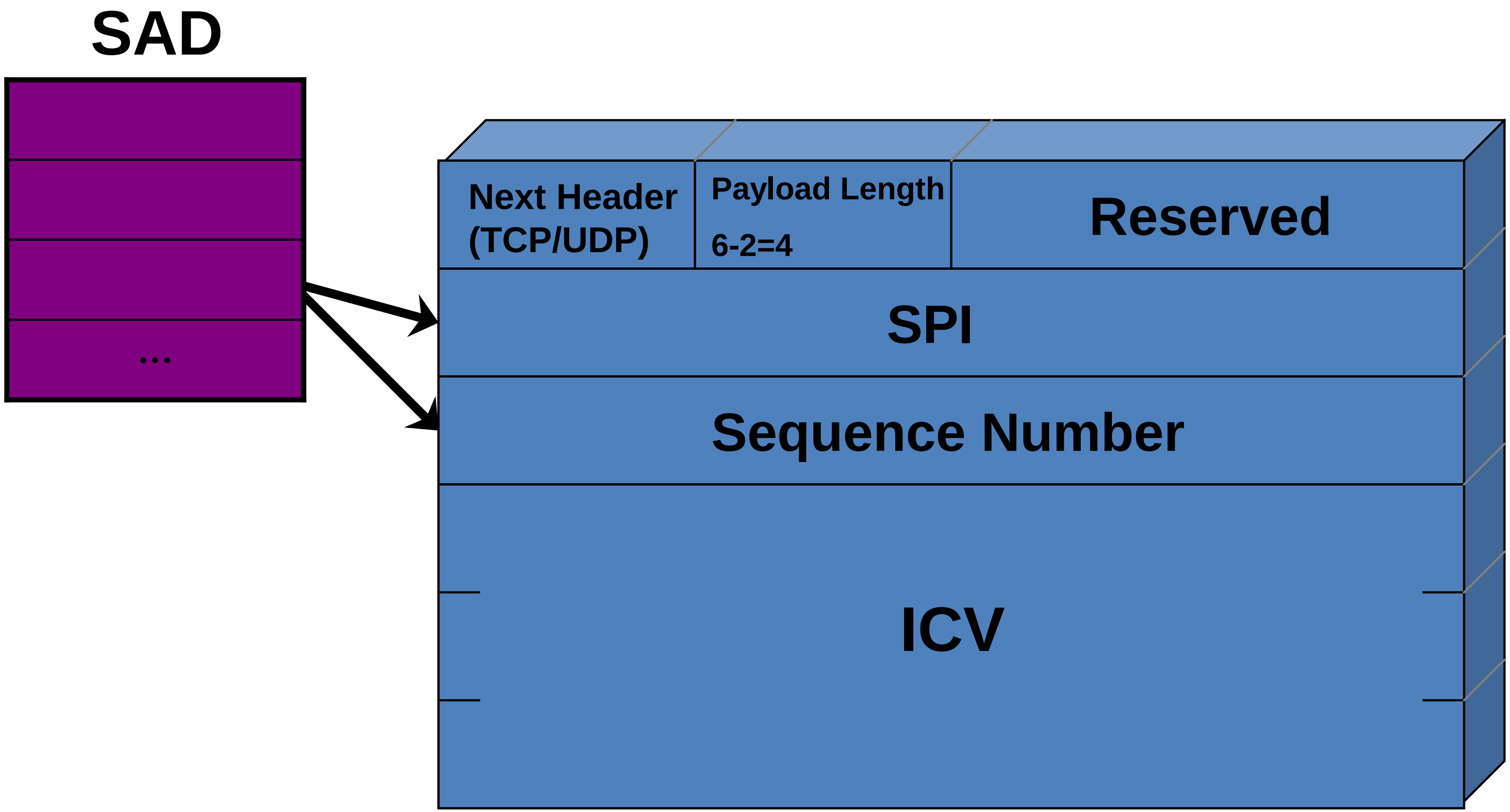
Inbound Processing



Authentication Header (AH)

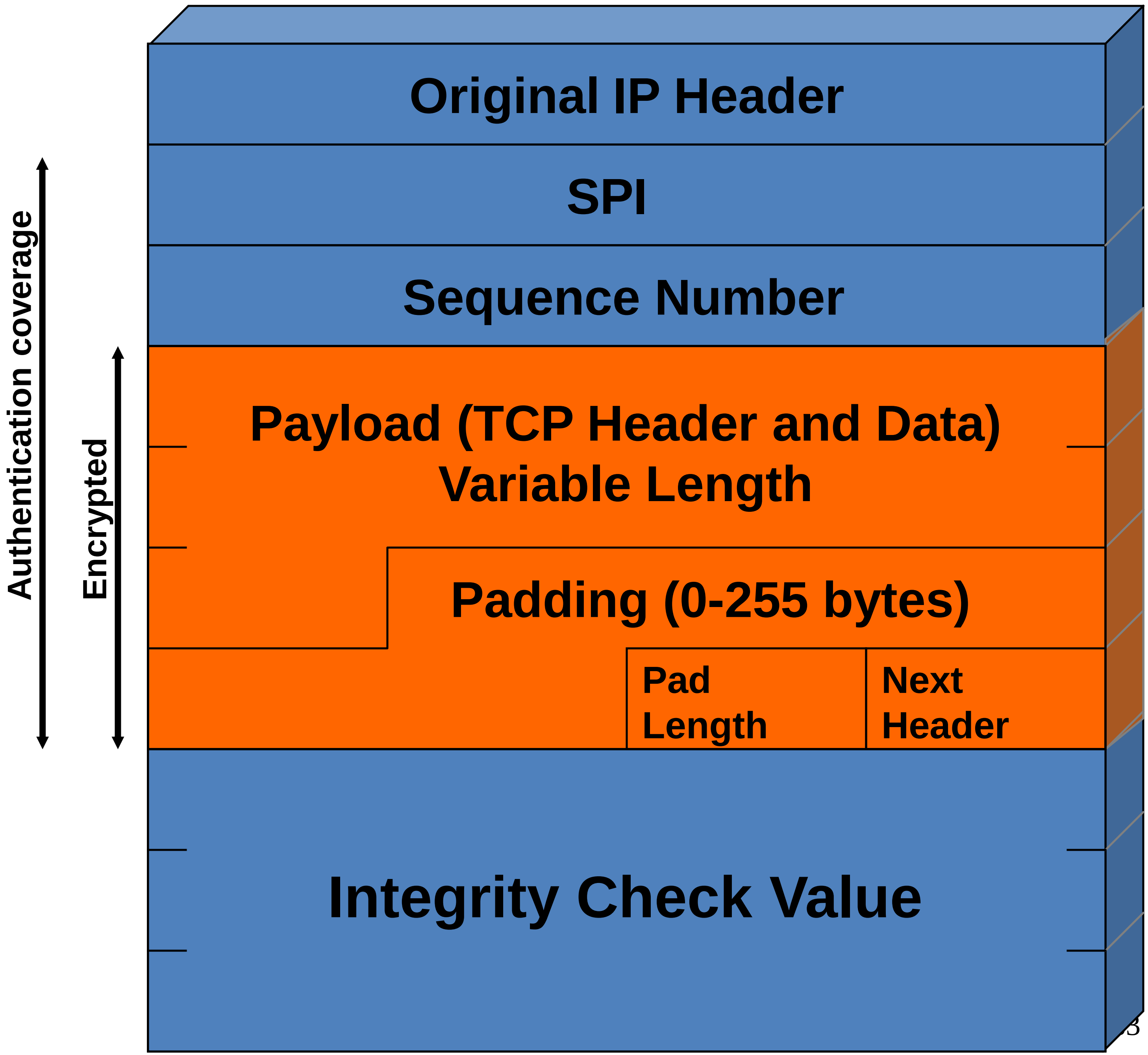
- Data integrity
  - Entire packet has not been tampered with
- Authentication
  - Can “trust” IP address source
  - Use MAC to authenticate
- Anti-replay feature
- Integrity check value

IPsec Authentication Header



Encapsulated Security Protocol (ESP)

- Confidentiality for upper layer protocol
- Partial traffic flow confidentiality (Tunnel mode only)
- Data origin authentication


Original IP Header
SPI
Sequence Number
Payload (TCP Header and Data)
Variable Length
Padding (0-255 bytes)
Pad Length
Next Header
Integrity Check Value
Authentication coverage
Encrypted

Key Management

- Why do we need Internet key management
  - AH and ESP require encryption and authentication keys
- Process to negotiate and establish IPsec SAs between two entities

Security Principles (Cont'd)

- Perfect forward secrecy (PFS)
  - Compromise of current keys (session key or long-term key) doesn't compromise past session keys.
  - Concern for encryption keys but not for authentication keys.

Examples of Non Perfect Forward Secrecy

- Alice sends all messages with Bob's public key, Bob sends all messages with Alice's public key
- Kerberos
- Alice chooses session keys, and sends them to Bob, all encrypted with Bob's public key

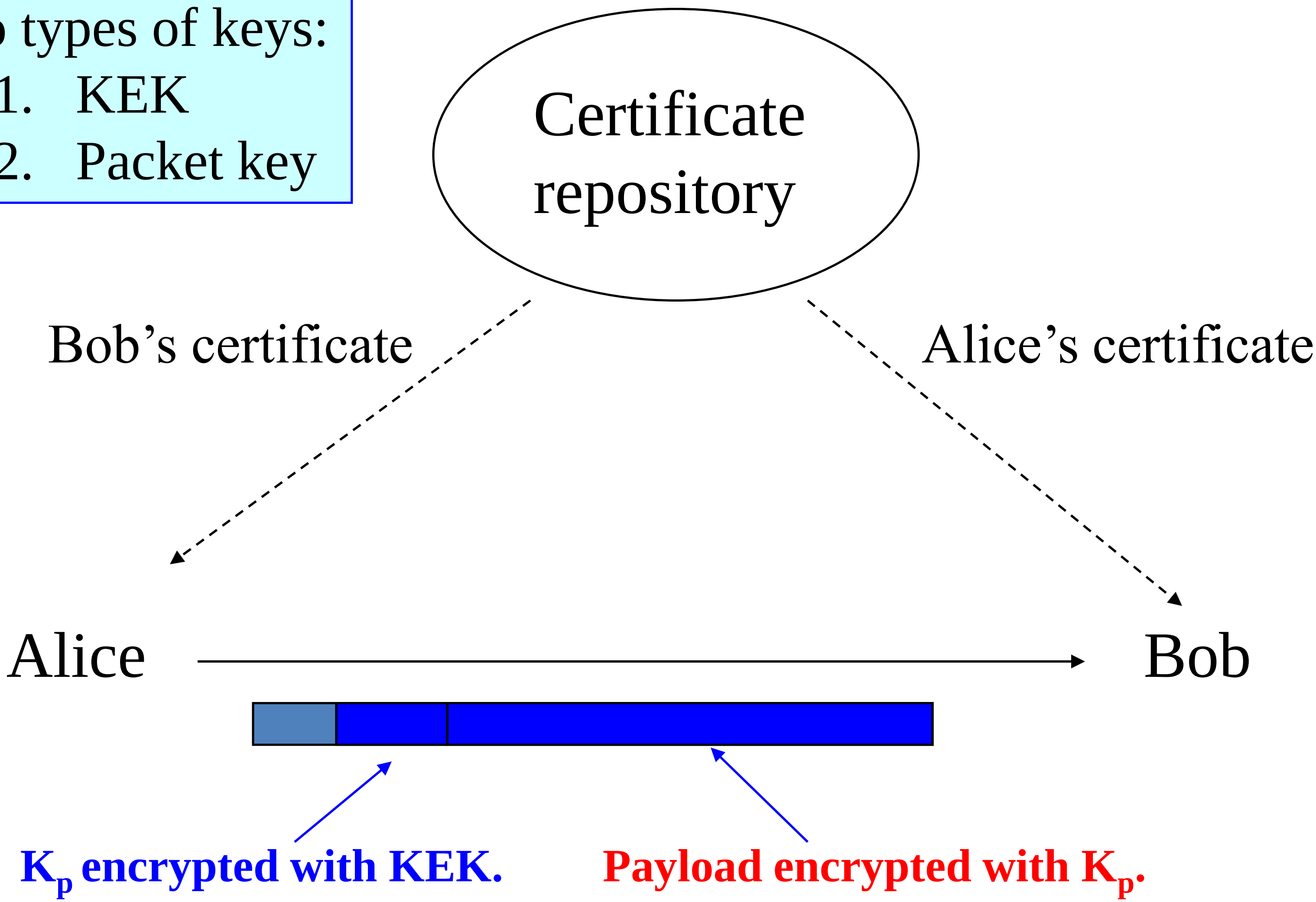
Automatic Key Management

- Key establishment and management combined
  - SKIP
- Key establishment protocol
  - Oakley
    - focus on key exchange
- Key management
  - Internet Security Association & Key Management Protocol (ISAKMP)
    - Focus on SA and key management
    - Clearly separated from key exchange.

SKIP (Cont'd)

Two types of keys:

1. KEK
2. Packet key



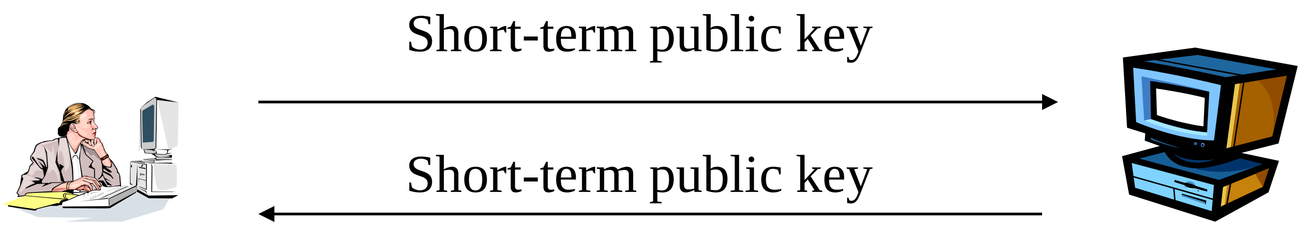
SKIP (Cont’d)

- Limitations
  - No Perfect Forward Secrecy
  - No concept of SA; difficult to work with the current IPsec architecture
- Not the standard, but remains as an alternative.

Oakley

- Oakley is a refinement of the basic Diffie-Hellman key exchange protocol.
- Why need refinement?
  - Resource clogging attack
  - Replay attack
  - Man-in-the-middle attack
  - Choice of D-H groups

Ephemeral Diffie-Hellman



- Session key is computed on the basis of short-term DH public keys.
- Exchange of these short-term public keys requires authentication and integrity.
  - Digital signatures.
  - Keyed message digests.
- Perfect forward secrecy?

Ephemeral Diffie-Hellman

- Question: What happens if the long term key is compromised?

ISAKMP

- Oakley
  - Key exchange protocol
  - Developed to use with ISAKMP
- ISAKMP
  - Internet security association and key management protocol
  - Defines procedures and packet formats to establish, negotiate, modify, and delete security associations.
  - Defines payloads for security association, key exchange, etc.

IKE Overview (Cont'd)

- Request-response protocol
  - Initiator
  - Responder
- Two phases
  - Phase 1: Establish an IKE (ISAKMP) SA
  - Phase 2: Use the IKE SA to establish IPsec SAs

IKE Overview (Cont'd)

- Several Modes
  - Phase 1:
    - Main mode: identity protection
    - Aggressive mode
  - Phase 2:
    - Quick mode
  - Other modes
    - New group mode
      - Establish a new group to use in future negotiations
      - Not in phase 1 or 2;
      - Must only be used after phase 1
    - Informational exchanges

IKE Phase 1

- Negotiating cryptographic parameters
  - Specifies suites of acceptable algorithms:
    - {(3DES, MD5, RSA public key encryption, DH),
    - (AES, SHA-1, pre-shared key, elliptic curve), …}
  - Specifies a MUST be implemented set of algorithms:
    - Encryption=DES, hash=MD5/SHA-1, authentication=pre-shared key/DH
  - The lifetime of the SA can also be negotiated

IKE Phase 1

- Four authentication methods
  - Authentication with public signature key
  - Authentication with public key encryption
  - Authentication with public key encryption, revised
  - Authentication with a pre-shared key

IKE Phase 2 -- Quick Mode

- Negotiates parameters for the phase-2 SA
- Information exchanged with quick mode must be protected by the phase-1 SA
- Essentially a SA negotiation and an exchange of nonces
- Used to derive keying materials for IPsec SAs

IKE Phase 2 -- Quick Mode (Cont'd)

- 3-messages protocol

X, Y, CP, traffic, SPI_A , $nonce_A$, $g^a \bmod p$

→

X, Y, CPA, traffic, SPI_B , $nonce_B$, $g^b \bmod p$

←

X, Y, ack

→