CS 114: Network Security

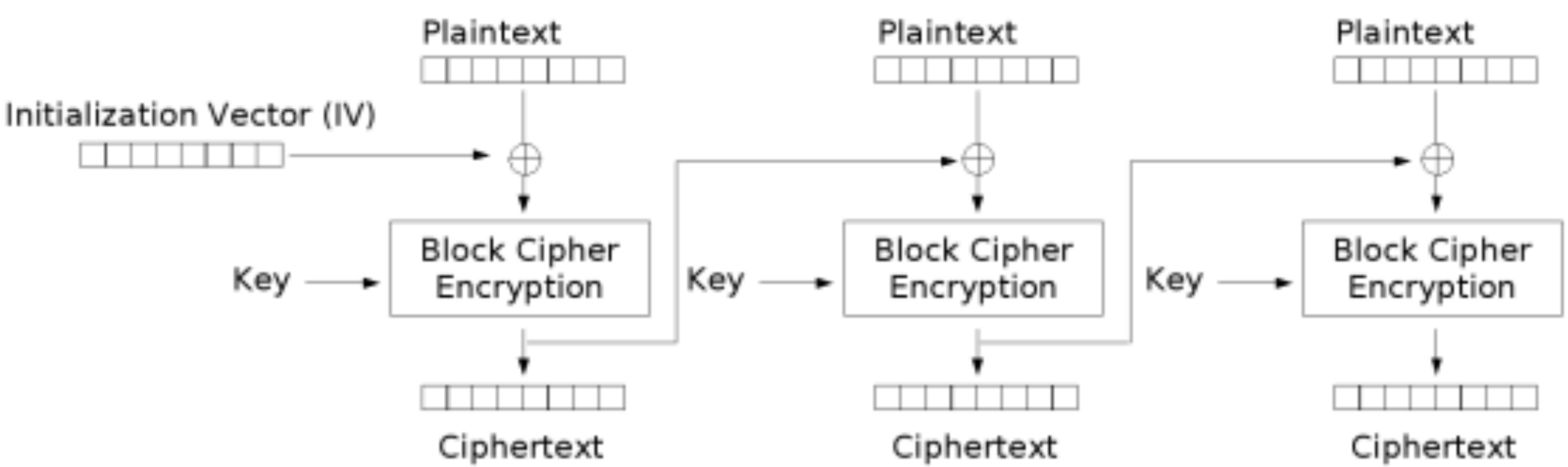
Lecture 8 - Authentication Part II

Prof. Daniel Votipka
Spring 2023

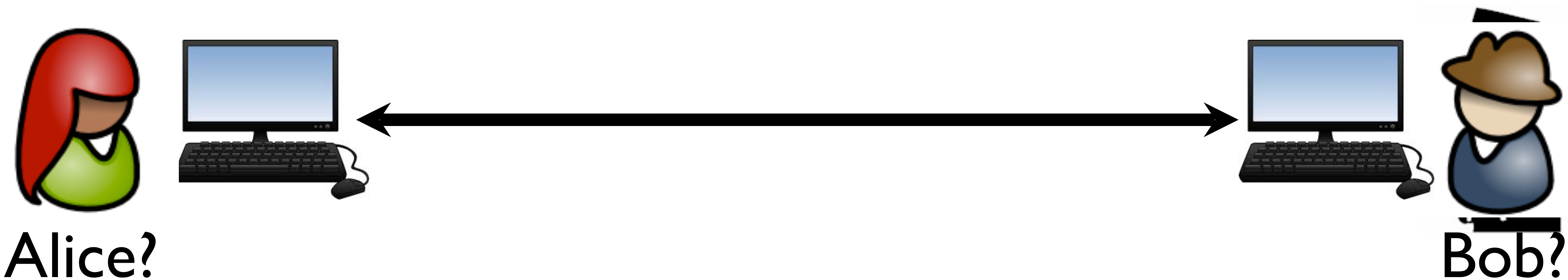
(some slides courtesy of Prof. Micah Sherr, Patrick McDaniel, and Vitaly Shmatikov)

Administrivia

- Exam 1 on Thursday **<u>in class</u>**
  - Review at the end of this lecture
- Homework 1, part 2 note:
  - Encryption and decryption with CBC must be in the same order



Authentication



What is Authentication?

- Establishes identity
  - Answers the question: To whom am I speaking?
  - Credential – proof of identity
  - Evaluation – process that assesses the correctness of the association between credential and claimed identity

Three Flavors of Credentials

- … are evidence used to prove identity
- Credentials can be
  1. **Something I am**
  2. **Something I know**
  3. **Something I have**

"Salt"ing passwords

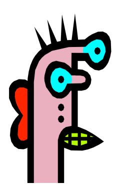
- Suppose you want to make an *offline dictionary attack* more difficult
- A *salt* is a random number added to the password
- This is the approach taken by any reasonable system

$$
\begin{array}{c}
salt_1, h(salt_1, pw_1) \\
salt_i, h(salt_2, pw_2) \\
salt_i, h(salt_3, pw_3) \\
\ldots \\
salt_n, h(salt_n, pw_n)
\end{array}
$$

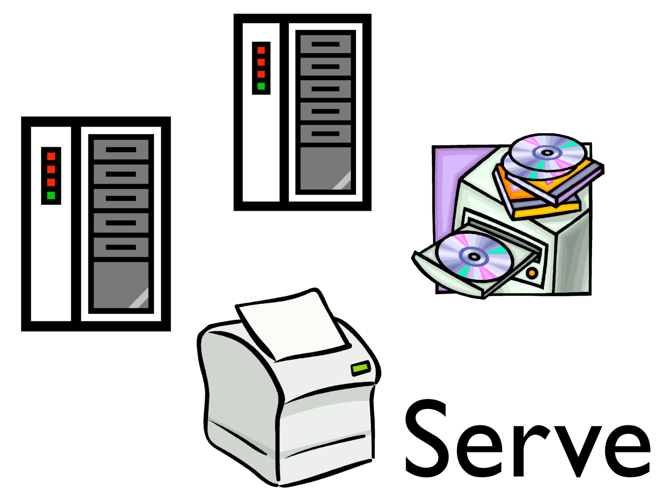
Three Flavors of Credentials

- … are evidence used to prove identity
- Credentials can be
  1. **Something I am**
  2. **Something I know**
  3. **Something I have**

Authentication



User



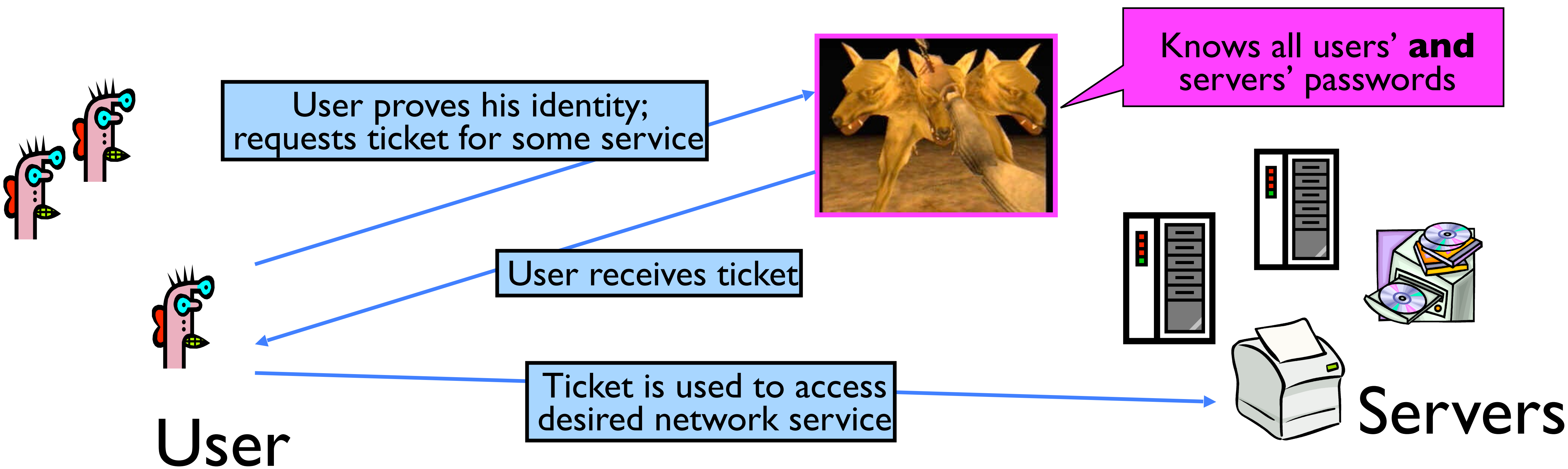
Servers

Kerberos

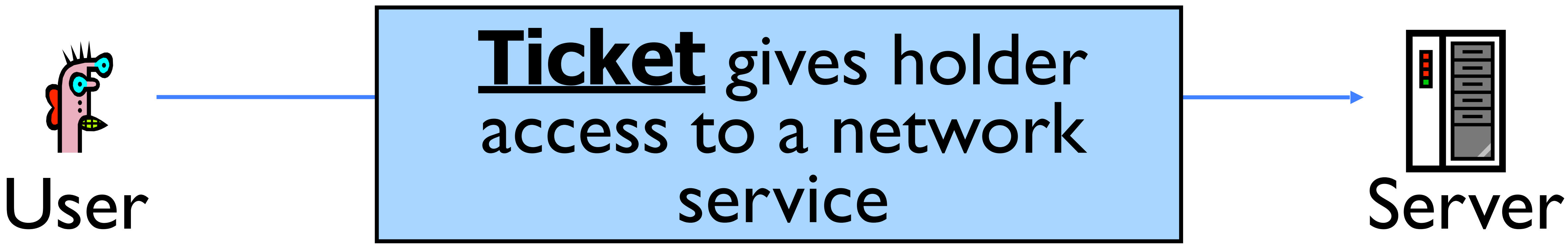
Kerberos

- An online system that resists password eavesdropping and achieves **mutual authentication**
- First single sign-on system (SSO)
- Easy application integration API
- Most widely used (non-web) centralized password system in existence
- Now part of Windows network authentication

Kerberos Overview

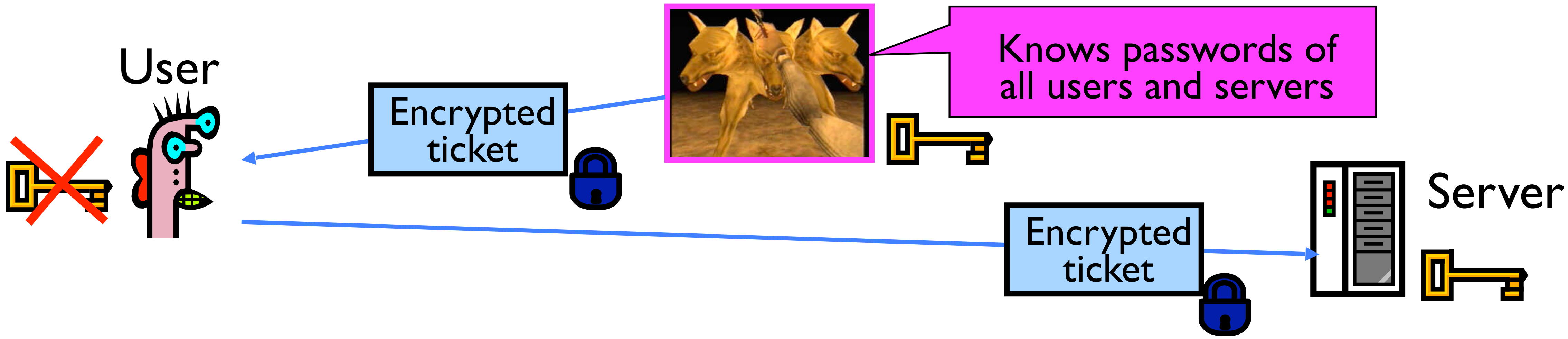


What Should a Ticket Look Like?



- Ticket cannot include server's plaintext password
  - Otherwise, next time user will access server directly without proving his identity to authentication service
- Solution: encrypt some information with a key known to the server (but not the user!)
  - Server can decrypt ticket and verify information
  - User does not learn server's key

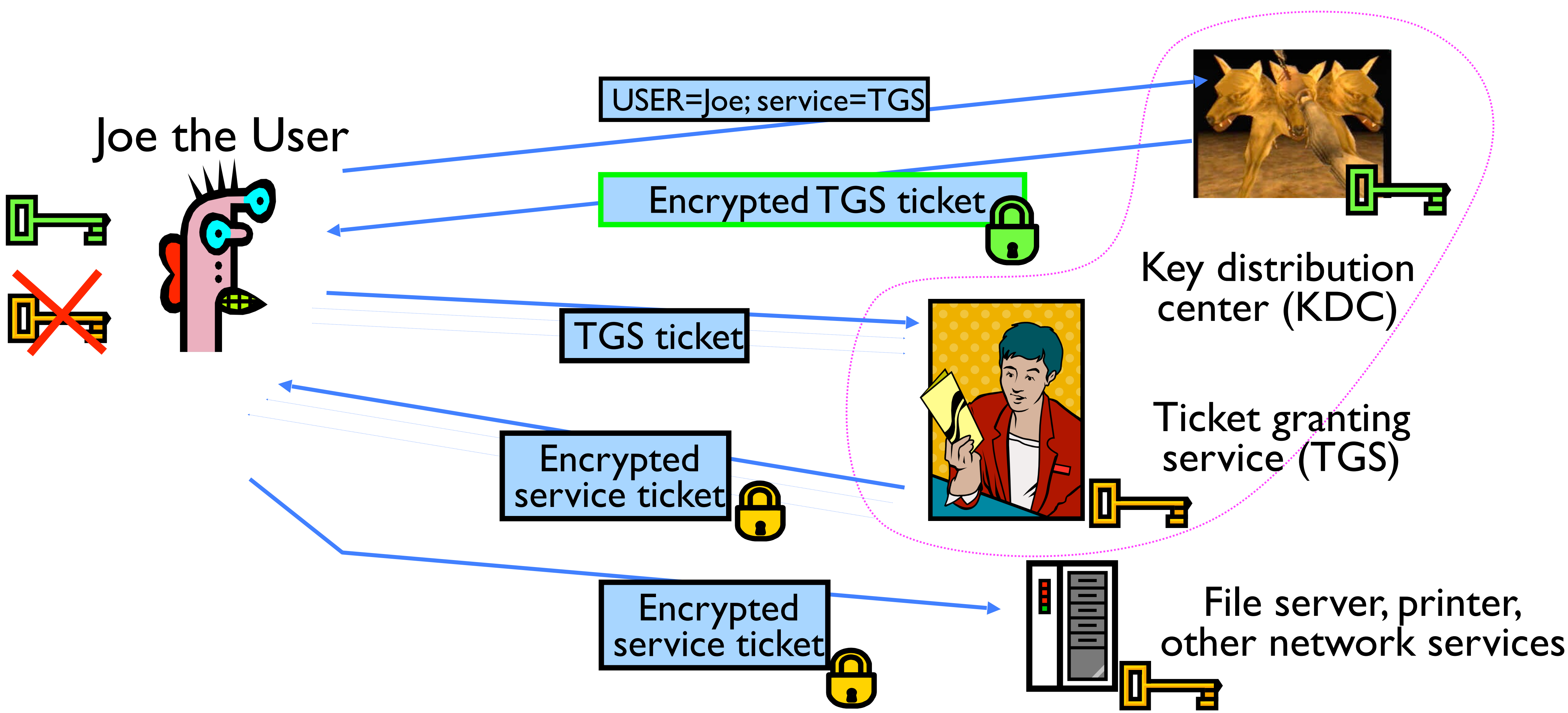
What should a ticket include?



- User name
- Server name
- Address of user's workstation -- **WHY?**
- Ticket lifetime -- **WHY?**
- A few other things (e.g., session key)

Two-Step Authentication

- Prove identity once to obtain special TGS ticket
- Use TGS to get tickets for any network service



Not quite good enuf...

- **Ticket hijacking**
  - Malicious user may steal the service ticket of another user on the same workstation and use it
    - IP address verification does not help
  - Servers must verify that the user who is presenting the ticket is the same user to whom the ticket was issued
- **No server authentication**
  - Attacker may misconfigure the network so that he receives messages addressed to a legitimate server
    - Capture private information from users and/or deny service
  - Servers must prove their identity to users
  - We want mutual authentication

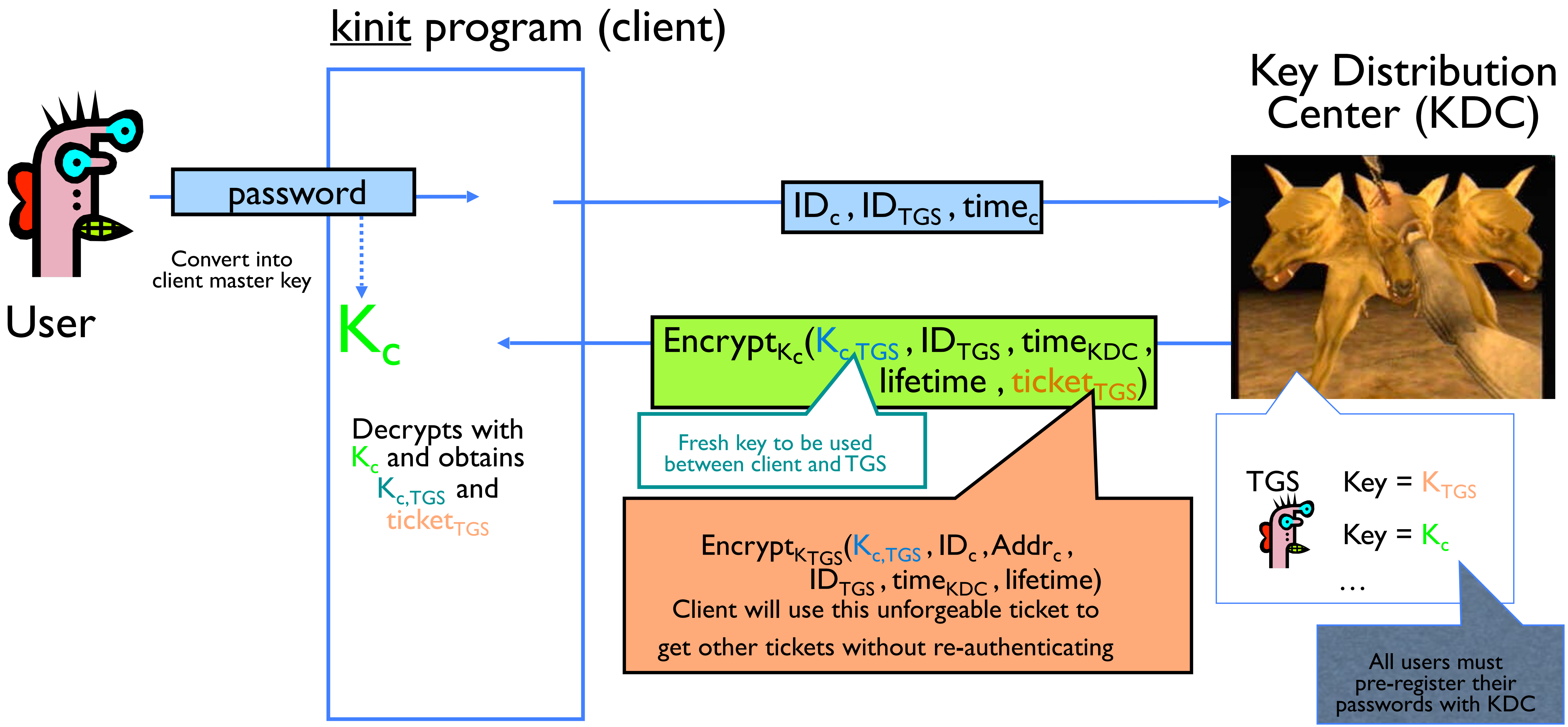
Symmetric Keys in Kerberos

- K_c is long-term key of client C
  - Derived from user's password
  - Known to client and key distribution center (KDC)
- K_{TGS} is long-term key of TGS
  - Known to KDC and ticket granting service (TGS)
- K_v is long-term key of network service V
  - Known to V and TGS; separate key for each service
- $K_{c,TGS}$ is short-term *session* key between C and TGS
  - Created by KDC, known to C and TGS
- $K_{c,v}$ is short-term session key between C and V
  - Created by TGS, known to C and V

Brace yourself!
It's Kerberos time!

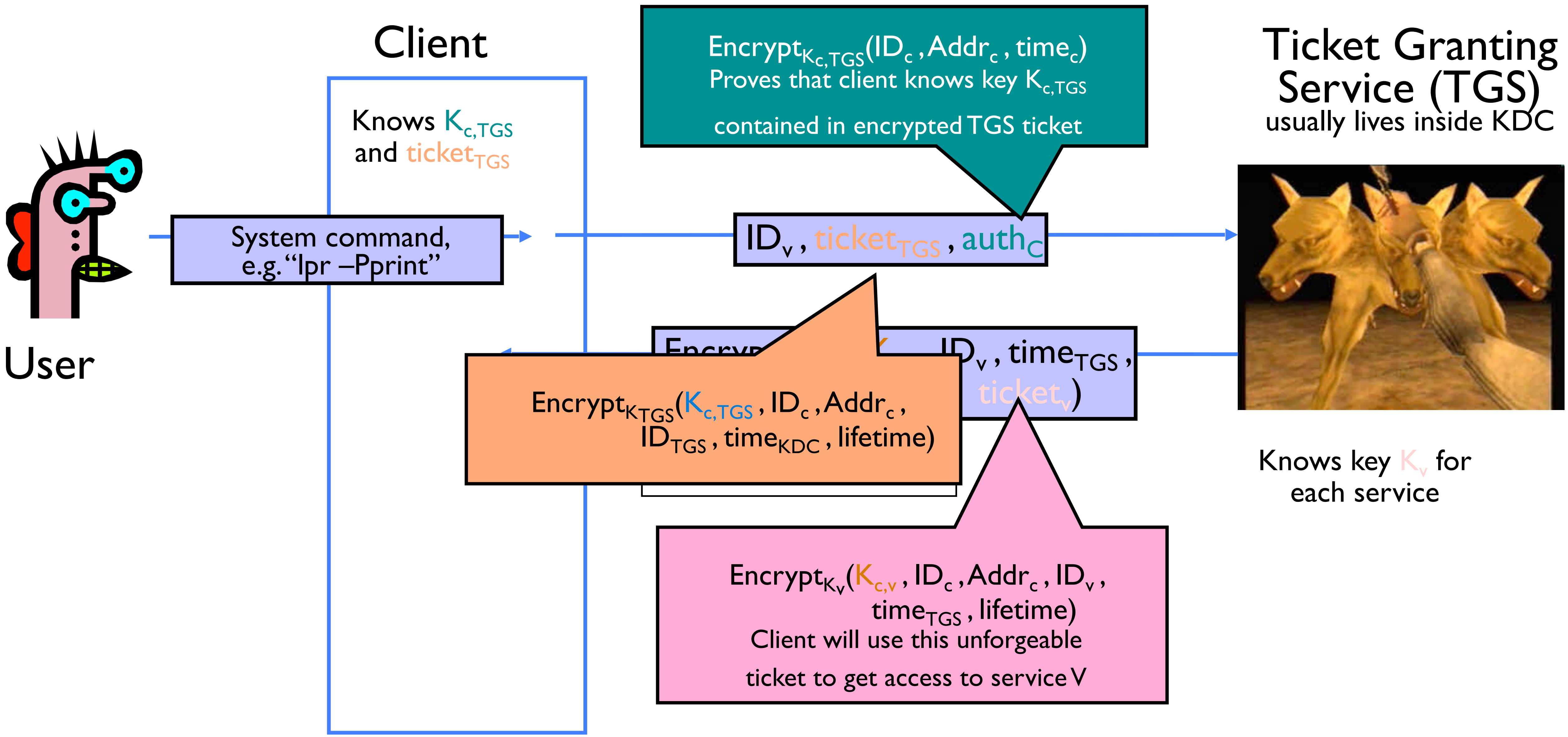
- Three-step process:
  - "Logon" -- obtain TGS ticket from KDC
  - Obtain "service ticket" from TGS
  - Use service

“Single Logon” Authentication



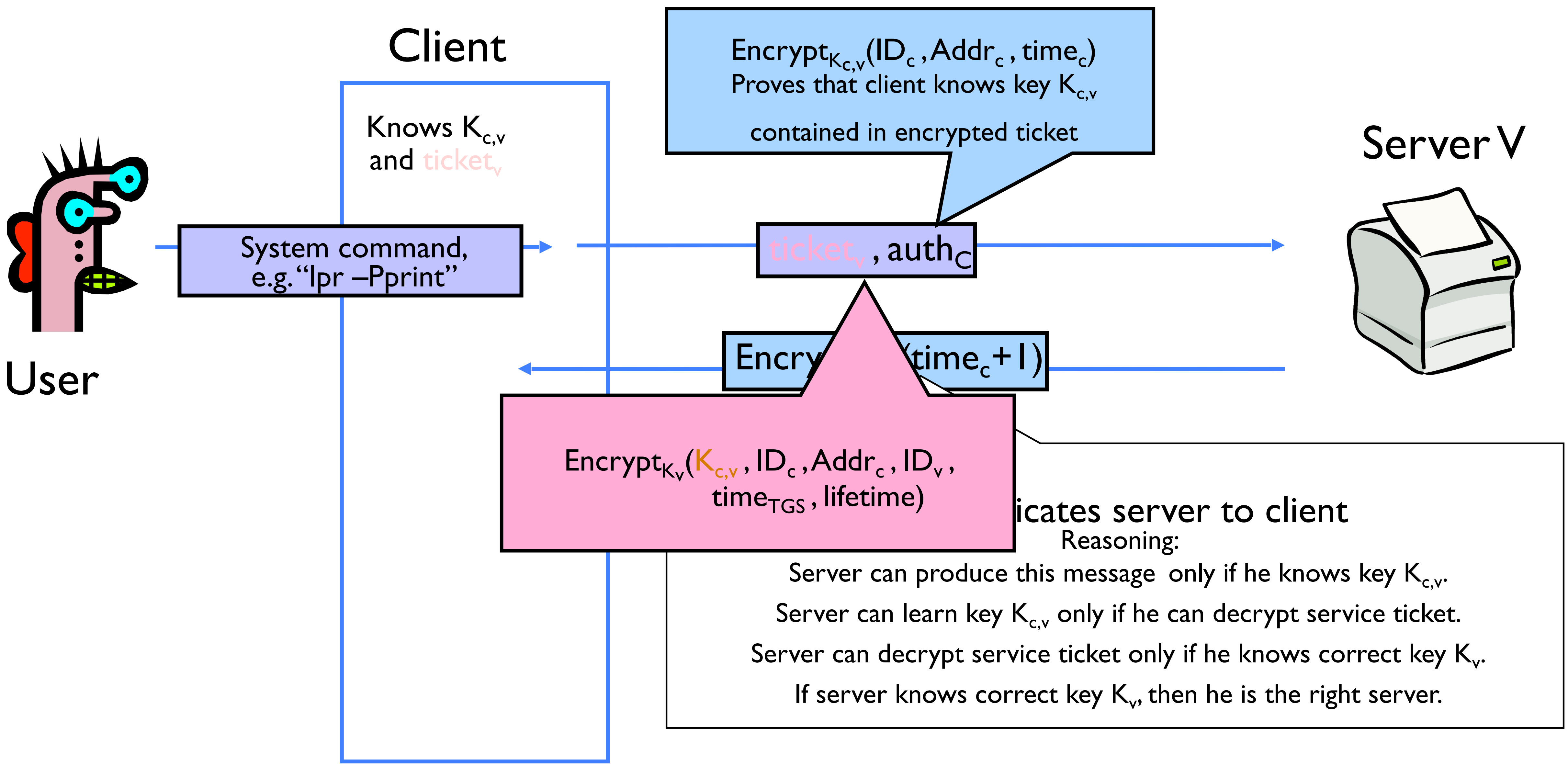
- Client only needs to obtain TGS ticket once (say, every morning)
  - Ticket is encrypted; client cannot forge it or tamper with it

Obtaining a Service Ticket



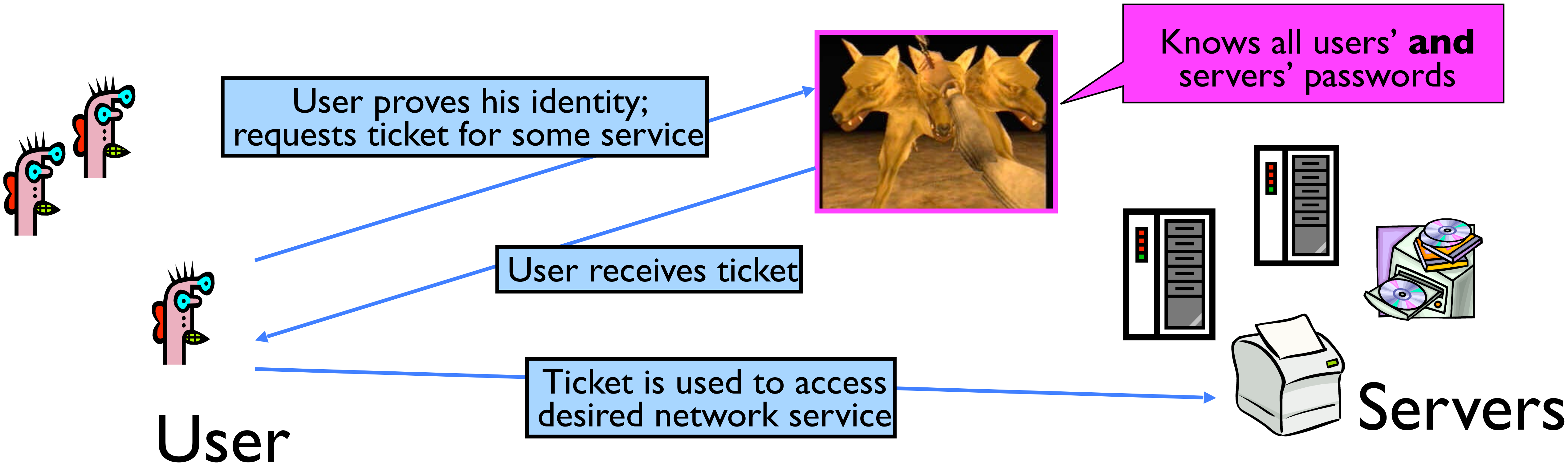
- Client uses TGS ticket to obtain a service ticket and a short-term key for each network service
  - One encrypted, unforgeable ticket per service (printer, email, etc.)

Use Service

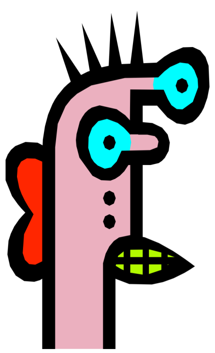


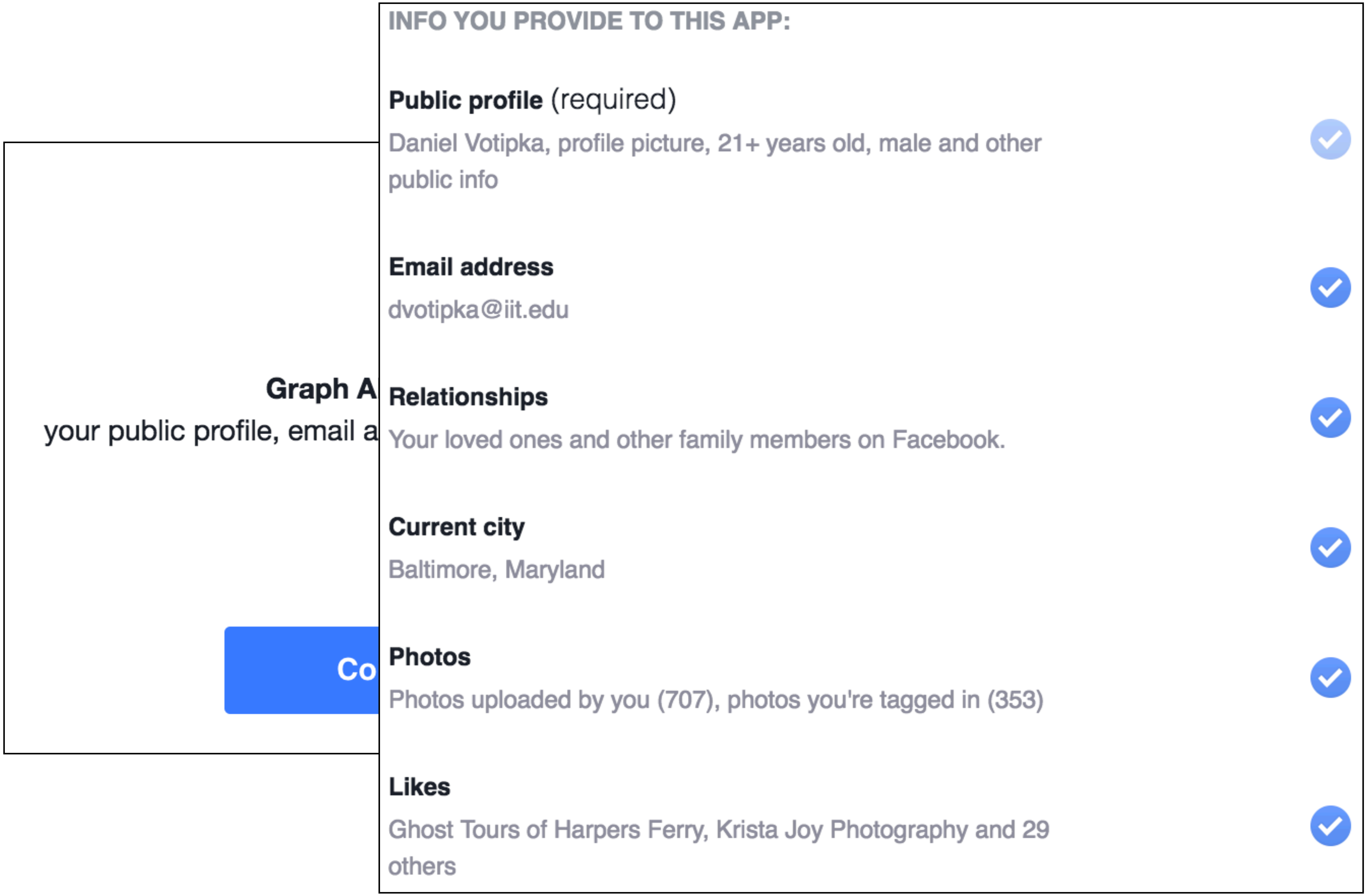
- For each service request, client uses the short-term key for that service and the ticket he received from TGS

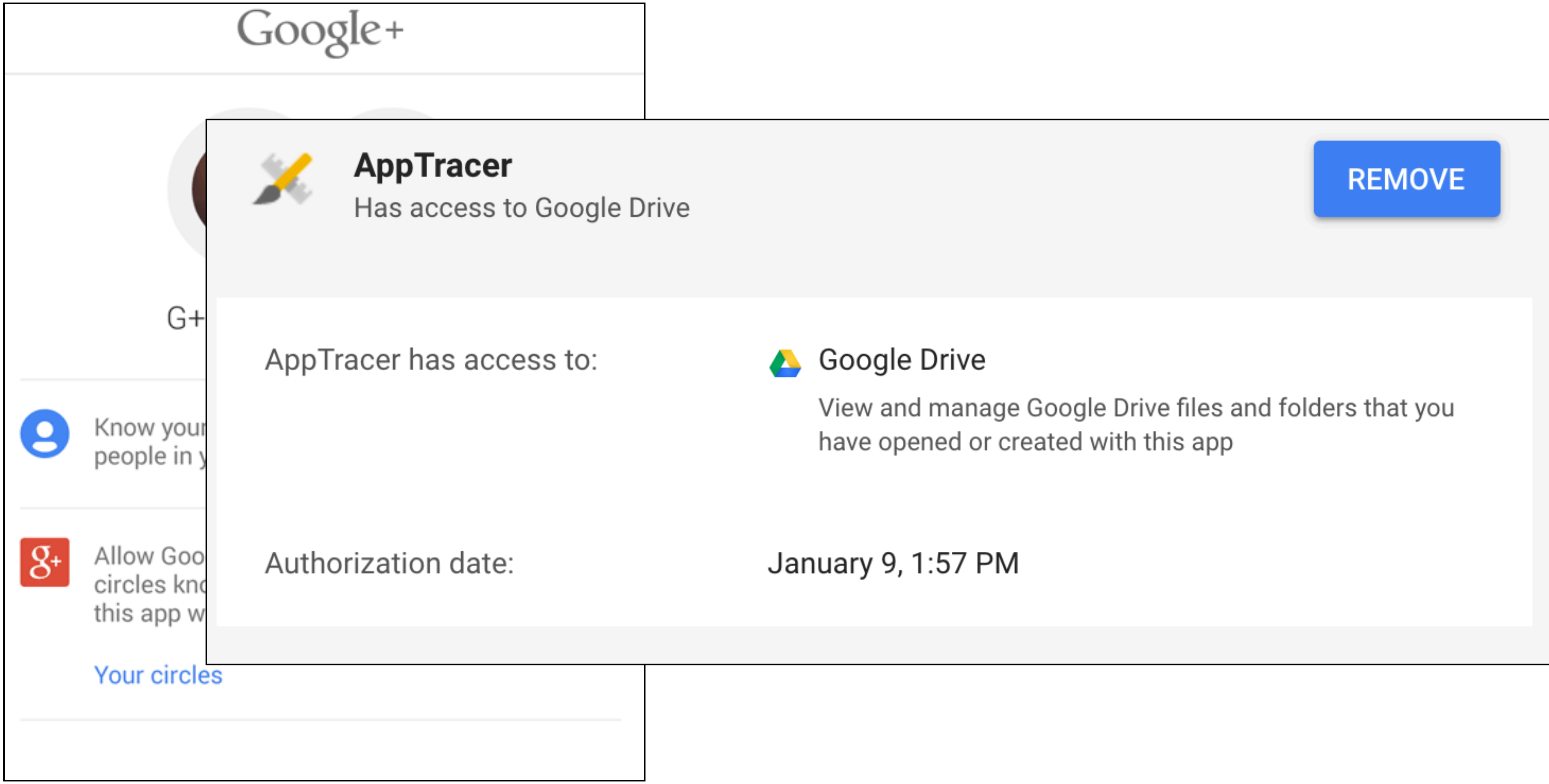
Kerberos Overview



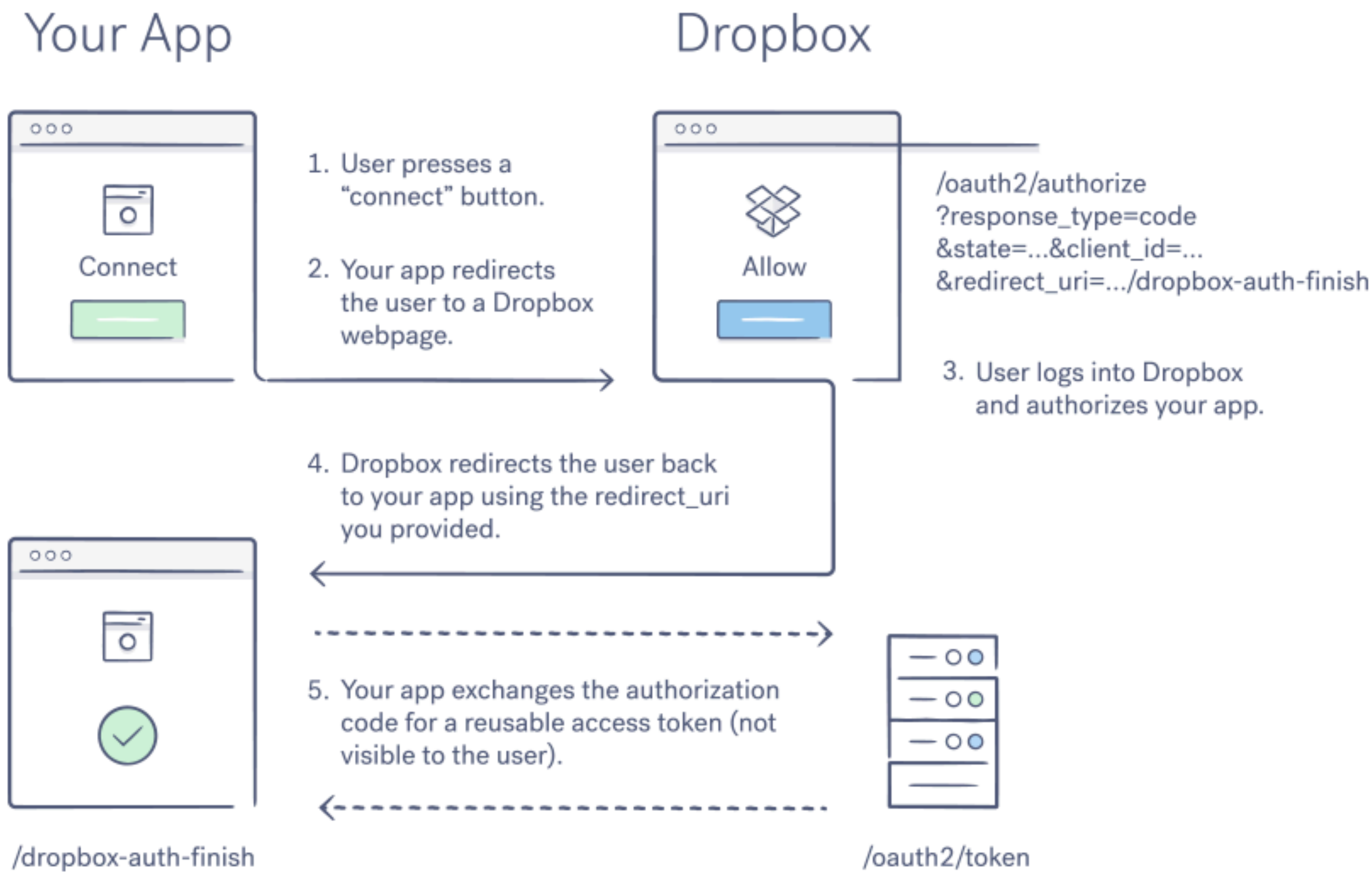
Open Authorization (OAuth)




INFO YOU PROVIDE TO THIS APP:
Public profile (required)
Daniel Votipka, profile picture, 21+ years old, male and other public info
Email address
dvotipka@iit.edu
Relationships
Your loved ones and other family members on Facebook.
Current city
Baltimore, Maryland
Photos
Photos uploaded by you (707), photos you're tagged in (353)
Likes
Ghost Tours of Harpers Ferry, Krista Joy Photography and 29 others
Graph A
your public profile, email a
Co


Google+
G+
Know your
people in y
Allow Goo
circles kno
this app w
Your circles
AppTracer
Has access to Google Drive
REMOVE
AppTracer has access to:
Google Drive
View and manage Google Drive files and folders that you have opened or created with this app
Authorization date:
January 9, 1:57 PM

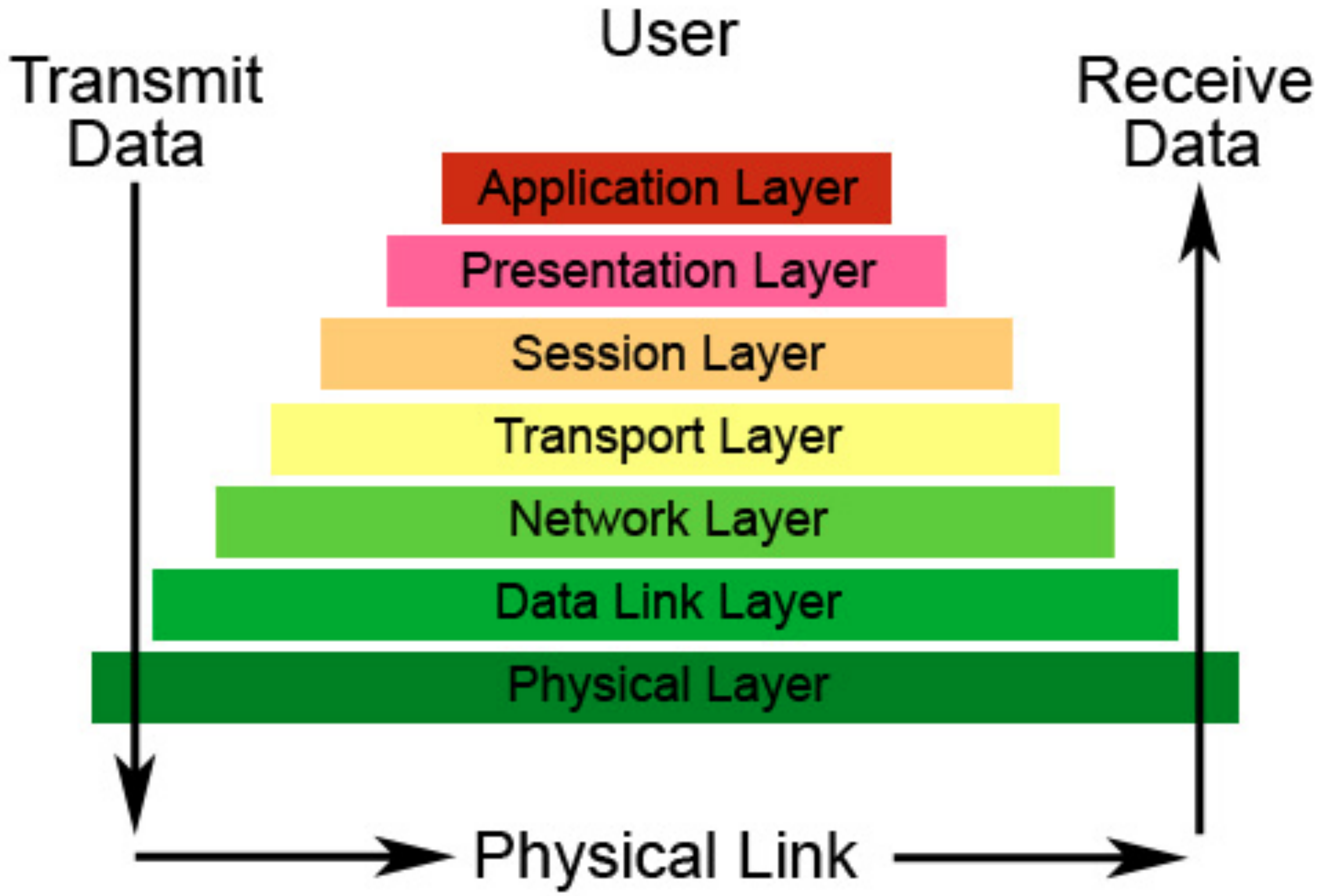
Open Authorization (OAuth)

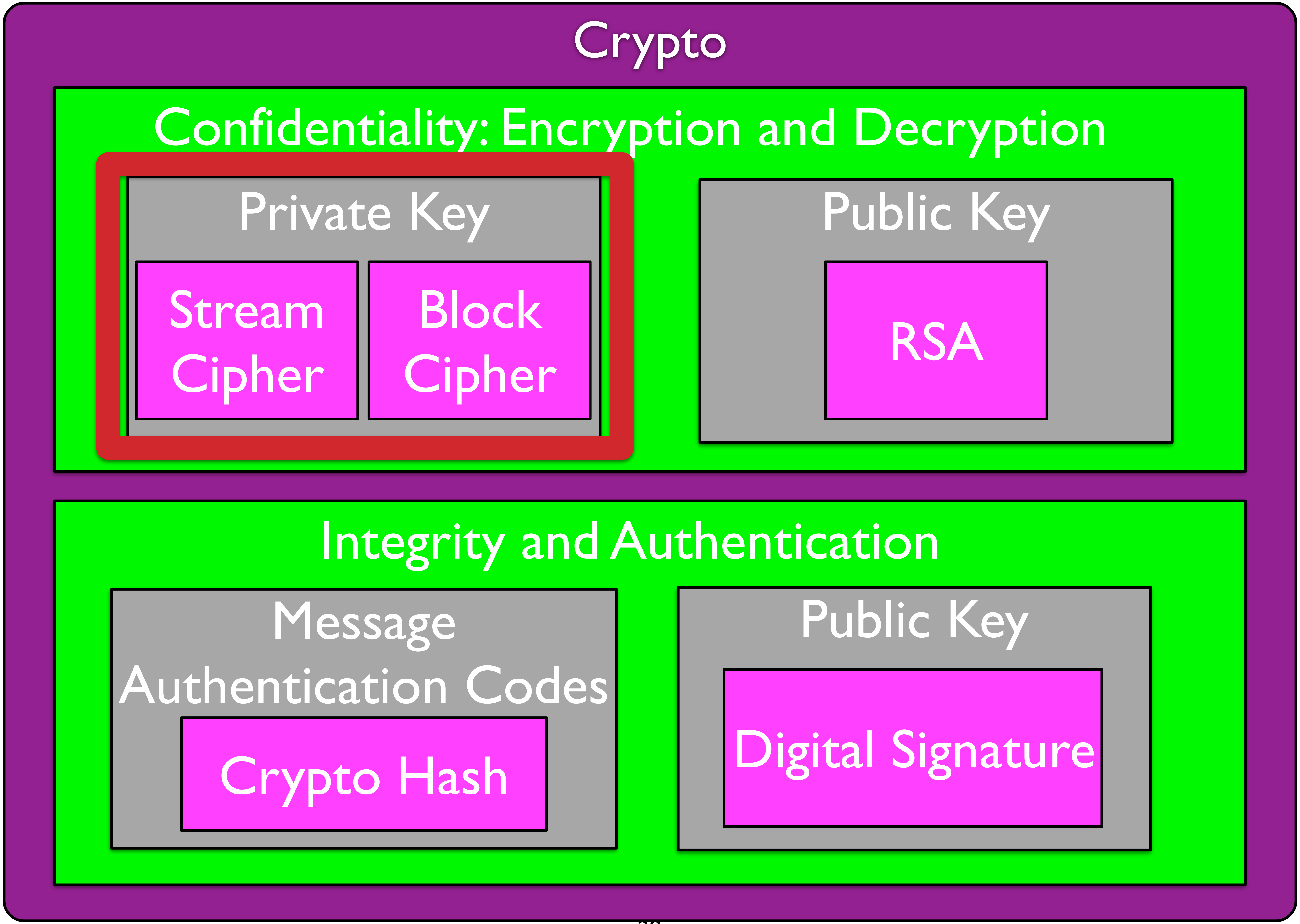


Review for Exam 1

- Closed-book, closed-notes, non-collaborative
- You'll have 75 minutes to complete the exam (1:30 - 2:45pm)
- Covers everything from Lecture 2 - 6

The Seven Layers of OSI



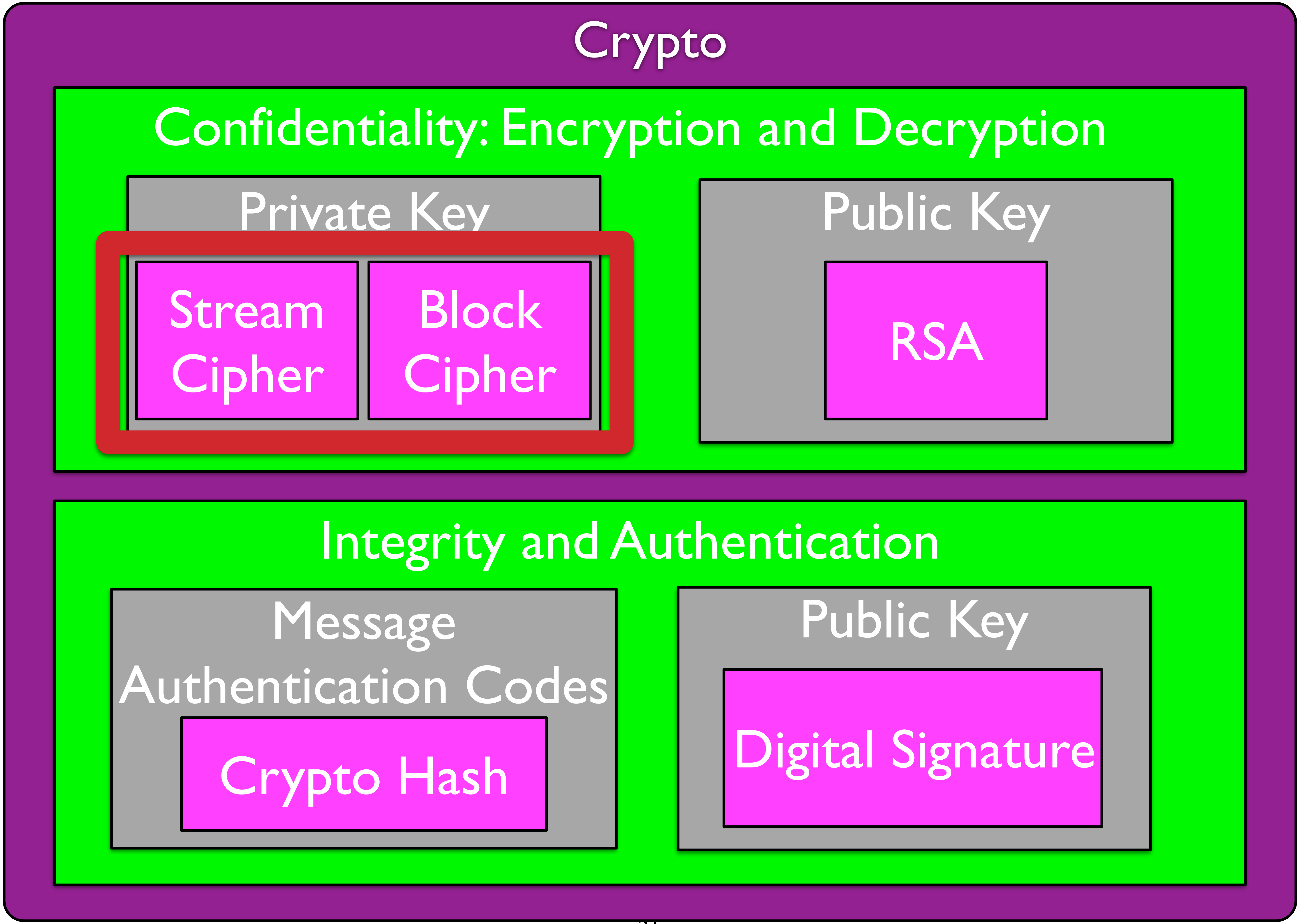

Crypto
Confidentiality: Encryption and Decryption
Private Key
Stream Cipher
Block Cipher
Public Key
RSA
Integrity and Authentication
Message Authentication Codes
Crypto Hash
Public Key
Digital Signature

Classic Private Key Crypto

- **Caesar Cipher**
- **Substitution Cipher**
- **One-Time Pad**

Kerckhoffs' Principles

- **Kerckhoffs' principles** [1883]:
  - Assume Eve knows cipher algorithm
  - Security should rely on choice of key
  - If Eve discovers the key, a new key can be chosen

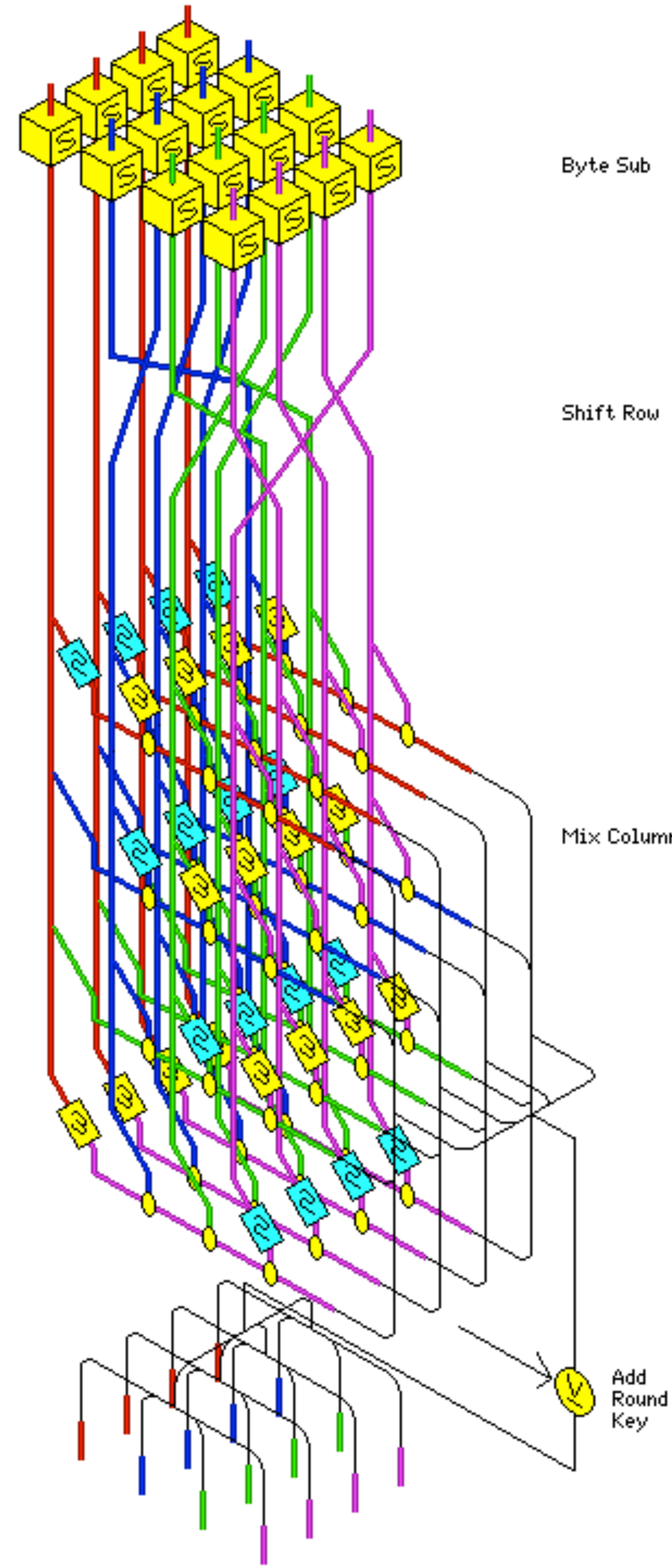

Crypto
Confidentiality: Encryption and Decryption
Private Key
Stream Cipher
Block Cipher
Public Key
RSA
Integrity and Authentication
Message Authentication Codes
Crypto Hash
Public Key
Digital Signature

Stream Ciphers

- **Key reuse:** [C(K) = pseudorandom stream produced using key K]
  - E(M1) = M1 ⊕ C(K)
  - E(M2) = M2 ⊕ C(K)
  - Suppose Eve knows ciphertexts E(M1) and E(M2)
  - E(M1) ⊕ E(M2) = M1 ⊕ C(K) ⊕ M2 ⊕ C(K) = M1 ⊕ M2
  - M1 and M2 can be derived from M1 ⊕ M2 using frequency analysis
- Countermeasure is to use IV (**initialization vector**)
  - IV sent in clear and is combined with K to produce pseudorandom sequence
  - E.g., replace C(K) with C(K⊕IV) or C(f(K,IV))
  - IVs should never be reused and should be sufficiently large
  - WEP broken partly because IVs were insufficiently large
  - modern stream ciphers take IVs, but it's up to the programmer to generate them

Block Ciphers

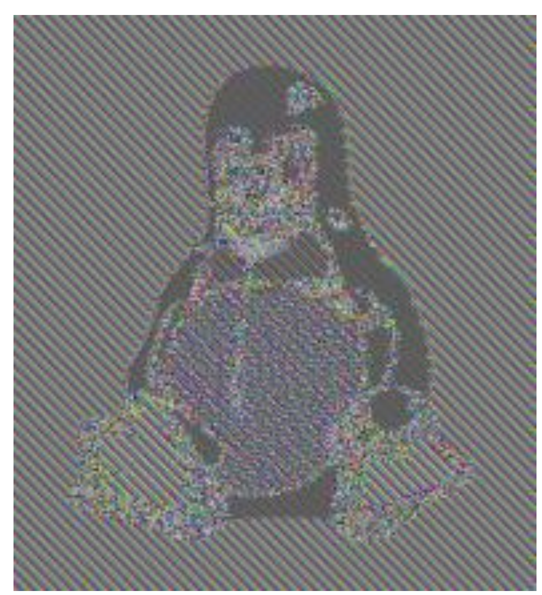
- Plaintext broken into fixed-sized blocks
- Each block individually encrypted
- Substitution-Permutation Networks
  - **S-Box**
    - Input: sequence of x bits
    - Output: new sequence of x bits
    - Mapping from one bit string to another
  - **Permutation**
    - Input: sequence of x bits
    - Output: permutation of the input
- Symmetric key encryption typically uses many rounds of S-Boxes and permutations, incorporating the key



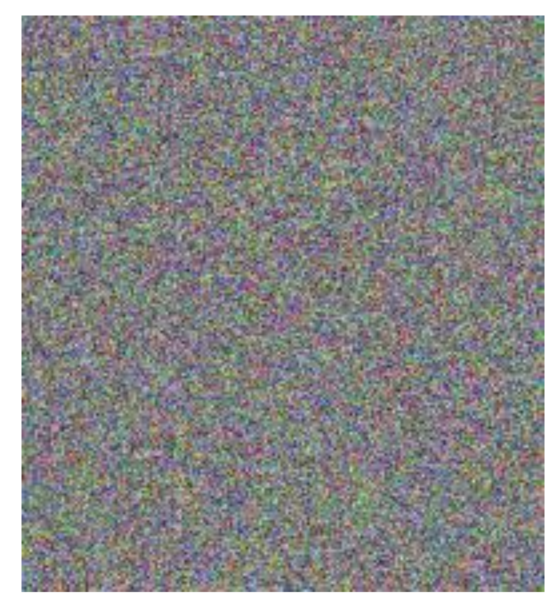
Modes of Operation: Electronic Codebook (ECB)

- Blocks are individually encrypted and concatenated together
- Problems:
  - Identical plaintext blocks produce identical ciphertext blocks
  - Encrypted blocks can be shuffled without detection

Plaintext



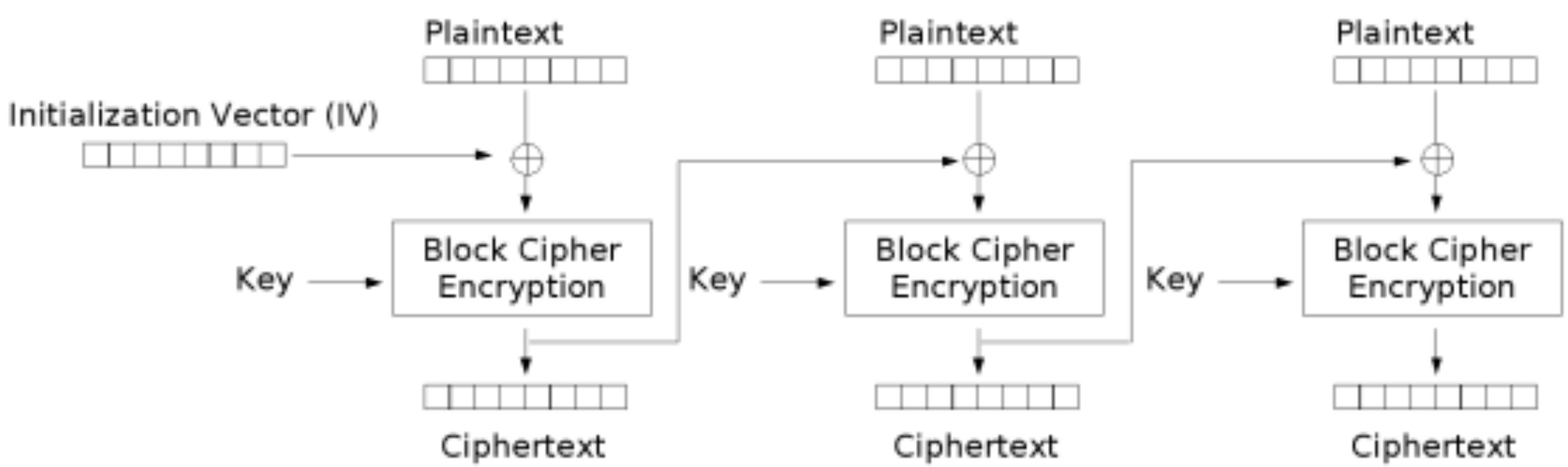
ECB



Other modes

Modes of Operation: Cipher-block Chaining (CBC)

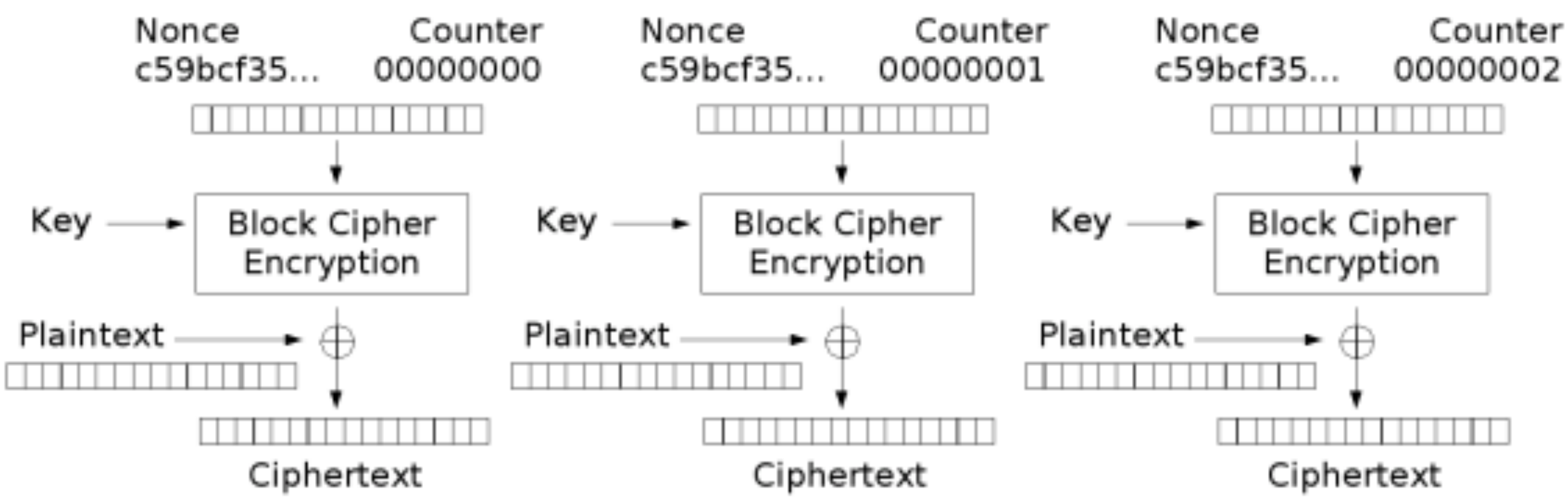
- Each block xor'd with ciphertext of previous block before encrypting
- Uses **initialization vector** (IV) to kickoff randomness
- IVs sent in the clear; should be randomly chosen for each session



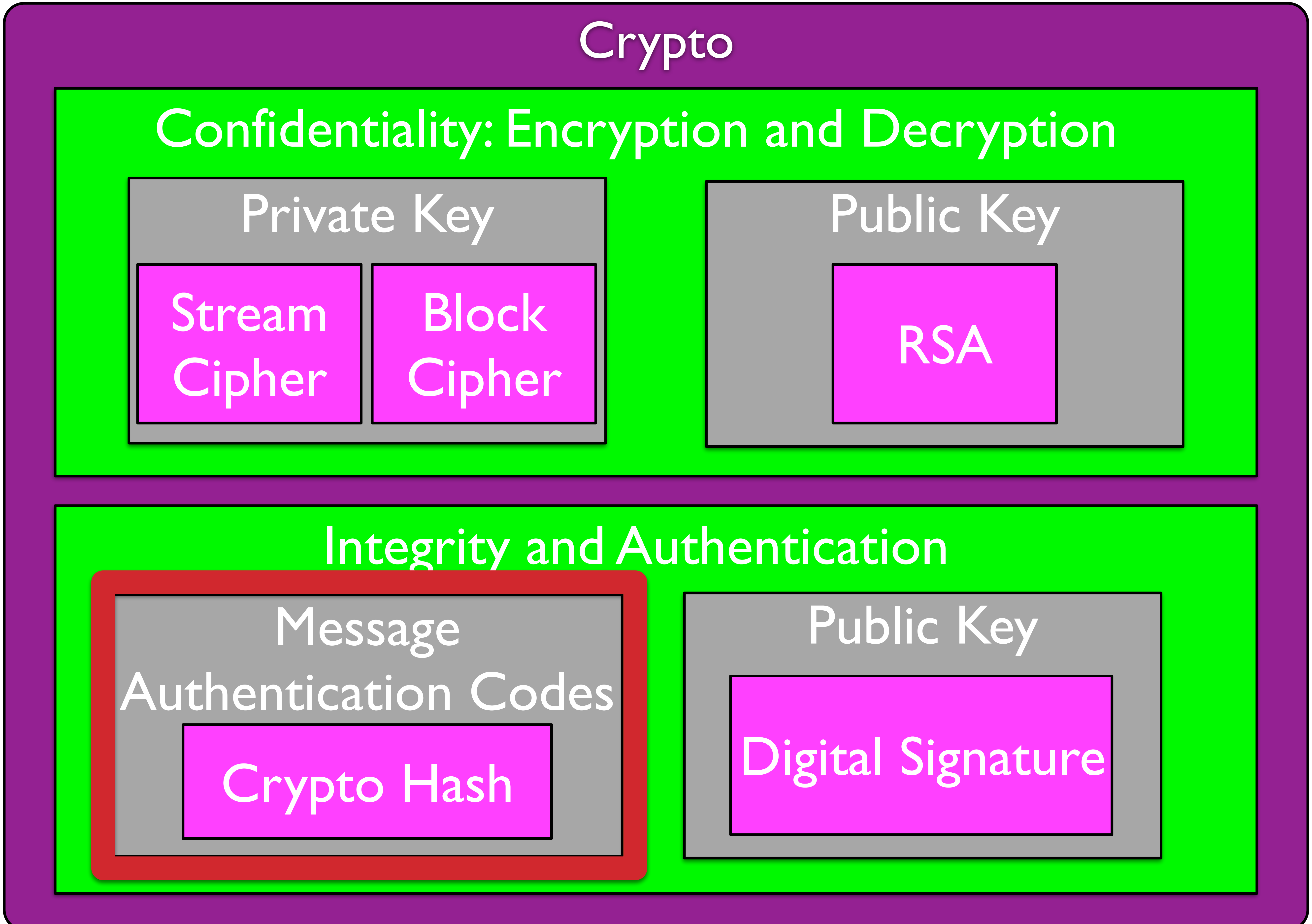
Cipher Block Chaining (CBC) mode encryption

Modes of Operation: Counter Mode (CTR)

- Allows random-access encryption/decryption
- Encrypts the IV plus a counter (incremented with each block), and xor the result with the plaintext
- Causes block cipher to function as a stream cipher



Counter (CTR) mode encryption


Crypto
Confidentiality: Encryption and Decryption
Private Key
Stream Cipher
Block Cipher
Public Key
RSA
Integrity and Authentication
Message Authentication Codes
Crypto Hash
Public Key
Digital Signature

Message Authentication Codes (MACs)

- MACs provide message **integrity** and **authenticity**
- $MAC_K(M)$ – use symmetric encryption to produce short sequence of bits that depends on both the message (M) and the key (K)
- MACs should be resistant to **existential forgery**: Eve should not be able to produce a valid MAC for a message M' without knowing K
- To provide confidentiality, authenticity, and integrity of a message, Alice sends
  - MAC-then-Encrypt: $E_K(M,MAC_K(M))$ where $E_K(X)$ is the encryption of X using key K; or
  - Encrypt-then-MAC: $E_K(M),MAC_K(E_K(M))$ ← Best option

  or

  - Encrypt-and-MAC: $E_K(M),MAC_K(M)$
  - Proves that M was encrypted (confidentiality) by someone who knew K (authenticity) and hasn't been changed (integrity)

Cryptographic Hash Functions

- **Hash function** `h`: deterministic one-way function that takes as input an arbitrary message `M` (sometimes called a *preimage*) and returns as output `h(M)`, a small fixed length *hash* (sometimes called a *digest*)
- Hash functions should have the following two properties:
  - *compression*: reduces arbitrary length string to fixed length hash
  - *ease of computation*: given message `M`, `h(M)` is easy to compute

Cryptographic Hash Functions

- Properties of good <u>cryptographic</u> hash functions:
  - **preimage resistance:** given digest y, computationally infeasible to find preimage x' such that h(x')=y
  - **2nd-preimage resistance:** given preimage x, computationally infeasible to find preimage x' such that h(x)=h(x')
  - **collision resistance:** computationally infeasible to find preimages i,j such that h(i)=h(j)

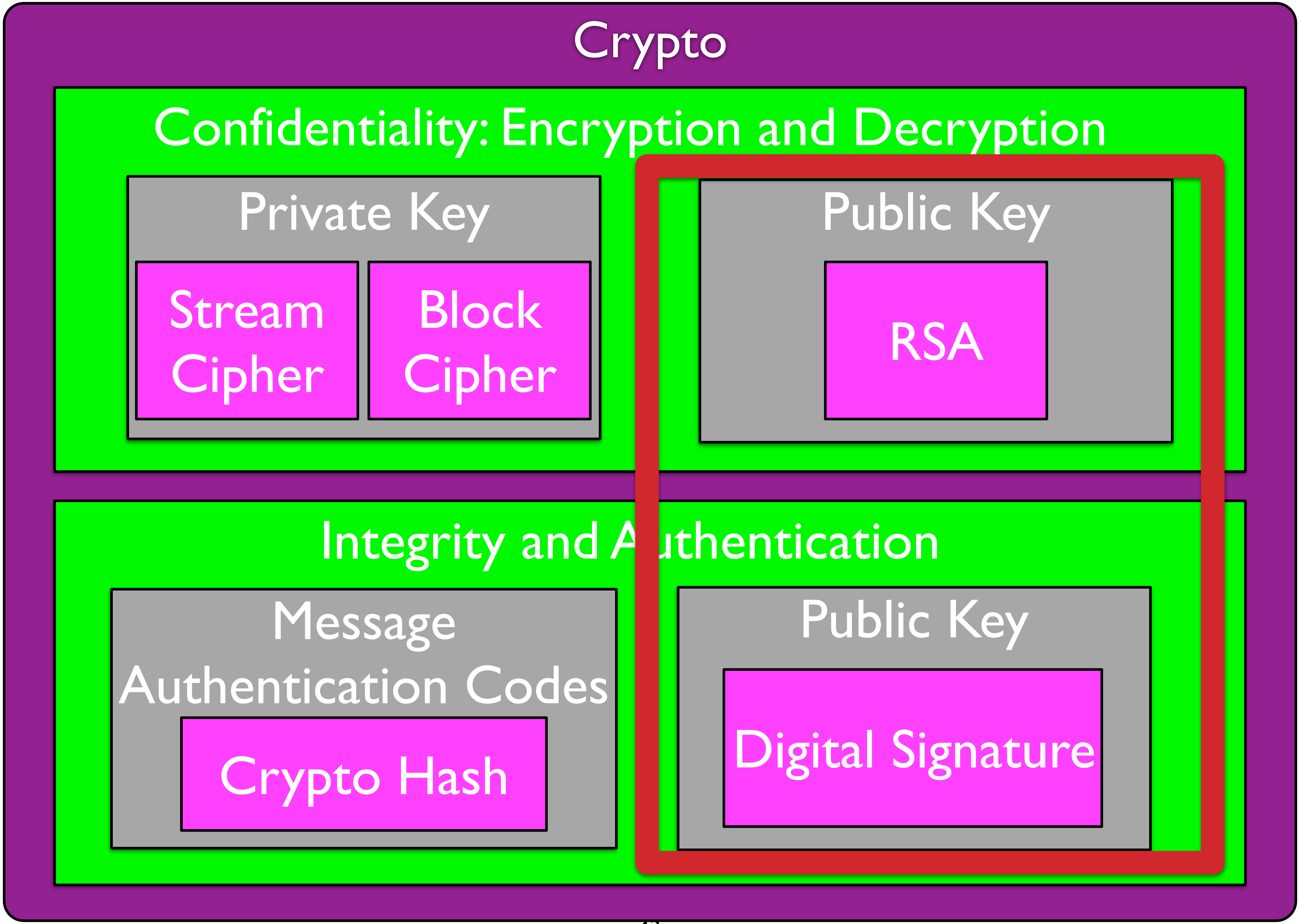
How do we use to make a MAC?

- $MAC_K(M) = h(M|K)$
  - Only computable if you know K
  - Any change in data will cause change in hash

Encryption and Message Authenticity

Without knowing *k1*, Eve can't read Alice's message.

Without knowing *k2*, Eve can't compute a valid MAC for her forged message.


Crypto
Confidentiality: Encryption and Decryption
Private Key
Stream Cipher
Block Cipher
Public Key
RSA
Integrity and Authentication
Message Authentication Codes
Crypto Hash
Public Key
Digital Signature

Public Key Cryptography

$E_{B+}($ “cs114 is cool” $)$ $D_{B-}()$

Alice

(A^+,A^-)

Bob

(B^+,B^-)

RSA Key Generation

- Choose distinct primes p and q
- Compute n = pq
- Compute $\Phi(n) = \Phi(pq) = (p-1)(q-1)$
- Randomly choose $1<e<\Phi(pq)$ such that e and $\Phi(pq)$ are coprime. e is the **public key exponent**
- Compute $d=e^{-1} \bmod(\Phi(pq))$. d is the **private key exponent**

Example:

let p=3, q=11

n=33

$\Phi(pq)=(3-1)(11-1)=20$

let e=7

$ed \bmod \Phi(pq) = 1$

$7d \bmod 20 = 1$

d = 3

RSA Encryption/ Decryption

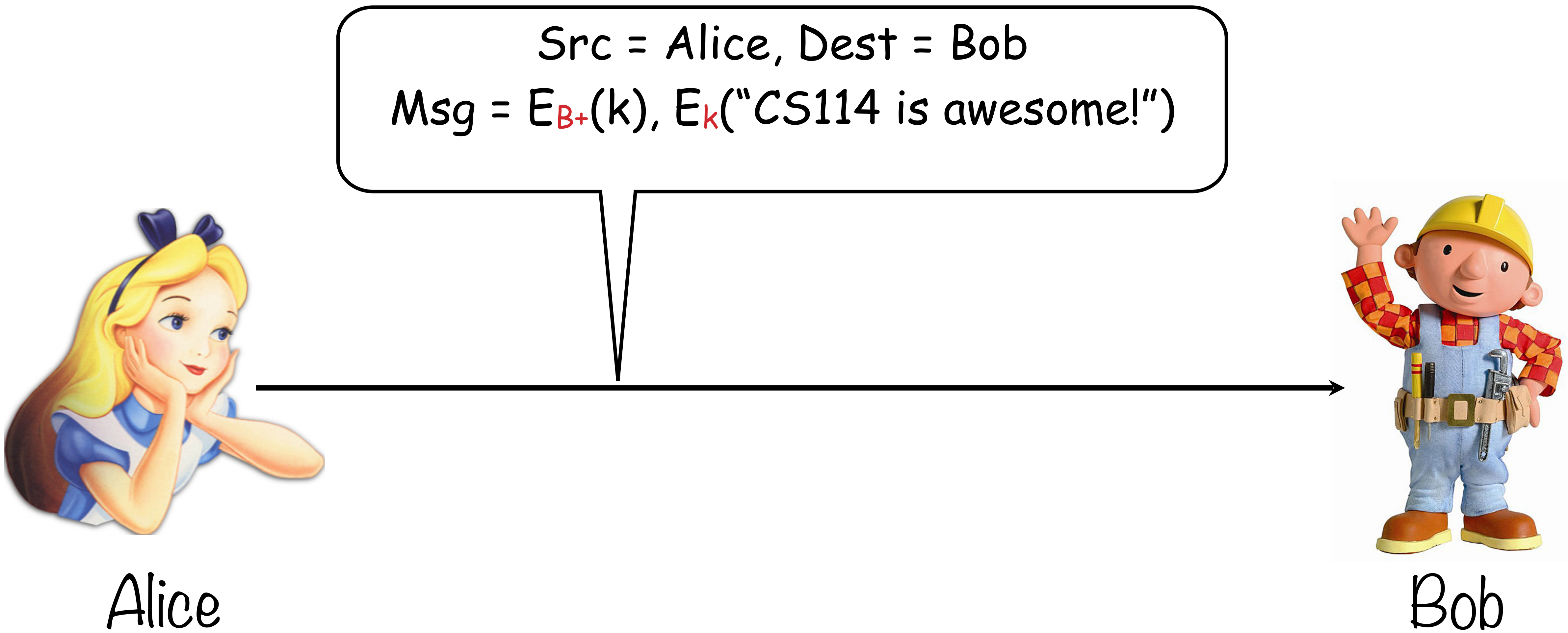
- Public key k^+ is {e,n} and private key k^- is {d,n}
- Encryption and Decryption

$$E_{k+}(M) : \text{ciphertext} = \text{plaintext}^e \bmod n$$

$$D_{k-}(\text{ciphertext}) : \text{plaintext} = \text{ciphertext}^d \bmod n$$

- Example
  - Public key (7,33), Private Key (3,33)
  - Plaintext: 4
  - $E_{\{7,33\}}(4) = 4^7 \bmod 33 = 16384 \bmod 33 = 16$
  - $D_{\{3,33\}}(16) = 16^3 \bmod 33 = 4096 \bmod 33 = 4$

Hybrid Cryptosystems

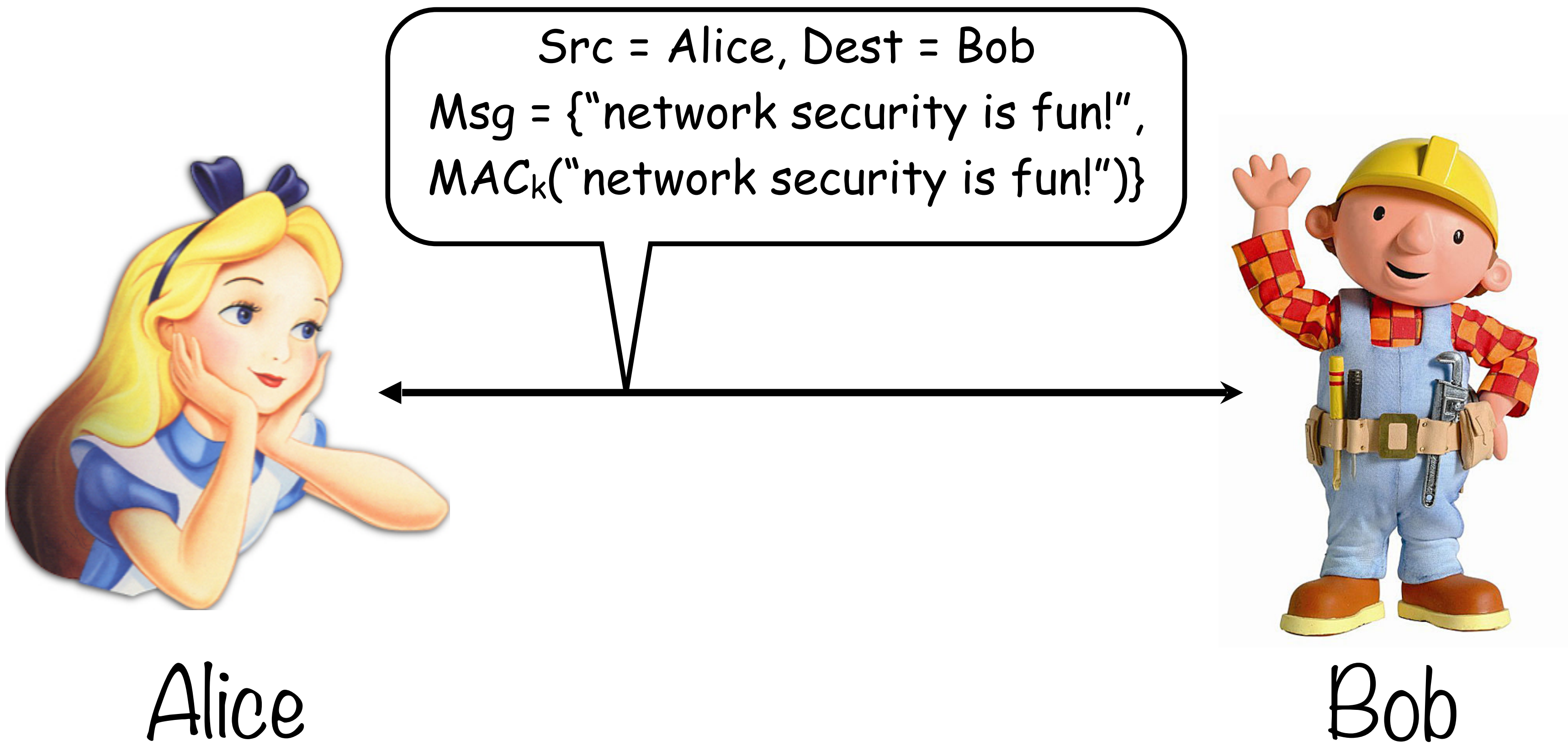


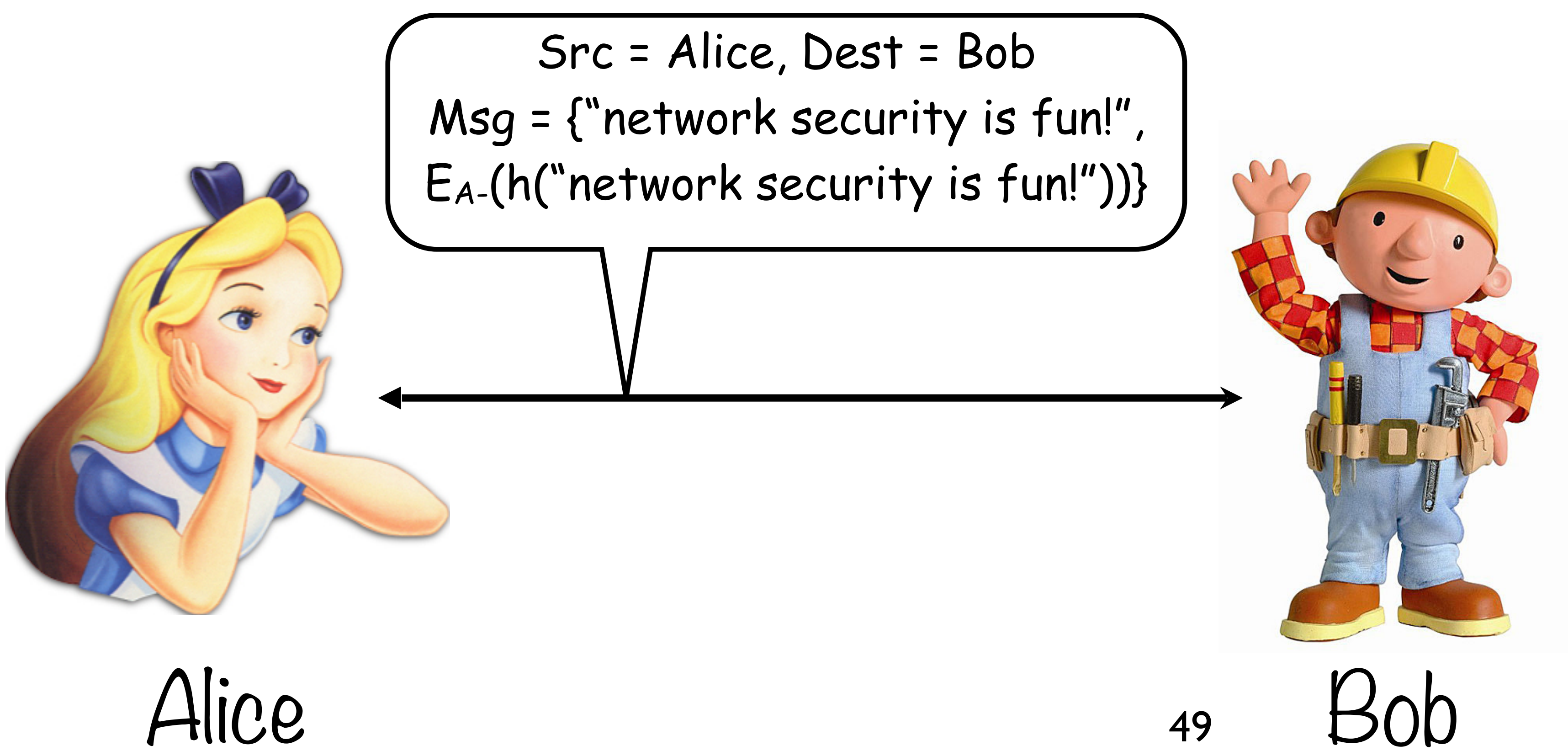
(B^+,B^-) is Bob's long-term public-private key pair.
k is the session key; sometimes called the **ephemeral key**.

How can Alice *sign* a digital document?

- Digital document: M
- Since RSA is slow, hash M to compute digest: $m = h(M)$
- Signature: $Sig(M) = E_{k-}(m) = m^d \bmod n$
  - Since only Alice knows k^-, only she can create the signature
- To verify: Verify(M,Sig(M))
  - Bob computes h(m) and compares it with $D_{k+}(Sig(M))$
  - Bob can compute $D_{k+}(Sig(M))$ since he knows k^+ (Alice's public key)
  - If and only if they match, the signature is verified (otherwise, verification fails)

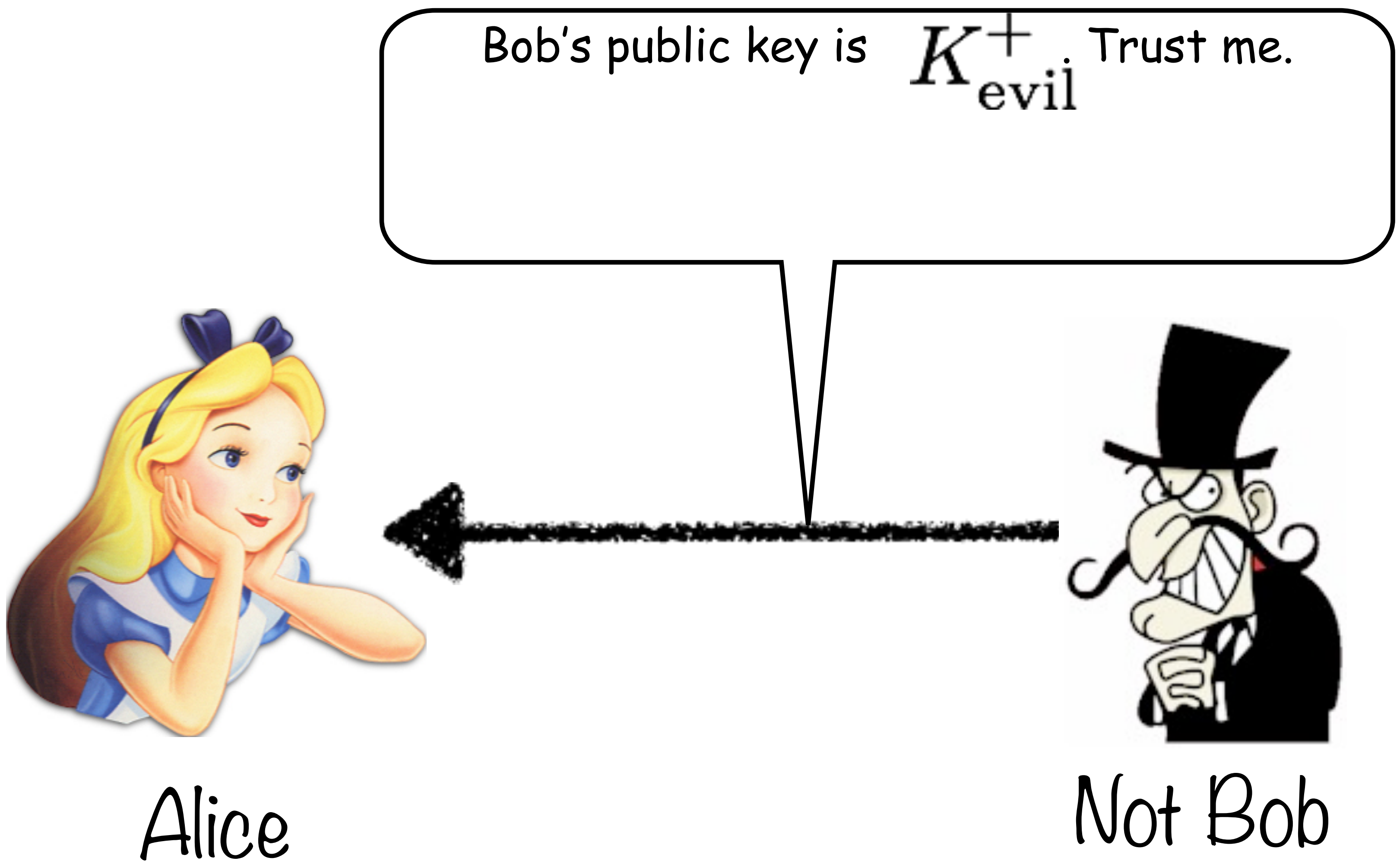
Non-Repudiation





Which of these offer non-repudiation?

But how do we *verify* we're using the correct public key?

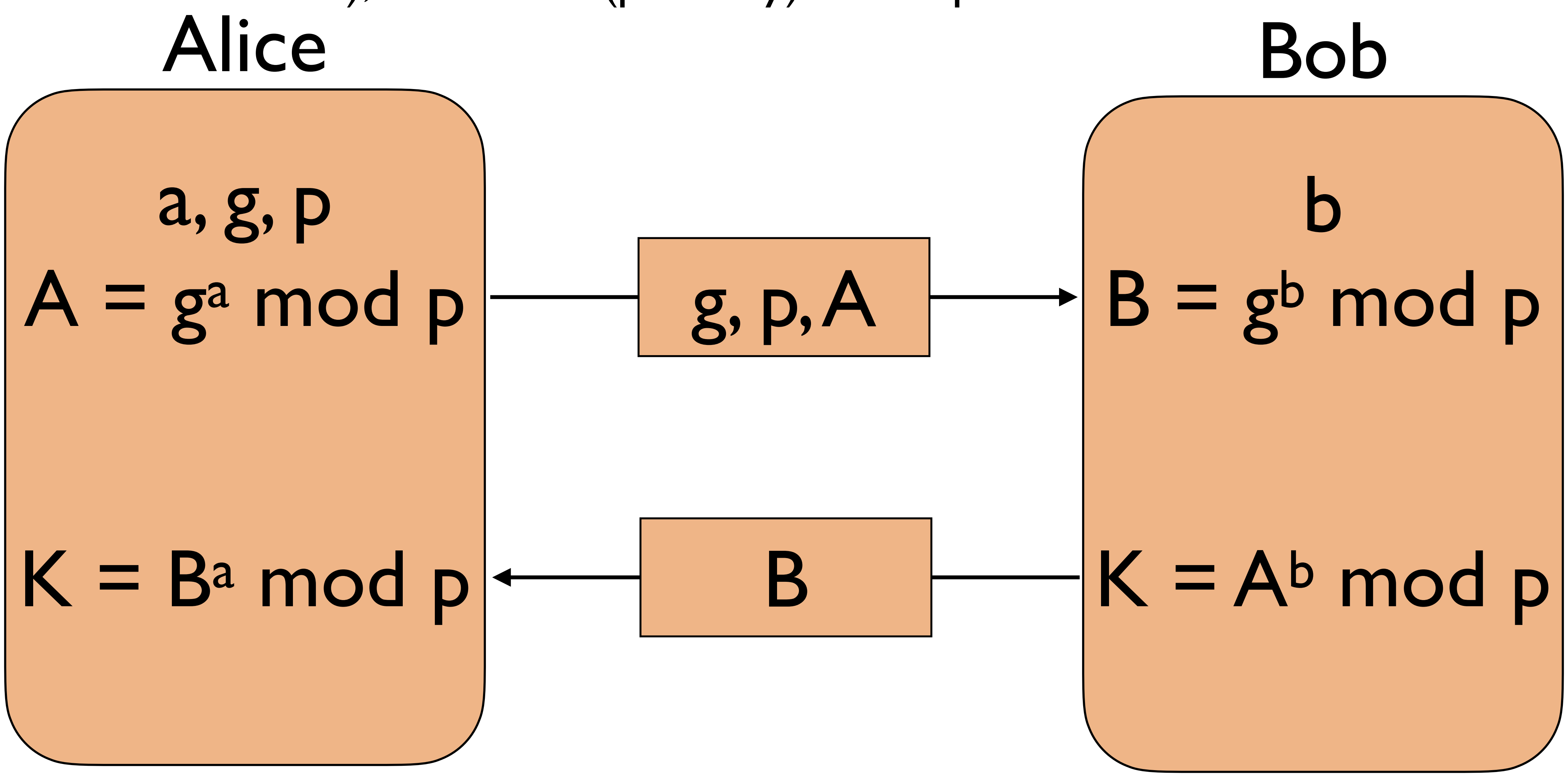


Key Distribution and Key Agreement

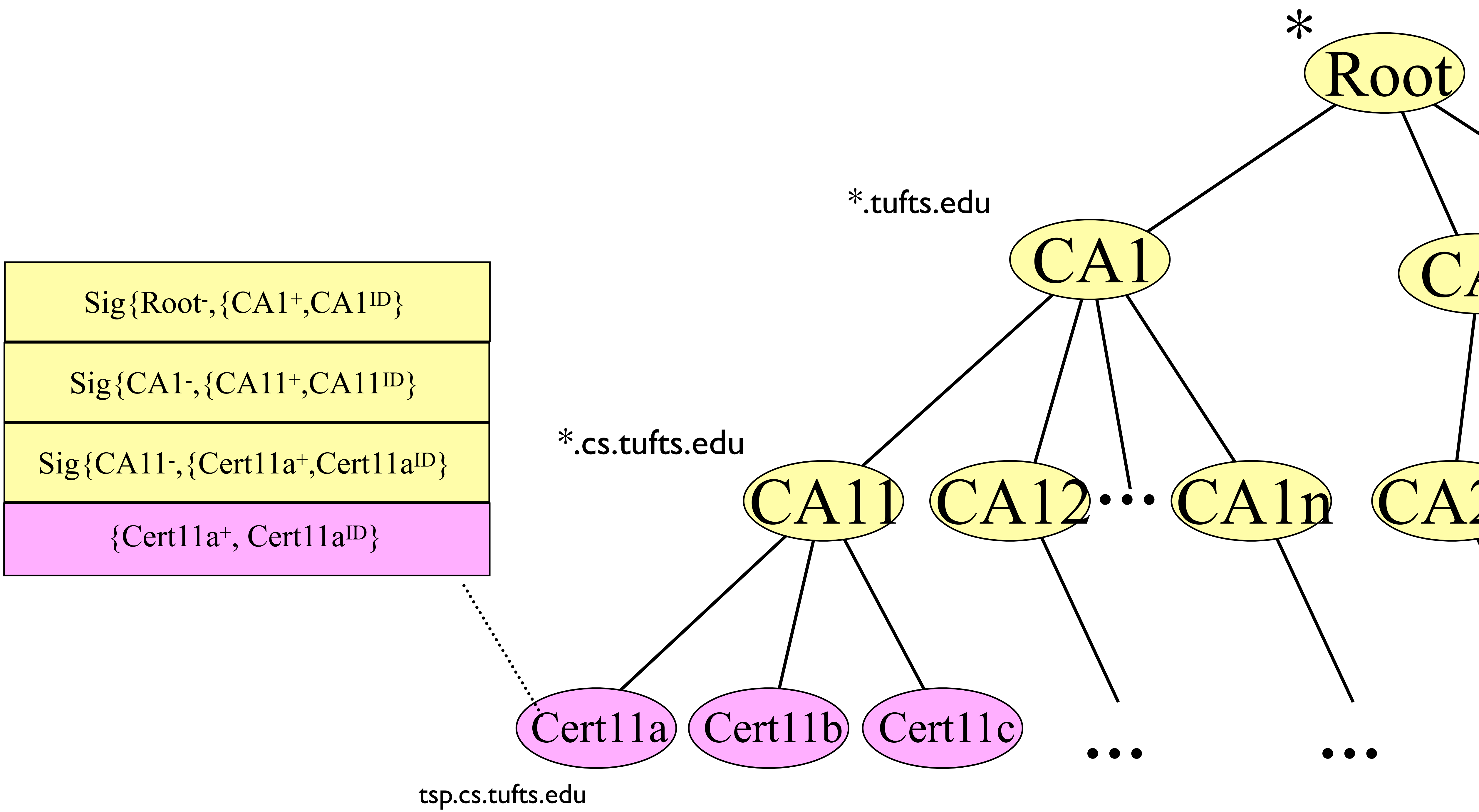
- **Key Distribution** is the process where we assign and transfer keys to a participant
- **Key Agreement** is the process whereby two or more parties negotiate a key

Diffie-Hellman (DH) Key Agreement

- Proposed by Whitfield Diffie and Martin Hellman in 1976
- g=base, p=prime, a=Alice's secret, b=Bob's secret
- Eve cannot compute K without knowing either a or b (neither of which is transmitted), even if she (passively) intercepts all communication!



Certificate Validation



Logistics for Exam 1

- Closed-book, closed-notes, non-collaborative
- You'll have 75 minutes to complete the exam (1:30 - 2:45pm)
- Covers everything from Lecture 2 - 6