CS 352
Public Key Cryptography

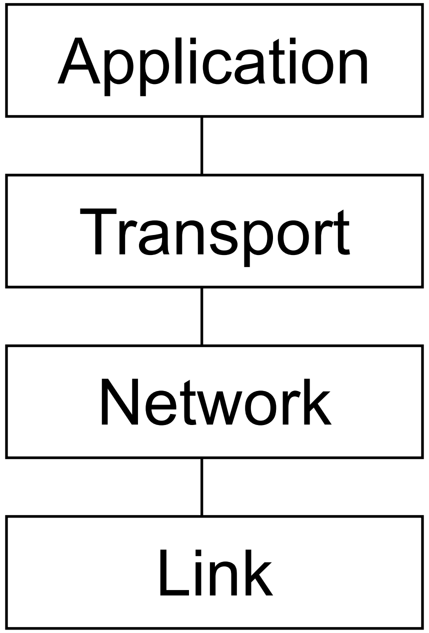
CS 352, Lecture 26.1

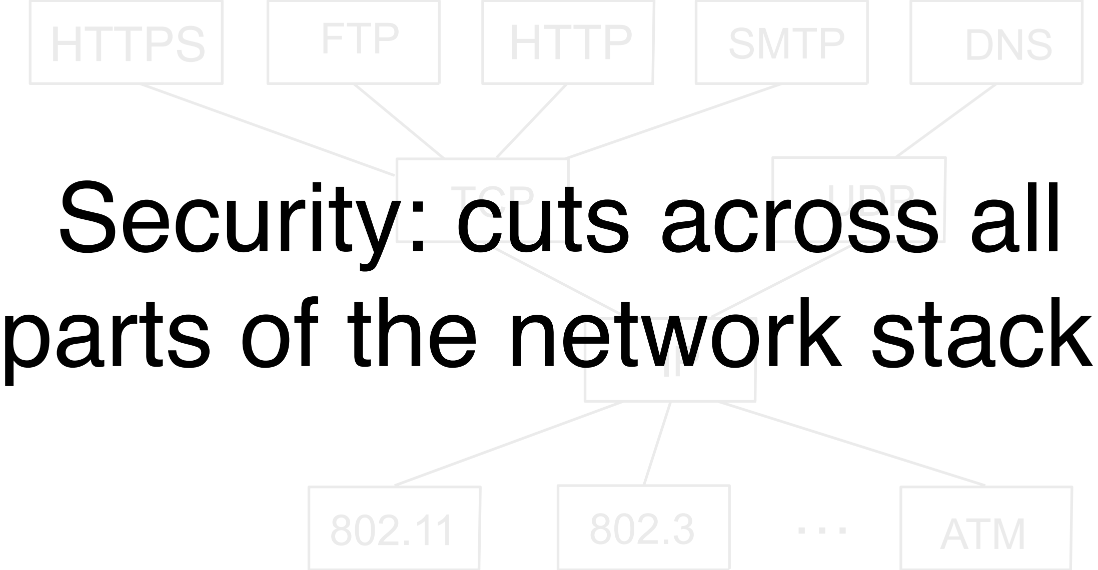
http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

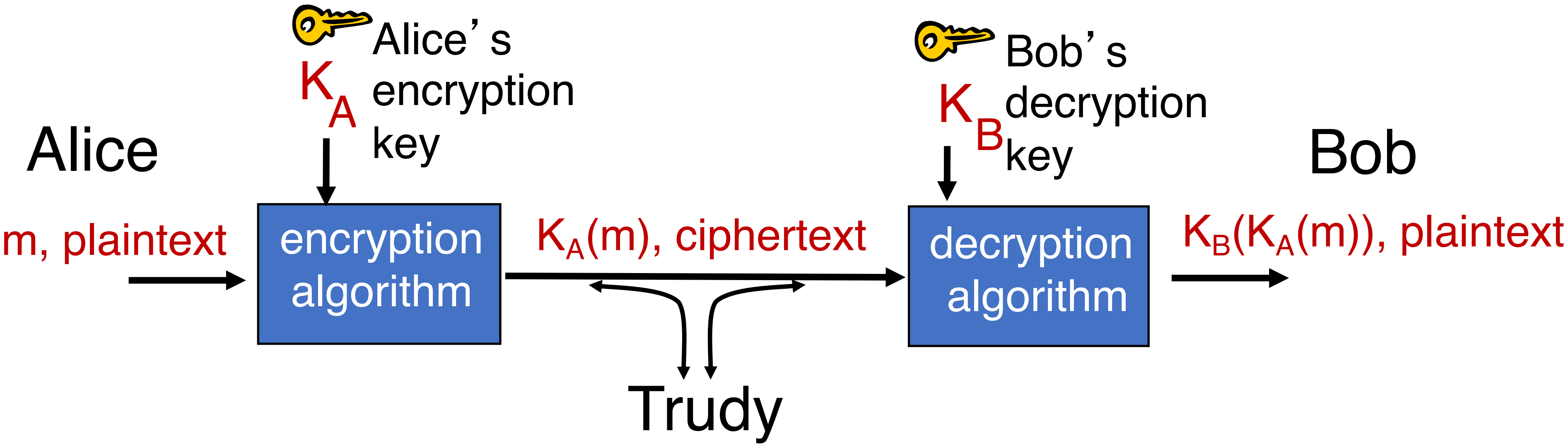
Security and the Network Stack


Application
Transport
Network
Link



Security: cuts across all parts of the network stack!

Review: Cryptography



- K_A and K_B are the same: symmetric key cryptography (last lecture)
- K_A and K_B are different: public key cryptography
  - This lecture!

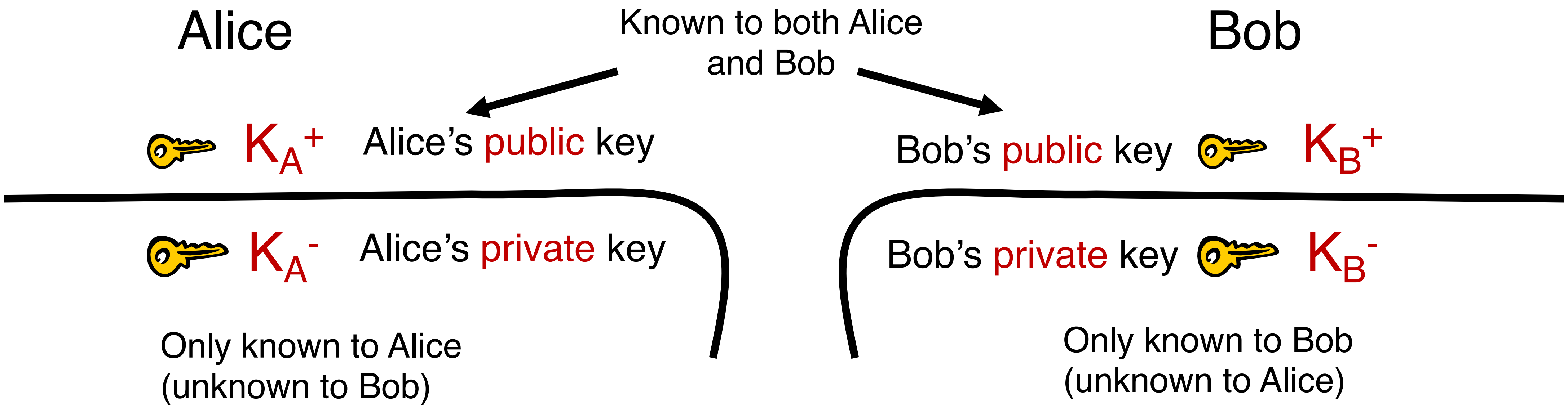
Agreeing on shared secret key is hard

- Communicating parties may never meet in person
  - It’s very common not to meet someone you talk to over the Internet
  - Amazon? Your bank?
- And what if the shared secret is stolen?
  - Must exchange keys securely again!
- Q: how to exchange keys securely over an insecure network?

Use public key cryptography to bootstrap a shared secret key

Terminology

- Alice and Bob each have a pair of keys
- One key is public: the public key is known to all.
  - Assume public keys can be exchanged securely
- The other key is secret to each communicating party: private key



Alice

Known to both Alice and Bob

Bob

K_A^+ Alice's public key

Bob's public key K_B^+

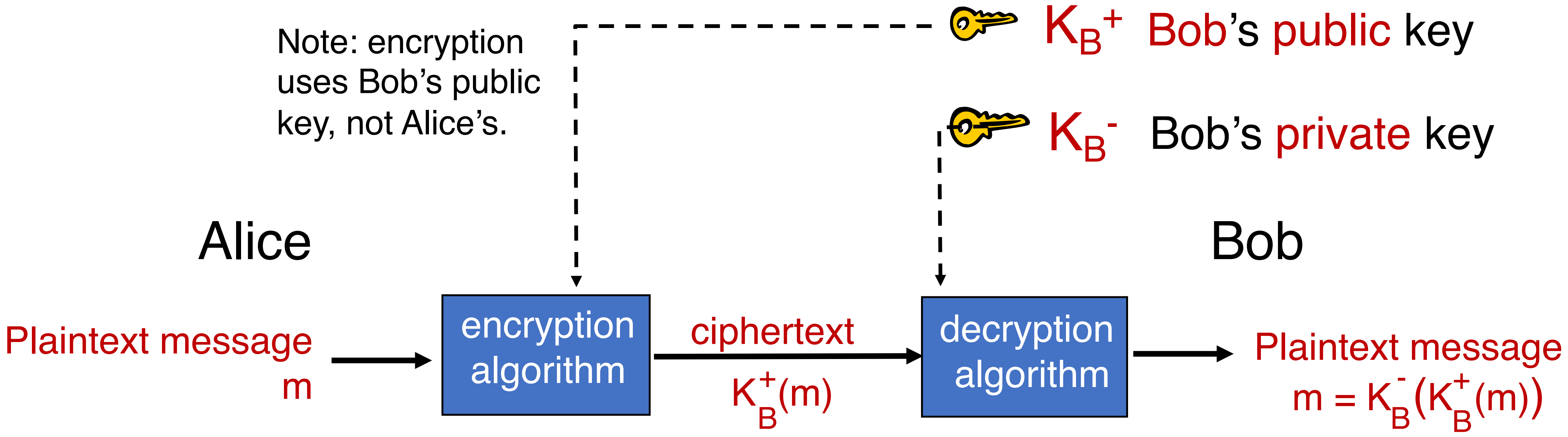
K_A^- Alice's private key

Bob's private key K_B^-

Only known to Alice (unknown to Bob)

Only known to Bob (unknown to Alice)

Public key cryptography



Note: encryption uses Bob's public key, not Alice's.

K_B^+ Bob's public key

K_B^- Bob's private key

Alice

Bob

Plaintext message m

encryption algorithm

ciphertext

$K_B^+(m)$

decryption algorithm

Plaintext message $m = K_B^-(K_B^+(m))$

A message encrypted with Bob's public key can only be decrypted using Bob's private key.
The message cannot be decrypted with Bob's public key.
So, only Bob can decrypt them.

Public key crypto: What do we need?

- Invertible encryption: For each communicating entity, we need algorithms and keys K^+ and K^- such that

$$m = K^-(K^+(m))$$

- Given a public key K^+, it must be intractable to compute the private key K^- (let's call this the one-way property)
- Given ciphertext $K^+(m)$, it must be intractable to compute the plaintext m (confidentiality)
- Sometimes, also authentication/non-repudiation (more later)

$$m = K^+(K^-(m))$$

Public key cryptosystems

- Diffie-Hellman
  - Key distribution, confidentiality
- Digital Signature Algorithm (Schnorr/ElGamal)
  - Authentication and non-repudiation
- Rivest, Shamir, Adleman (RSA)
  - Key distribution, confidentiality, authentication, non-repudiation

Subject of next module

CS 352
The RSA cryptosystem

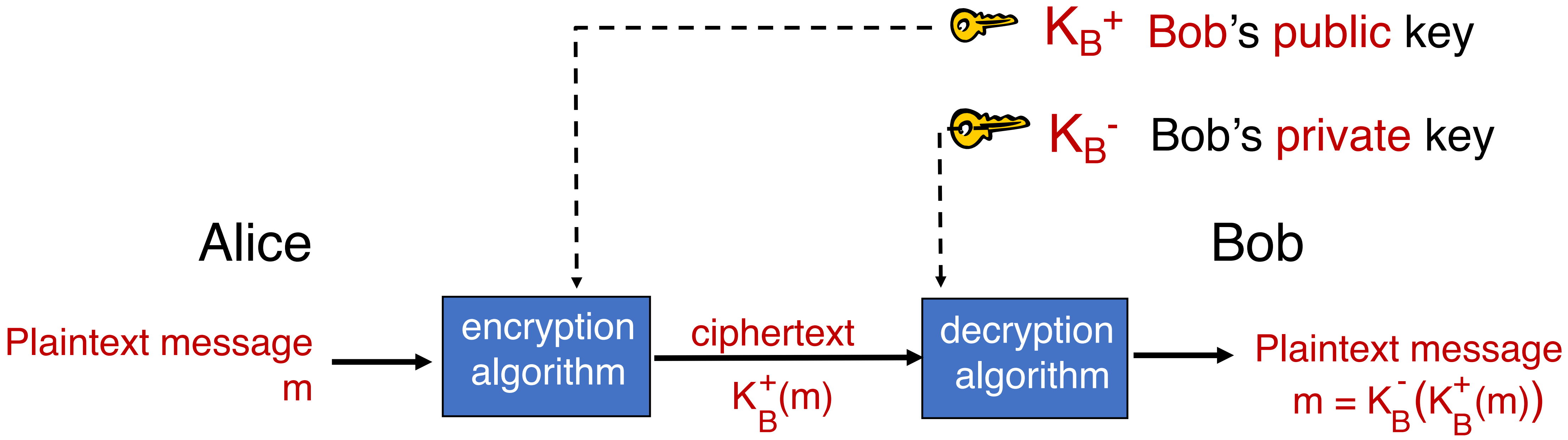
CS 352, Lecture 26.2

http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Review: Public key cryptography



K_B^+ Bob's public key

K_B^- Bob's private key

Alice

Bob

Plaintext message m

encryption algorithm

ciphertext $K_B^+(m)$

decryption algorithm

Plaintext message $m = K_B^-(K_B^+(m))$

Three requirements for a public key cryptosystem:

(1) Invertible encryption: $m = K^-(K^+(m))$

(2) One-way property: intractable to compute K^- from K^+

(3) Confidentiality: intractable to compute m from $K^+(m)$

This module: the RSA cryptosystem

Prerequisite (1): modular arithmetic

- x mod n = remainder of x when divided by n (% operator in C)
- Some facts:

  (x mod n) mod n = x mod n
  [(a mod n) + (b mod n)] mod n = (a+b) mod n
  [(a mod n) * (b mod n)] mod n = (a*b) mod n

- Can use these to show other useful properties:

  $(a \bmod n)^d \bmod n = a^d \bmod n$

- example: x=14, n=10, d=2:
  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

Prerequisite (2): Integer interpretation

- Messages (plaintext and ciphertext) are just bit sequences, can be broken into blocks of fixed length

- Blocks (of fixed length) may be interpreted as integers
  - Algorithms over integers may be applied to message blocks

- Example: suppose m = $10010001_{2.}$ This is 145_{10}
  - It is meaningful to say “we apply modular arithmetic on a message”

RSA Key Generation

1. choose two large prime numbers p, q.
   (e.g., 1024 bits each)
2. compute $n = pq$, and $z = (p-1)(q-1)$
3. choose e (with $e < n$) that has no common factors
   with z (e, z are “relatively prime”).
4. choose d such that $ed-1$ is exactly divisible by z.
   (in other words: $ed \bmod z = 1$).
5. public key is $\underbrace{(n,e)}_{K_B^+}$. private key is $\underbrace{(n,d)}_{K_B^-}$.

RSA Encryption and Decryption

0. given (n,e) and (n,d) (computed during key generation)

1. to encrypt message m $(< n)$, compute

$$c = m^e \bmod n$$

2. to decrypt the received ciphertext, c, compute

$$m = c^d \bmod n$$

An example of RSA

- Bob chooses p=5, q=7. Then n=35, z=24.
- Say e=5 (so e, z relatively prime).
- d=29 (so ed-1 exactly divisible by z).
- Suppose we are encrypting 8-bit messages.

encrypt: $\underbrace{\text{plaintext } m}_{00001100}$ $\underbrace{\text{Int}}_{12}$ $\underbrace{m^e}_{248832}$ $\underbrace{c = m^e \bmod n}_{17}$ Ciphertext

decrypt: $\underbrace{\text{ciphertext } c}_{17}$ $\underbrace{c^d}_{481968572106750915091411825223071697}$ $\underbrace{m = c^d \bmod n}_{12}$ Same plaintext!

RSA satisfies the three requirements

- Given $c = m^e \bmod n$ and $m' = c^d \bmod n$
- Invertible encryption: can show that $m' == m$:
  - i.e., $m == \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$
- Fact: for $n = pq$ and $z = (p-1)(q-1)$, $x^y \bmod n = x^{(y \bmod z)} \bmod n$
- Then $c^d \bmod n = (m^e \bmod n)^d \bmod n$

$$= m^{ed} \bmod n$$

$$= m^{(ed \bmod z)} \bmod n$$

$$= m^1 \bmod n == m$$

RSA satisfies the three requirements

- One-way property: Suppose we know the public key (n, e). How hard is it to determine the private key (n, d)?

- The most viable method that exists is to factor n into p and q, determine z=(p-1)(q-1), then use e to find d, since ed mod z = 1.

- This assumes n can be factored into p and q: no one (publicly) knows efficient algorithms to factor large products of primes (integer factoring problem)

RSA satisfies the three requirements

- Confidentiality: Suppose we know the public key (n, e) and ciphertext c (m^e mod n). How hard is it to find the message m?

- The RSA problem: computing the e’th root of c mod n. The most viable method requires factoring large numbers, for which efficient algorithms are not (publicly) known.

- Note: small numbers can be factored quite effectively
  - Your RSA public and private keys must contain many bits (2048 or more)

RSA can also provide authentication!

- Turns out that $\underbrace{K^{+}(K^{-}(m))}_{\substack{\text{Encrypt with private key} \\ \text{decrypt with public key}}} == \underbrace{K^{-}(K^{+}(m))}_{\substack{\text{Encrypt with public key} \\ \text{decrypt with private key}}}$

- Follows from the rules of modular exponentiation:

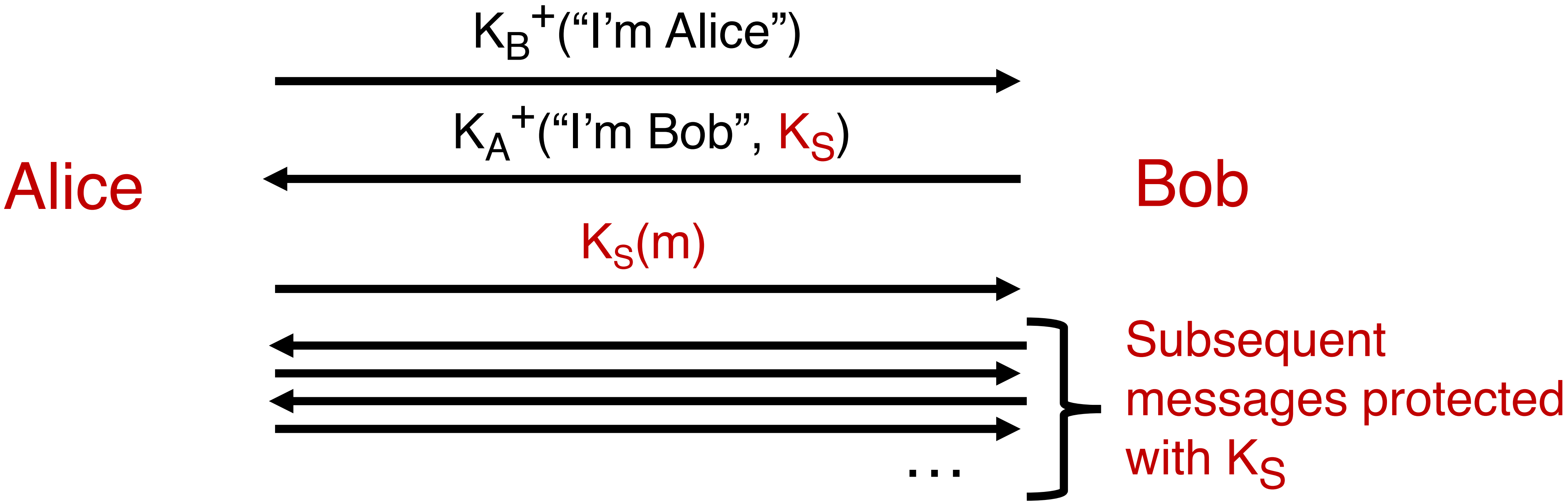
$$\begin{aligned}(m^{e} \bmod n)^{d} \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^{d} \bmod n)^{e} \bmod n\end{aligned}$$

- Next module: we’ll see how to use this for authentication!

RSA is computationally expensive

- Exponentiation in RSA is computationally intensive
- DES (symmetric cipher) is orders of magnitude faster than RSA
- Strategy: use public key crypto to establish a secure connection, then establish second key, a symmetric session key K_S for encrypting and decrypting the data.

Session keys: A simple example



K_B^+(“I’m Alice”)

K_A^+(“I’m Bob”, K_S)

Alice

Bob

K_S(m)

…

Subsequent messages protected with K_S

Use public key crypto to exchange per-session symmetric keys

All further communication occurs with symmetric key crypto

RSA: Summary

- A public key cryptosystem: use a pair of keys, public and private.
    - Only public keys need to be exchanged
- RSA key generation, encryption, and decryption all involve modular arithmetic
- Security guarantees rely on the hardness of factorizing large numbers
- RSA is computationally heavy
    - Use as a method to establish per-session symmetric keys used to encrypt and decrypt data

CS 352
Key Certification

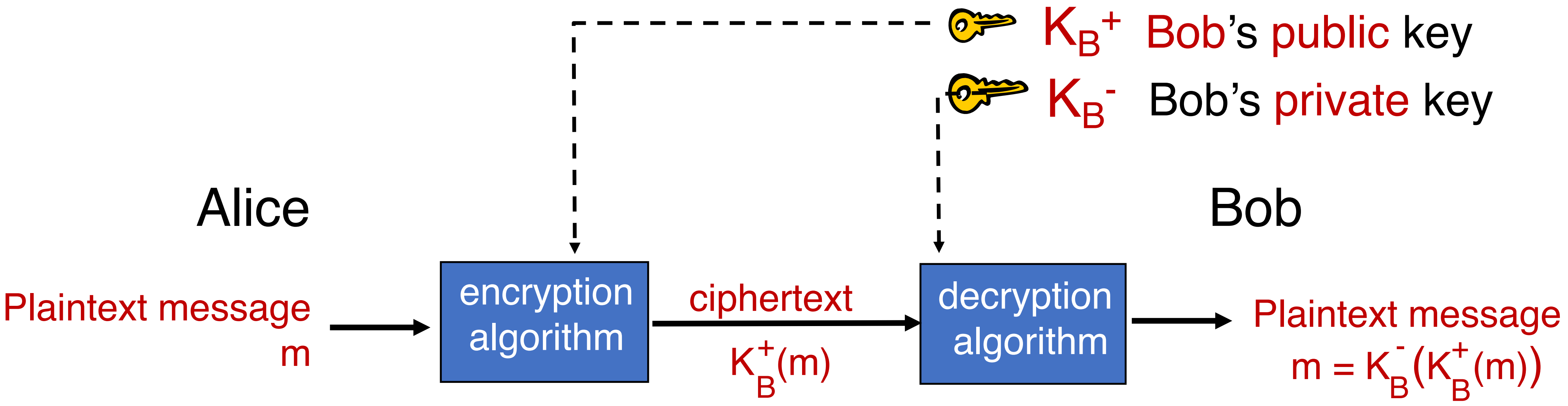
CS 352, Lecture 26.3

http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Review: Public key cryptography



K_B^+ Bob's public key

K_B^- Bob's private key

Alice

Bob

Plaintext message m

encryption algorithm

ciphertext $K_B^+(m)$

decryption algorithm

Plaintext message $m = K_B^-(K_B^+(m))$

Three requirements for a public key cryptosystem:

(1) Invertible encryption: $m = K^-(K^+(m))$

(2) One-way property: intractable to compute K^- from K^+

(3) Confidentiality: intractable to compute m from $K^+(m)$

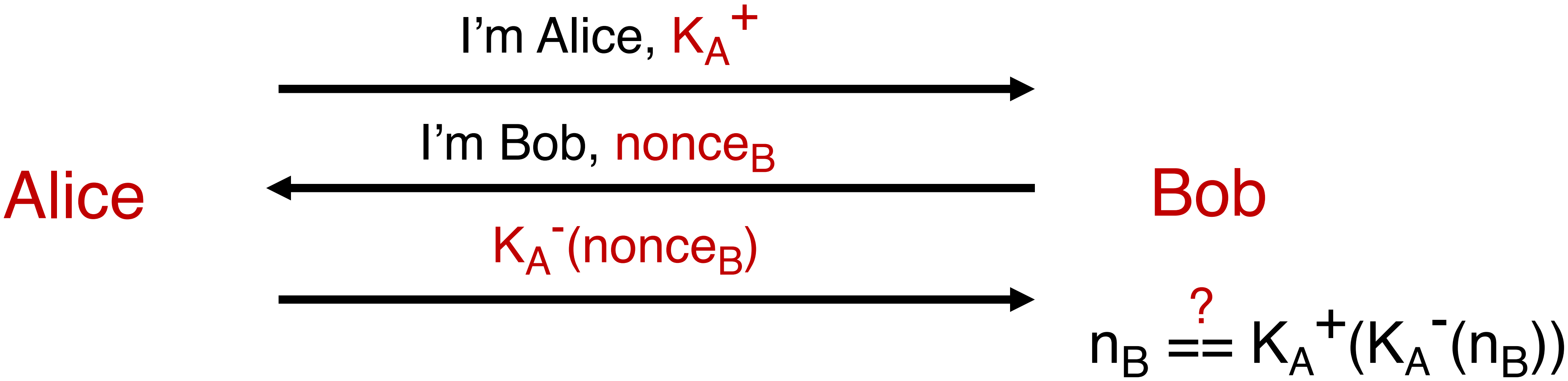
RSA: satisfies all 3, and also, $m = K^+(K^-(m))$ → How to use this for authentication

RSA for authentication: Login system

- Bob runs a login server to provide access to protected resources

- Can Alice and Bob use RSA for authentication, rather than a pre-determined password?

Simple authentication using RSA



I'm Alice, K_A^+

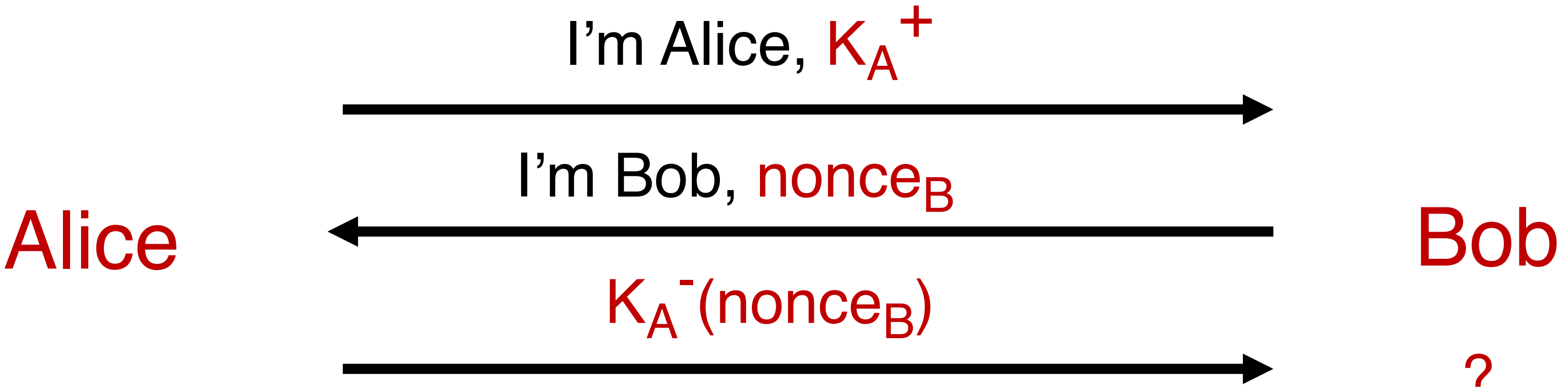
I'm Bob, $nonce_B$

$K_A^-(nonce_B)$

$n_B \overset{?}{==} K_A^+(K_A^-(n_B))$

- Alice sends her public key to Bob
- Bob sends a nonce
- The nonce is the challenge that Alice must use to show that she holds the private key corresponding to the public key
- Alice responds with the nonce encrypted with Alice's private key
  - Bob can decrypt the nonce using Alice's public key to check its validity

Simple authentication using RSA



Alice

I'm Alice, K_A^+

I'm Bob, nonce_B

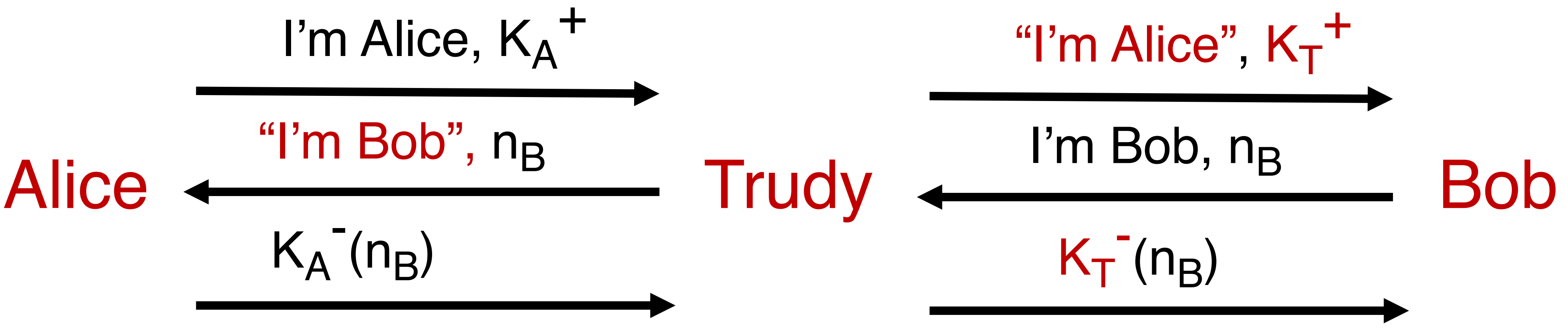
$K_A^-(\text{nonce}_B)$

Bob

$$n_B \overset{?}{==} K_A^+(\underbrace{K_A^-(n_B)})$$

- Assume only Alice holds Alice's private key
  - Only Alice could have encrypted Bob's nonce with Alice's private key.
- So Bob can authenticate Alice to use server resources
- Do you see a problem?

Bad to exchange keys insecurely!



Alice

I'm Alice, K_A^+

"I'm Bob", n_B

$K_A^-(n_B)$

Trudy

"I'm Alice", K_T^+

I'm Bob, n_B

$K_T^-(n_B)$

Bob

Indeed $n_B = K_T^+(K_T^-(n_B))$

- Trudy can perform an entity-in-the-middle attack!
- Trudy pretends to be Alice to Bob and Bob to Alice
- Bob thinks K_T^+ is Alice's key, Alice thinks Bob is sending nonce
- Problem exists even if Bob encrypts nonce with Alice's public key

One cannot “just trust” public keys

- Suppose Alice sends Bob K_A^+, Bob sends Alice K_B^+

Alice

I’m Alice, K_A^+

I’m Bob, K_T^+

Trudy

I’m Alice, K_T^+

I’m Bob, K_B^+

Bob

- Every message can be decrypted and re-encrypted by Trudy
  - Including symmetric session keys that might be exchanged

- Trudy can evade detection completely: plaintext received by Alice and Bob are identical to those without the attack

You can't just trust a public key, since it is unclear whether the key is tied to the entity you're talking to.
(it's like someone calling you and claiming they're from your bank or the IRS)

Q: Is there a way to reliably know the public key of an entity you're communicating with?

Key certification

- Trust someone else (a centralized authority) to check public keys for us

- On the Internet, and in real life, trust is transitive
  - If X trusts Y, and Y trusts Z, then X can trust Z

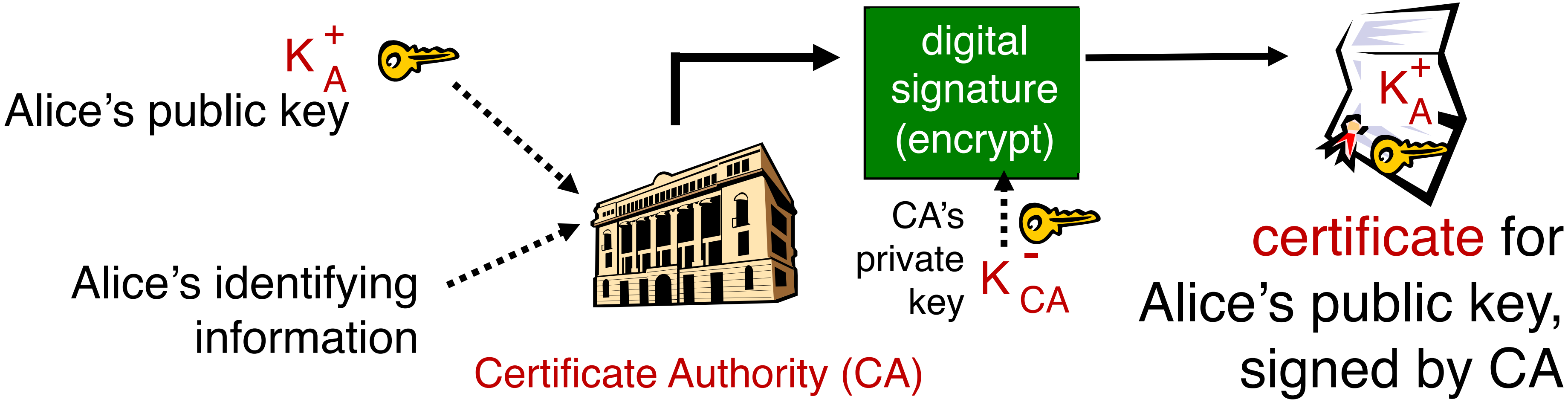
- E.g., Bob trusts a key certification authority (CA)
- The certification authority trusts Alice's public key
- Hence, Bob can trust Alice's public key

Certificate Authority

- Certificate authority (CA): binds a public key to particular entity
  - Analogy: the Department of Motor Vehicles binds your details (name, address, age, etc.) to your identity (social security) after checking them
- Entity E (e.g., web site, router) registers its public key with CA
  - E provides proof of its identity to the CA
- CA creates a certificate binding E to its public key
  - Analogy: the driver's license is your certificate

Certificate Authority

- The CA uses a mechanism called a digital signature to perform this binding in an unforgeable manner
  - We’ll learn about digital signatures in the next lecture
- Effectively, the CA attests “this is E’s public key”
- Checking the signature requires the CA’s public key
  - For the web, this key is shipped with your browser installation



K_A^+

Alice’s public key

Alice’s identifying information

digital signature (encrypt)

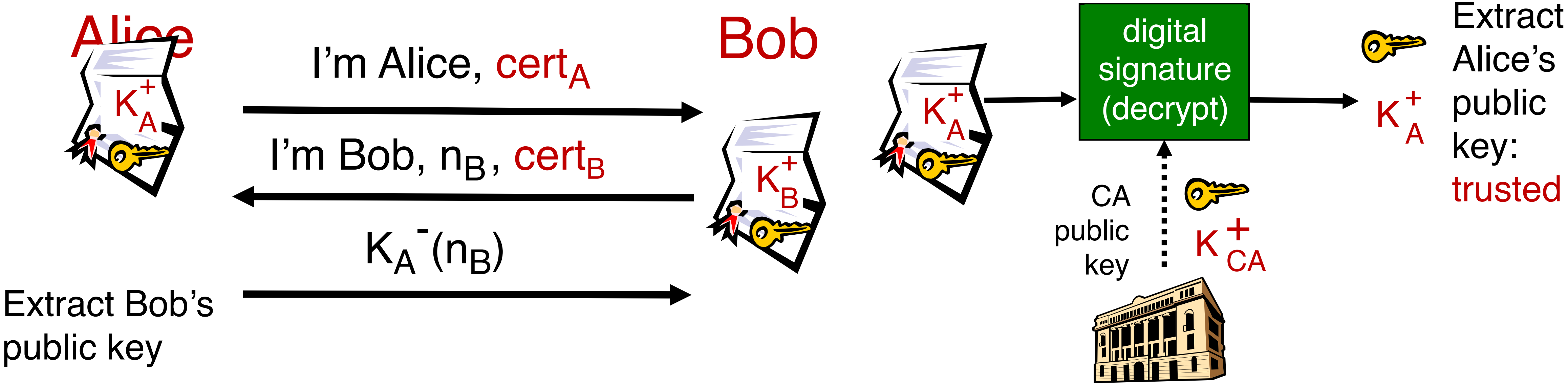
CA’s private key K_{CA}^-

Certificate Authority (CA)

K_A^+

certificate for Alice’s public key, signed by CA

Authentication using certificates



Alice

I'm Alice, cert_A

I'm Bob, n_B, cert_B

$K_A^-(n_B)$

Extract Bob's public key

Bob

digital signature (decrypt)

CA public key

K^+_{CA}

Extract Alice's public key: K^+_A trusted

- When Alice authenticates herself, she sends Bob her certificate
    - Bob extracts Alice's public key using the certificate and CA's public key
- If needed, Bob can authenticate himself to Alice using his cert
- It is possible for Bob to start trusting Alice's public key using other methods: e.g., a web of trust, like PGP

Summary of key certification

- Exchanging public keys over an insecure channel is bad
    - Need a way to bind a public key to an entity in a trustworthy manner
- Certificate authorities bind public keys to entities
    - Mechanism of digital signatures (next lecture)
    - Need the CA’s public key to extract the entity’s public key
- Extracted public key can then be used to challenge the communicating entity, e.g., through nonces

Public Key Cryptography: Summary

- Public key cryptography is powerful
  - No need to exchange secret keys securely
  - Only the receiver of encrypted information holds the secret key
  - Public keys are exactly that: public!
  - Useful as a mechanism to exchange symmetric keys later on
- Crypto algorithms fundamentally support Internet security
  - Algorithms like AES and RSA are used widely on servers
  - HTTPS uses these ciphers (more later)
- Next lecture: use crypto as building block for integrity and non-repudiation