Distributed Systems

24. Cryptographic Systems: A Brief Introduction

Paul Krzyzanowski

Rutgers University

Fall 2018

Cryptography ≠ Security

Cryptography may be a component of a secure system

Adding cryptography may not make a system secure

Cryptography: what is it good for?

- Confidentiality
  - others cannot read contents of the message
- Authentication
  - determine origin of message
- Integrity
  - verify that message has not been modified
- Nonrepudiation
  - sender should not be able to falsely deny that a message was sent

Confidentiality

Encryption

Plaintext (cleartext) message P

Encryption *E*(P)

Produces Ciphertext, C = *E*(P)

Decryption, P = *D*(C)

Cipher = cryptographic algorithm

Terms: types of ciphers

- Symmetric algorithm
  - Shared key

    $C = E_K(P) \quad P = D_K(C)$
  - Key length → difficulty of attack

- Public key algorithm
  - Key pair: private key (k) & a shared public key (K)

    $C = E_k(P) \quad P = D_K(C)$

    $C = E_K(P) \quad P = D_k(C)$

Key distribution

Secure key distribution is the biggest problem with symmetric cryptography

Distributing Keys

- **Manual: pre-shared keys**
  - Initial configuration, out of band (send via USB key, recite, …)
- **Trusted third party**
  - Knows all keys
  - Alice creates a **session key**
  - Encrypts it with her key – sends to Trent
  - Trent decrypts it and sends it to Bob
  - Alternatively: Trent creates a session key – encrypts it for Alice & for Bob
- **Public key cryptography**
  - Alice encrypts a message with Bob's public key
  - Only Bob can decrypt
- **Diffie-Hellman**
- **Hybrid cryptosystems**

Diffie-Hellman Key Exchange

Key distribution algorithm

- First algorithm to use public/private "keys"
- *Not* public key encryption
- Uses a **one-way function**
  Based on difficulty of computing discrete logarithms in a finite field compared with ease of calculating exponentiation
- Compute
  common key = *f(your_private_key, their_public_key)*
- Eavesdroppers cannot compute this

Hybrid Cryptosystems

- Session key: randomly-generated key for one communication session
- Use a public key algorithm to send the session key
- Use a symmetric algorithm to encrypt data with the session key

Public key algorithms are almost never used to encrypt messages

– MUCH slower; vulnerable to *chosen-plaintext attacks*

– RSA-2048 approximately 55x slower to encrypt and 2,000x slower to decrypt than AES-256

Message Integrity

Hash functions

- **Cryptographic hash function** (also known as a digest)
  - Input: arbitrary data
  - Output: fixed-length bit string
- Properties of a cryptographic hash, *H=hash(M)*:
  - **One-way function**
    - Given *H*, it should be difficult to compute *M*
  - **Collision resistant**
    - Given *H,* it should be difficult to find *M’,* such that *H=hash(M’)*
    - For a hash of length L, a perfect hash would take $2^{(L/2)}$ attempts
  - **Efficient**
    - Computing *H* should be computationally efficient

Message Authentication Codes vs. Signatures

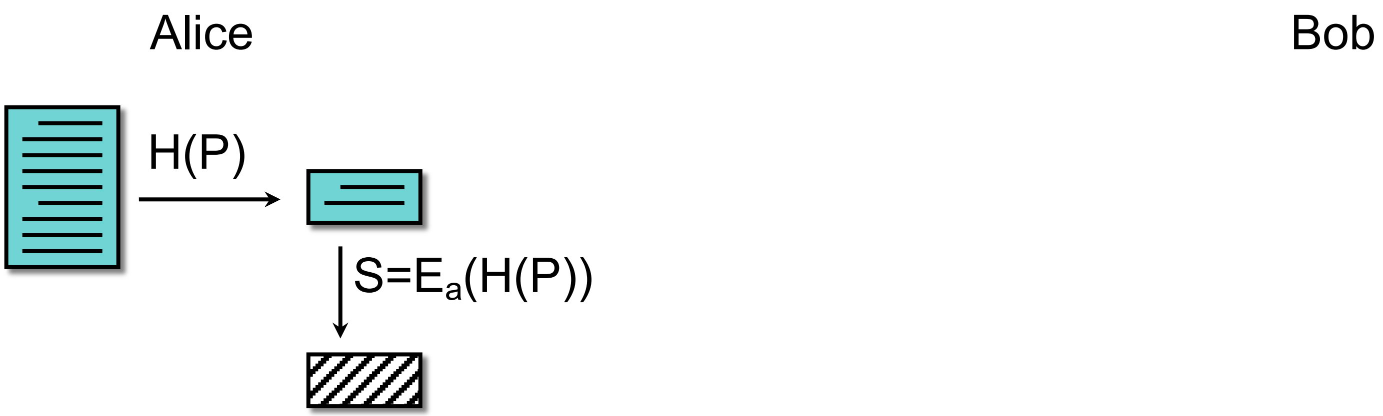
- **Message Authentication Code (MAC)**
  - Hash of message encrypted with a symmetric key:
    An intruder will not be able to replace the hash value

- **Digital Signature**
  - Hash of message encrypted with the owner's private key
    - Alice encrypts the hash with her private key
    - Bob validates it by decrypting it with her public key & comparing with *hash(M)*
  - Provides non-repudiation: recipient cannot change the encrypted hash

Digital signatures: public key cryptography

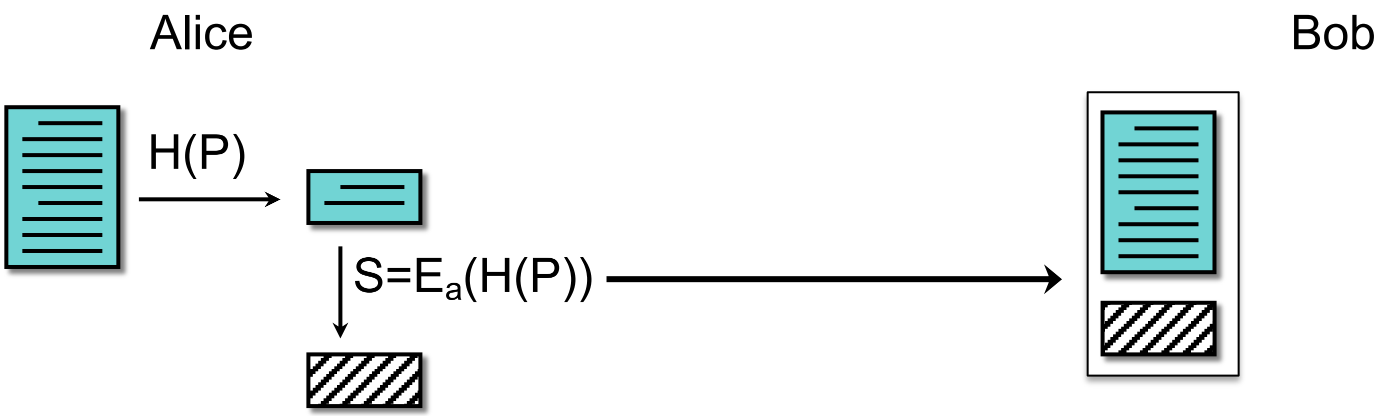
Alice generates a hash of the message

Digital signatures: public key cryptography



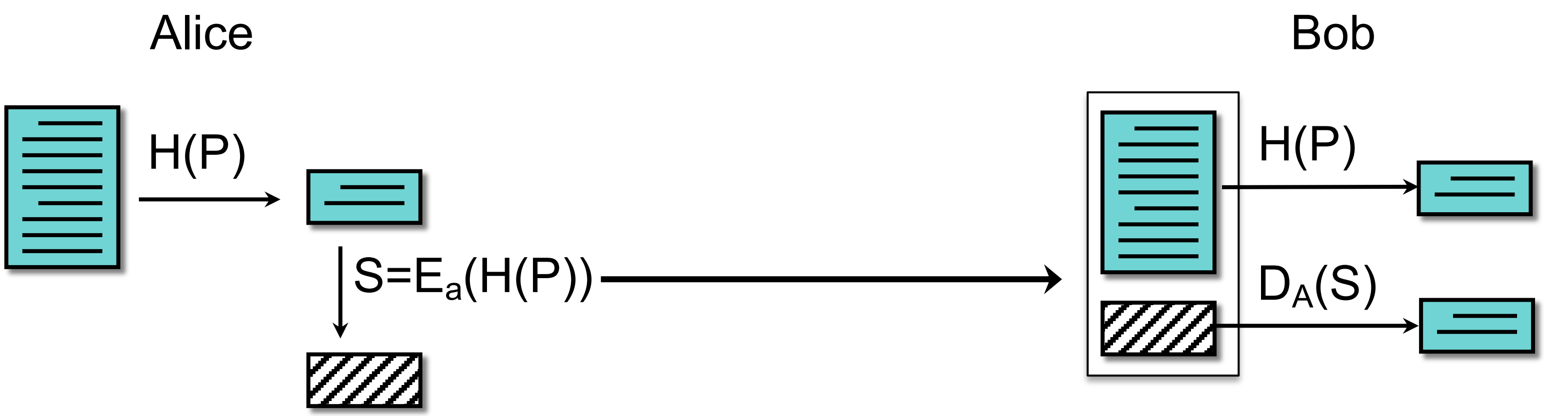
Alice encrypts the hash with her private key
This is her **signature**.

Digital signatures: public key cryptography



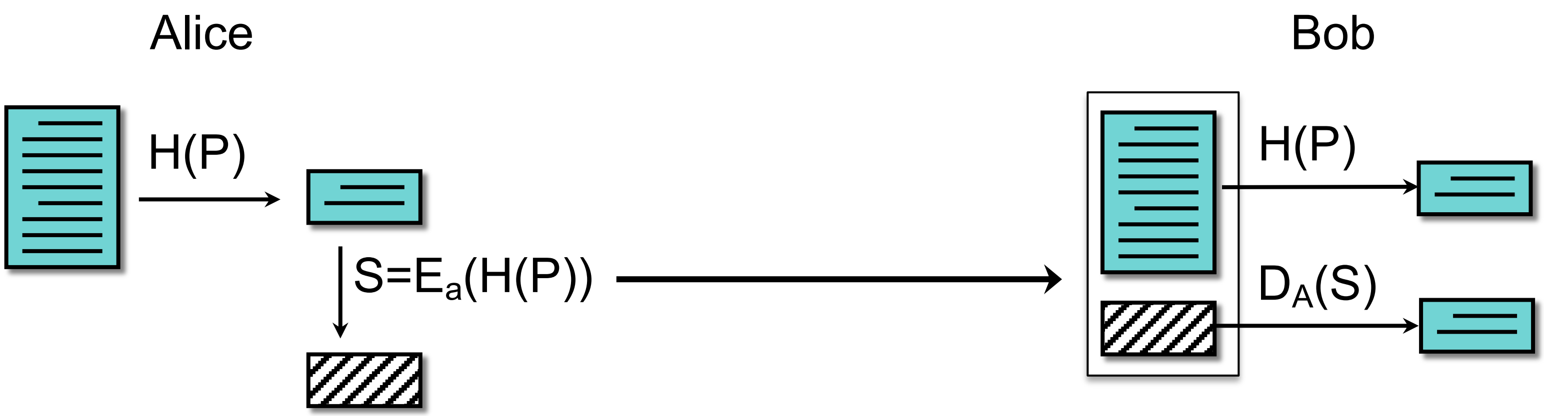
Alice sends Bob the message & the encrypted hash

Digital signatures: public key cryptography



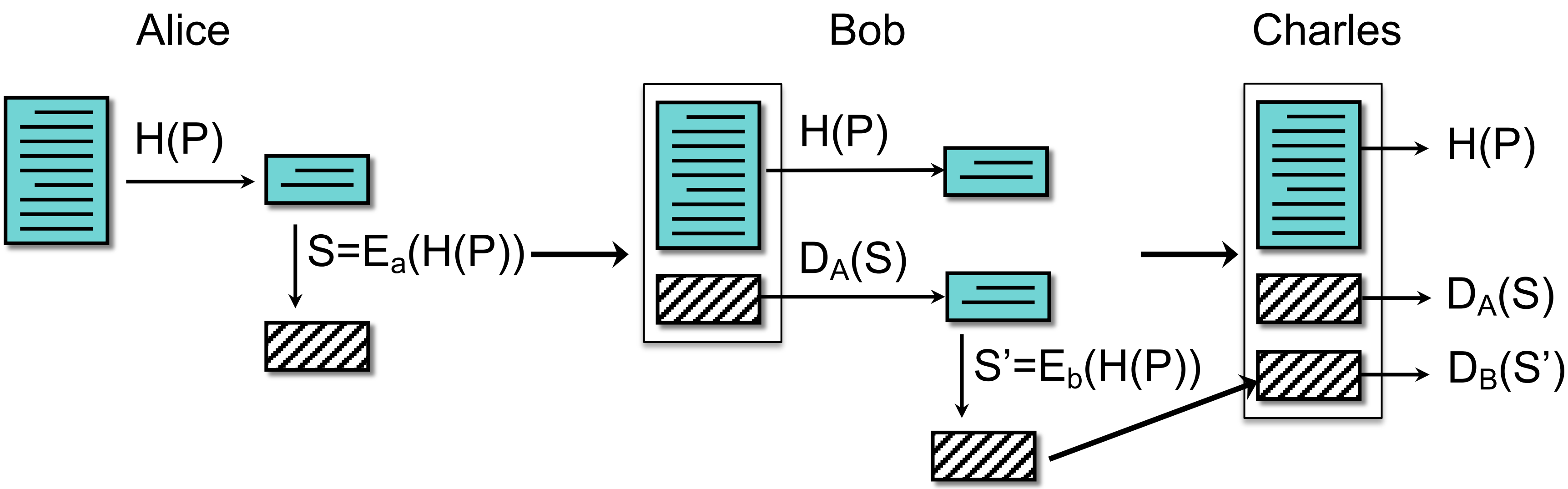
1. Bob decrypts the hash using Alice's public key
2. Bob computes the hash of the message sent by Alice

Digital signatures: public key cryptography



If the hashes match, the signature is valid
– the encrypted hash *must* have been generated by Alice

Digital signatures: multiple signers



Charles:

- Generates a hash of the message, H(P)
- Decrypts Alice's signature with Alice's public key
  - Validates the signature: $D_A(S) \stackrel{?}{=} H(P)$
- Decrypts Bob's signature with Bob's public key
  - Validates the signature: $D_B(S) \stackrel{?}{=} H(P)$

Covert AND authenticated messaging

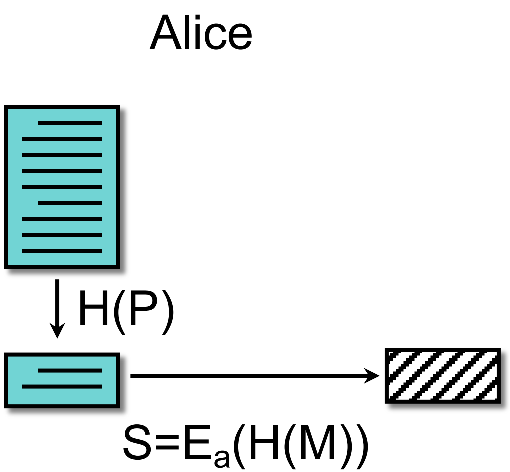
If we want to keep the message secret

- combine encryption with a digital signature

Use a session key:

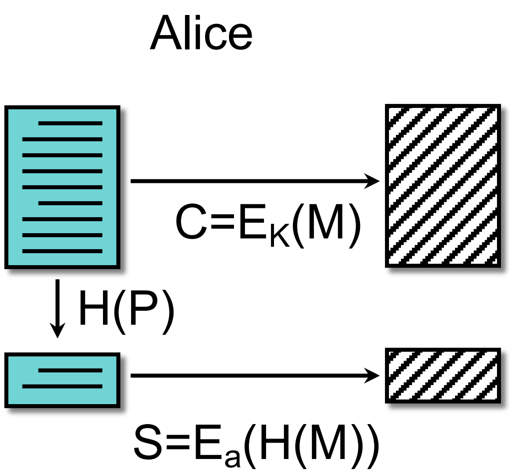
- Pick a random key, *K*, to encrypt the message with a symmetric algorithm
- encrypt *K* with the public key of each recipient
- for signing, encrypt the hash of the message with sender's private key

Covert and authenticated messaging



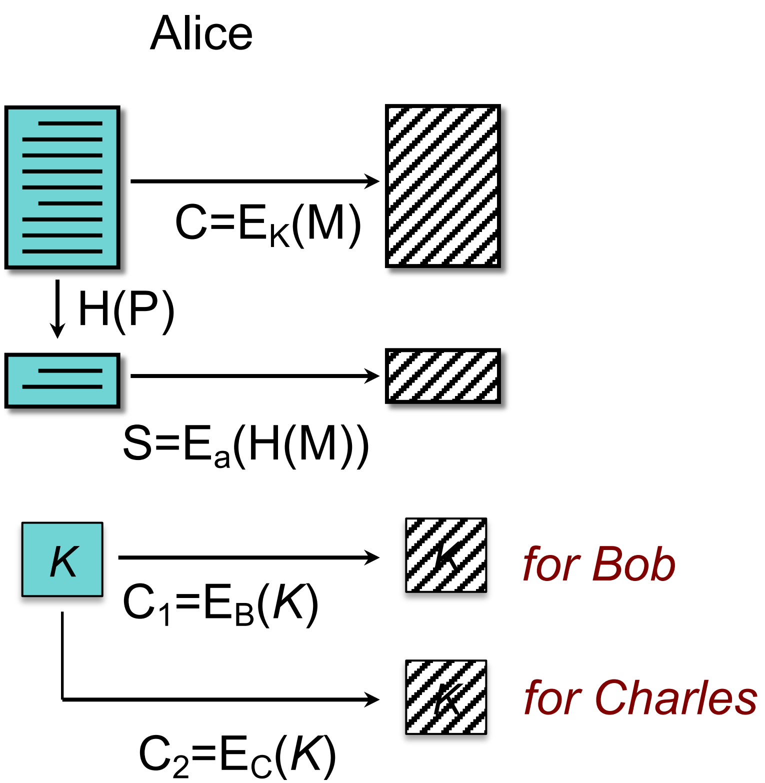
Alice generates a digital signature by
encrypting the message with her private key

Covert and authenticated messaging



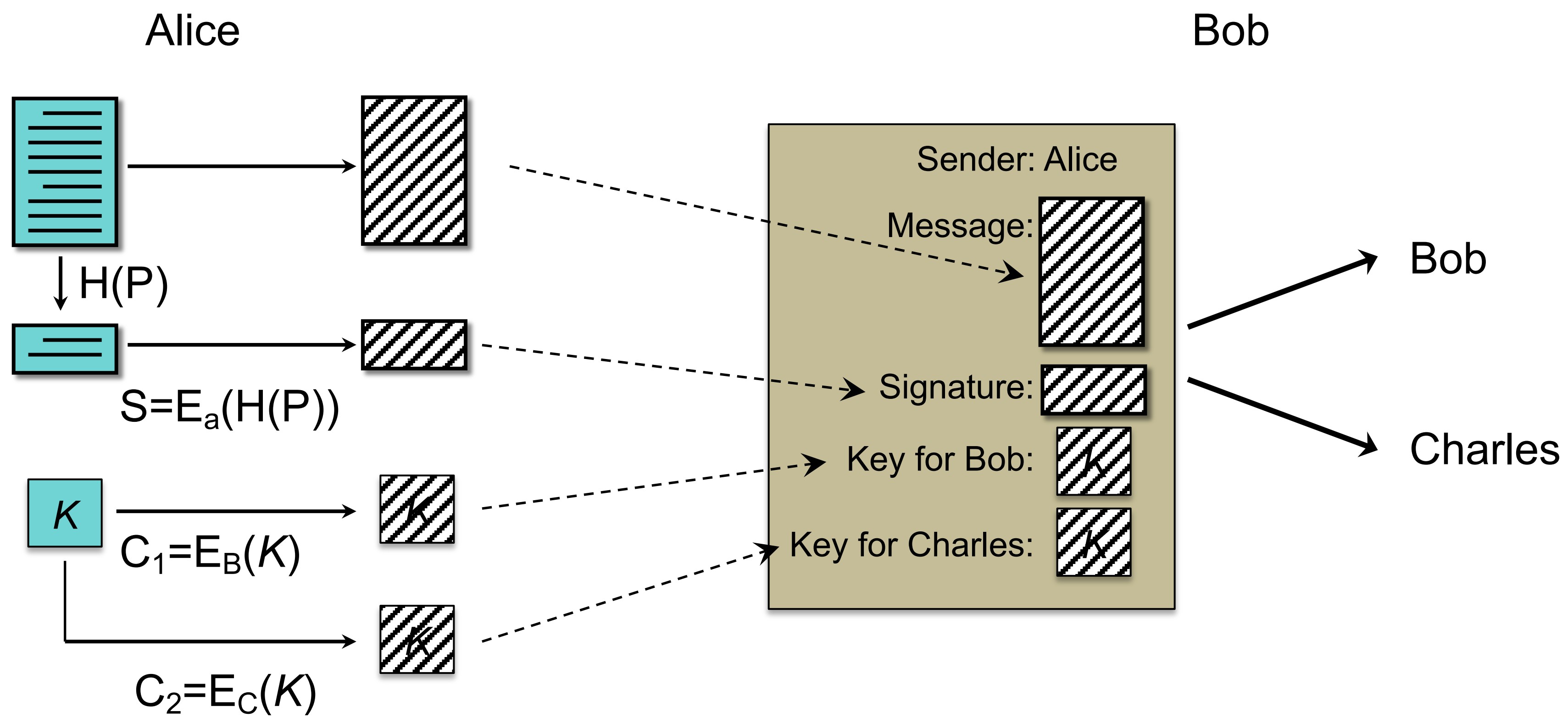
Alice picks a random key, *K*, and encrypts the message *P* with it using a symmetric cipher

Covert and authenticated messaging



Alice encrypts the session key for each recipient of this message using their public keys

Covert and authenticated messaging



The aggregate message is sent to Bob & Charles

The end