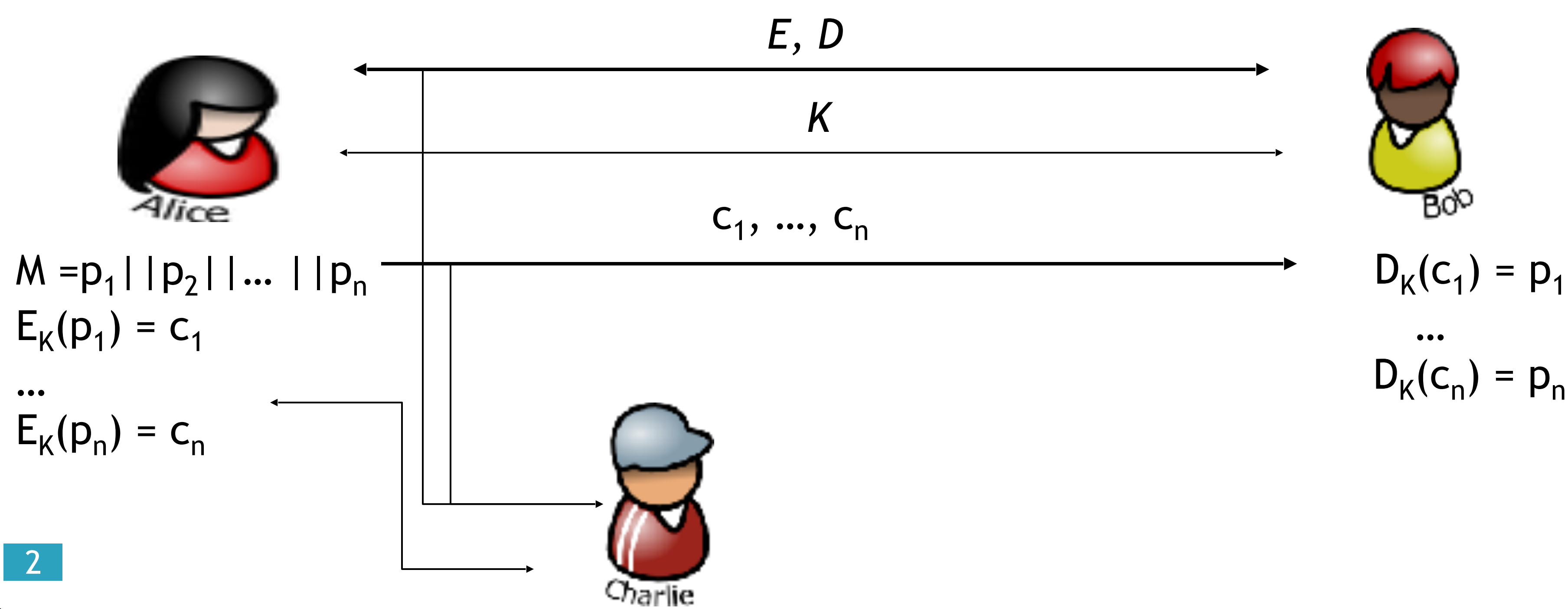
Cryptographic Primitives II

IND games, experiments

Indistinguishability against Chosen Plaintext Attack (IND-CPA)

- Charlie given: ciphertext, E,D, plaintext chosen by Charlie, corresponding ciphertext
- Charlie goal: Guess encryptions of plaintexts other than $(p_1,\ldots p_n)$



Indistinguishability against Chosen Plaintext Attack (IND-CPA)

- Give Charlie: $(c_1,\ldots,c_n)$, E,D, $(\ (p_1,c_1), (p_2,c_2), \ldots, (p_n,c_n)\ )$
- Charlie could pick p_i as a function of $((p_1,c_1), (p_2,c_2), \ldots, (p_{i-1},c_{i-1}))$ for $i = 1 \ldots n$ (static vs. adaptive)
- Charlie is tasked with guessing encryptions of plaintexts not in $(p_1,\ldots p_n)$, e.g., Alice later sends an encrypted file to Bob, …
- Minimum security requirement for encryption algorithms

Indistinguishability against Chosen Plaintext Attack (IND-CPA) Game

- IND-CPA game
- Query phase:
  - Charlie is given (E,D)
  - Charlie queries Alice n times (polynomially bounded) for encryptions of $p_1,..p_n$
  - Alice gives Charlie responses $c_1=E_K(p_1)$,… $c_n=E_K(p_n)$
- Challenge-response phase:
  - Charlie chooses messages (m_0,m_1), gives to Alice
  - Alice encrypts and returns $C_b = E_K(m_b)$

Indistinguishability against Chosen Plaintext Attack (IND-CPA) Game

- Charlie can query Alice again on n plaintexts (even m_0,m_1)! (why is this ok?!) (in adaptive model, not static)
- Finally Charlie returns his guess b' for b
- Charlie wins if $Pr[(b==b')] >> 1/2 + negl(\varepsilon)$

Chosen Plaintext Attack Example

1. Alice is an e-mail program (e.g., sendmail); Charlie wants to figure out Alice's encryption key

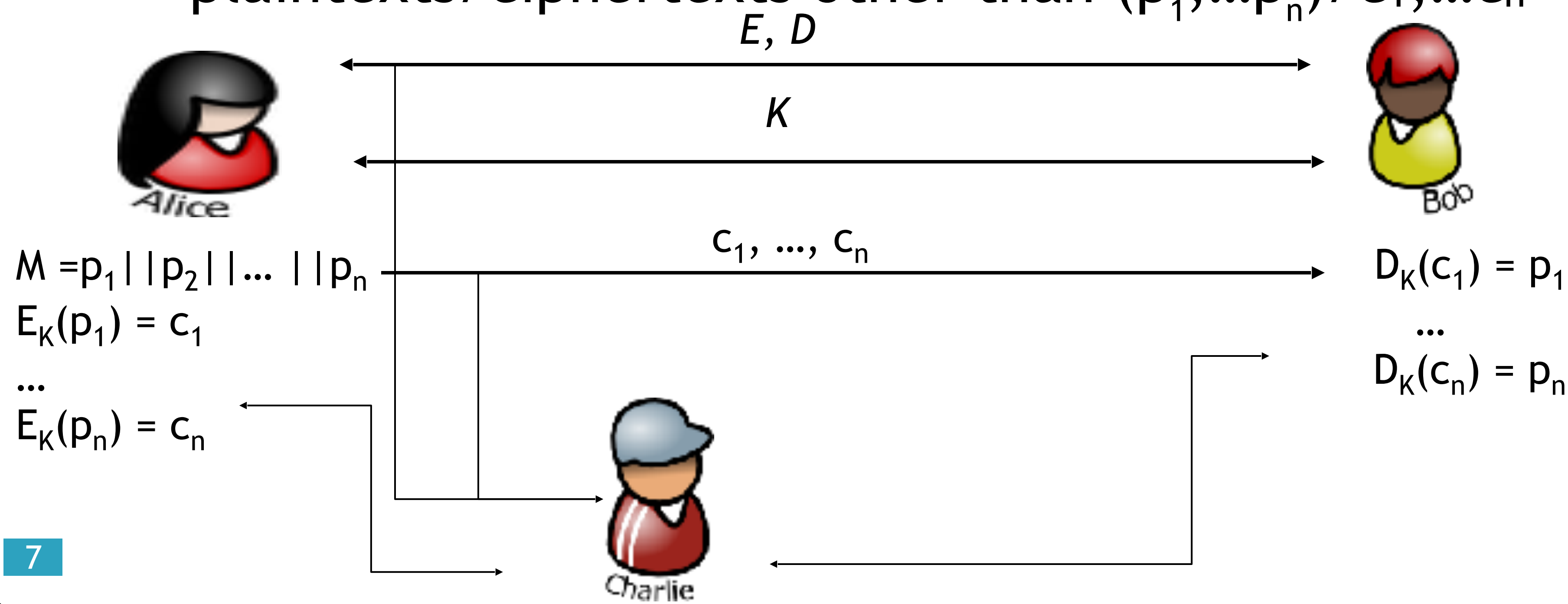
2. Charlie sends email messages to his friend Bob through Alice

3. Alice encrypts any message she receives and passes it on to the recipient

4. Charlie and Bob now have plaintext-ciphertext pairs of their choosing

Indistinguishability against Chosen Ciphertext Attack (IND-CCA)

- Charlie given: ciphertexts, E,D, plaintexts/ciphertexts chosen by him, corresponding encrypted/decrypted ciphertexts/plaintexts
- Charlie's goal: Guess encryptions/decrytions of plaintexts/ciphertexts other than $(p_1, \ldots p_n)/c_1, \ldots c_n$



Indistinguishability against Chosen Ciphertext Attack (IND-CCA)

- Give Charlie: $(c_1,\ldots,c_n)$, E,D, ((p_1,c_1), (p_2,c_2), …, (p_n,c_n))

- Charlie could pick p_i/c_i as a function of ((p_1,c_1), (p_2,c_2), …, (p_{i-1},c_{i-1})) for $i = 1 \ldots n$ (static vs. adaptive)
- Charlie is tasked with guessing encryptions/ decryptions other than those in $(p_1,..p_n)/(c_1,\ldots c_n)$, e.g., Bob later sends an encrypted file to Alice,…
- Provides solid level of security

Indistinguishability against Chosen Ciphertext Attack (IND-CCA) Game

- IND-CCA game
- Query phase:
  - Charlie is given (E,D)
  - Charlie queries Alice n times (polynomially bounded) for encryptions of $p_1,..p_n$, and Bob for decryptions of $c_i,..,c_k$ (sets could overlap)
  - Alice gives Charlie responses $c_1=E_K(p_1)$,… $c_n=E_K(p_n)$, and Bob gives responses $p_i=D_K(c_i)$,… $p_k=D_K(c_k)$

Indistinguishability against Chosen Ciphertext Attack (IND-CCA) Game

- Challenge-response phase:
  - Charlie chooses messages (m_0, m_1), gives to Alice
  - Alice encrypts and returns $C_b = E_K(m_b)$
  - Charlie can query Alice again on n plaintexts, even m_0, m_1 (in adaptive model)
  - Charlie can query Bob again on k ciphertexts, but not on C_b (in adaptive model): why not?

Indistinguishability against Chosen Ciphertext Attack (IND-CCA) Game

- Finally Charlie returns his guess b’ for b
- Charlie wins if Pr[(b==b’)] >> 1/2 + negl(ε)

- Some points:
  - In non-adaptive (static) model, Charlie cannot place calls to Bob or Alice in challenge-response phase
  - In adaptive model, Charlie can place calls in challenge-response phase, but still can’t query Bob on C_b

Chosen Ciphertext Attack Example

1. Charlie is a systems administrator in the CS dept. of a college.

2. Charlie has access to a smartchip (e.g., RSA Secure ID chip) using which he encrypts/ decrypts students' passwords.

3. The key for encryption/decryption is embedded in the smartchip, and isn't available to anyone, not even Charlie.

Chosen Ciphertext Attack Example

4. Think of the smartchip as a gadget. Charlie inputs a student's password, the chip encrypts it. Charlie punches in an encrypted password, the chip outputs the decrypted, plaintext password.

5. *Note that Charlie does not know the key.*

6. Charlie inputs student Alice's plaintext password, "alicepass", the chip produces encrypted password 0x23ac3ff4e5.

Chosen Ciphertext Attack Example

7. Charlie modifies 0x23ac3ff4e5 to 0x23ac**5de**4e5.

8. Charlie feeds in 0x23ac**5de**4e5 to the smartchip.

9. Smartchip decrypts 0x23ac**5de**4e5 to “aliziapaws”

10. Charlie is getting decryptions of his choice! Effectively he is mounting a chosen ciphertext attack on the smartchip

Chosen Plaintext Attack + Chosen Ciphertext Attack

- Chosen plaintext attack + Chosen ciphertext attack – **CPA + CCA**
  - Given: ciphertext, encryption/decryption algorithms, plaintext chosen by attacker and corresponding ciphertext, ciphertext chosen by attacker, corresponding decrypted plaintext

Adaptive Security

- Attacker gets to experiment with multiple plaintext/ciphertext pairs of their choices
- Attacker learns info from each query answer
- Use that information to tailor future queries
- Future queries are a function of past query answers

Adaptive vs. Non-adaptive Security

- What we've seen until now – adaptive security

- Non-adaptive security ("static" security)
  - Consider Chosen Plaintext Attack (CPA):
  - Give Charlie: $(c_1,\ldots,c_n)$, E, $(\ (p_1,c_1), (p_2,c_2), \ldots, (p_n,c_n)\ )$
  - Charlie picks all $(p_1,\ldots p_n)$ in advance (not on-the-fly)
  - Charlie <u>*cannot*</u> pick p_i as a function of $(\ (p_1,c_1), (p_2,c_2), \ldots, (p_{i-1},c_{i-1})\ )$ for $i = 1 \ldots n$

Adaptive vs. Non-adaptive Security

- Non-adaptive security ("static" security)
  - Consider Chosen Ciphertext Attack (CCA):
  - Give Charlie: $(c_1,\ldots,c_n)$, E, $(\ (p_1,c_1), (p_2,c_2), \ldots, (p_n,c_n)\ )$
  - Charlie picks all of the $(c_1,\ldots c_n)$ in advance (not on-the-fly)
  - Charlie <u>*cannot*</u> pick c_i as a function of $(\ (p_1,c_1), (p_2,c_2), \ldots, (p_{i-1},c_{i-1})\ )$ for $i = 1 \ldots n$

Another Approach – Brute Force Attack

- Recall "Breaking Crypto" slide

- Cryptanalysis -- we've seen this
  - Try out sample plaintexts, or ciphertexts or both
  - Try to guess key

- Exhaustive search or Brute-force attacks
  - Try all possible combinations of key
  - Try to guess key

Brute Force Attack

- Also called Exhaustive Search Attack
- Search entire keyspace, i.e., try all possible keys
- Give Charlie: E,(p_i,c_i), Charlie needs to find K
- How many possible keys does Charlie need to try out?

Brute Force Attack

- Keys are random bitstrings of length *k*: $K = \{0,1\}^k$
- For *k*-bit keys, possible keyspace is $K_1, \ldots, K_{2^k}$
  Why?
  - 1-bit key: 0,1 = 2^1
  - 2-bit key: 00, 01, 10, 11 = 2^2
  - 3-bit key: 000, 001, 011, 100, 101, 110, 111 = 2^3
  - …….
  - k-bit key: k permutations = 2^k

Brute Force Attack

Example

- Charlie gets input: $E,(p_i,c_i)$
- Charlie needs to guess encryption key K
- Charlie runs:

```
for (j=1; j<=2^k; j++ )
   if (E_Kj(p_i)=c_i)
   return K_j
```

- Key *K must be* one among 1, …, 2^k

Brute Force Attack

- Optimization: Do computations in parallel
- Each of the $E_{Kj}(p_i)=c_i$ comparisons can be done in parallel
- Remarkably fast: Can break DES in around 3.5 hours

Summary

- Historical crypto (Caesar, Vigenère, etc.)

- Formal definition

- One way of breaking crypto
  - Brute force attack or exhaustive search attack
  - Search entire keyspace
  - On an average, for X keys, need to try $X/2$ keys

Summary

- Another way of breaking crypto
  - Cryptanalysis
  - KCA, KPA, CPA, adaptive CPA, CCA, adaptive CCA, in that order
  - KCA least dangerous
  - Adaptive CCA most dangerous
- CPA a.k.a. **semantic security** is the minimum level of security an algorithm must provide (any usable algorithm *must* be resistant to CPA attacks)

Summary

- Adaptive CCA strongest notion of security
  - Very few algorithms resistant to adaptive CCA attacks
  - Cramer-Shoup cryptosystem (Ron Cramer, Victor Shoup. Crypto’98)
  - Hofheinz-Kiltz-Shoup cryptosystem (Dennis Hofheinz, Eike Kiltz, Victor Shoup. Journal of Cryptology‘14)