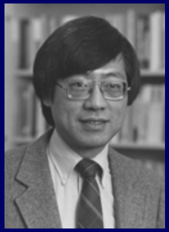
Formal Models of Cryptography: Symmetric Encryption

Overview

- Dolev-Yao model
- When is the Dolev-Yao model inadequate?
  - A cautionary tale: recursive authentication protocol [Ryan, Schneider '97]
- Formal definitions of security for cryptographic schemes
  - Symmetric and asymmetric (public-key) encryption
  - CPA and CCA indistinguishability
- Formal definitions of security for key exchange

“Dolev-Yao” Model

- Inspired by a 1983 paper
  - *D. Dolev and A. Yao. “On the security of public key protocols”. IEEE Transactions on Information Theory, 29(2):198-208.*
- Adversary is a nondeterministic process
  - Can read any message, decompose it into parts and re-assemble
  - Cannot gain partial knowledge, perform statistical tests, ...
- “Black-box” cryptography
  - Adversary can decrypt if and only if he knows the correct key
  - Assumes that cryptographic functions have no special properties
- Most mechanized formal methods for security analysis use some version of this model

Typical Dolev-Yao Term Algebra

Attacker's **term algebra** is a set of derivation rules

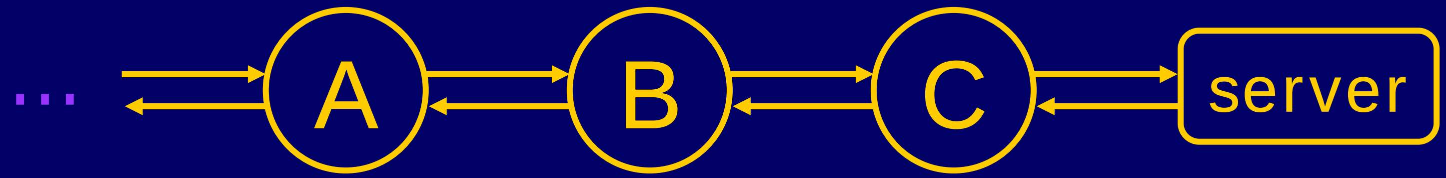
$$\frac{v \in T}{T \triangleright u} \text{ if } u = v\sigma \text{ for some } \sigma \qquad \frac{T \triangleright u \quad T \triangleright v}{T \triangleright [u,v]} \qquad \frac{T \triangleright u \qquad T \triangleright v}{T \triangleright \mathrm{crypt}_u[v]}$$

$$\frac{T \triangleright [u,v]}{T \triangleright u} \qquad \frac{T \triangleright [u,v]}{T \triangleright v} \qquad \frac{T \triangleright \mathrm{crypt}_u[v] \quad T \triangleright u}{T \triangleright v}$$

In the real world, there is no guarantee that attacker is restricted to these operations! He may perform probabilistic operations, learn partial information, etc.

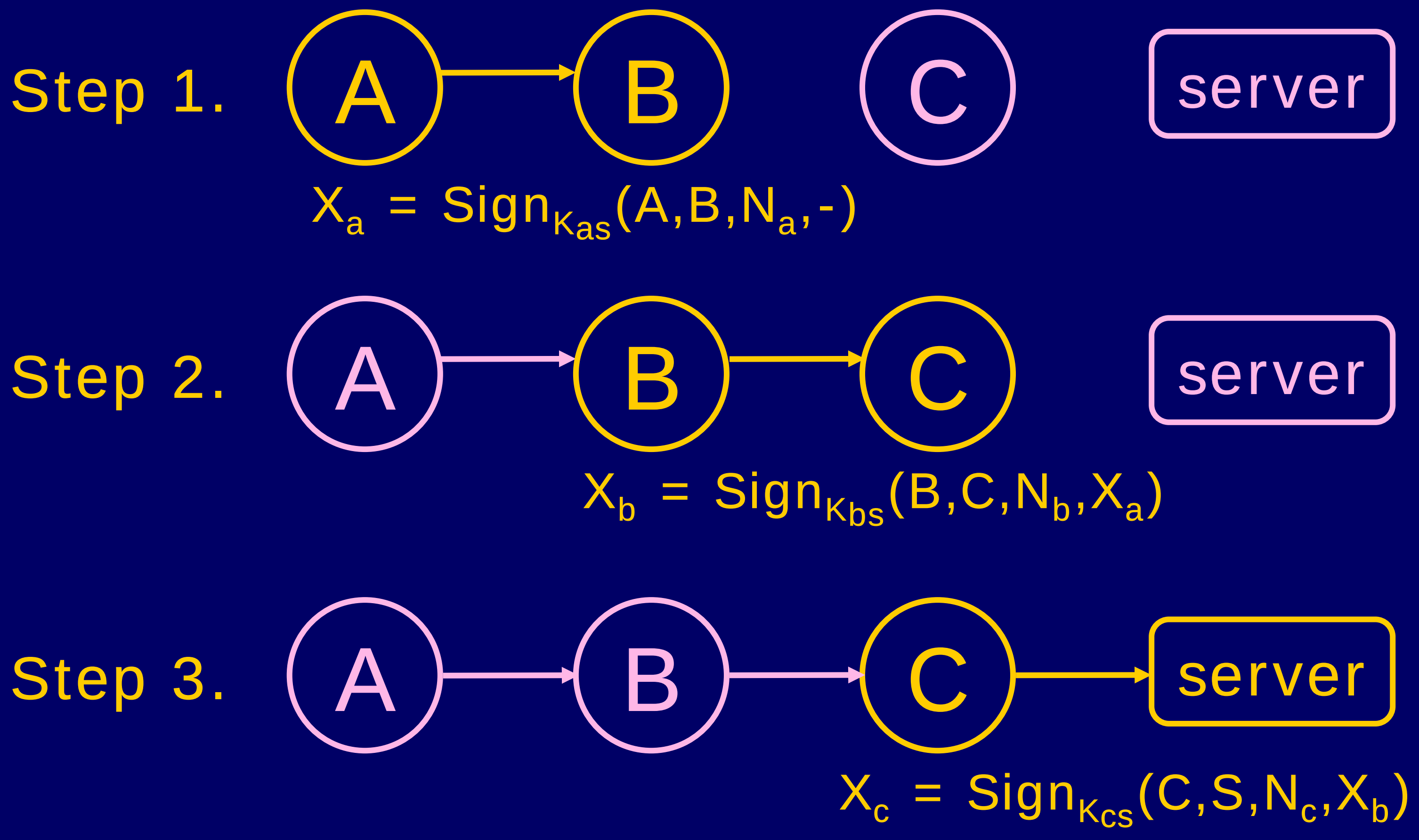
Recursive Authentication

[Bull '97]

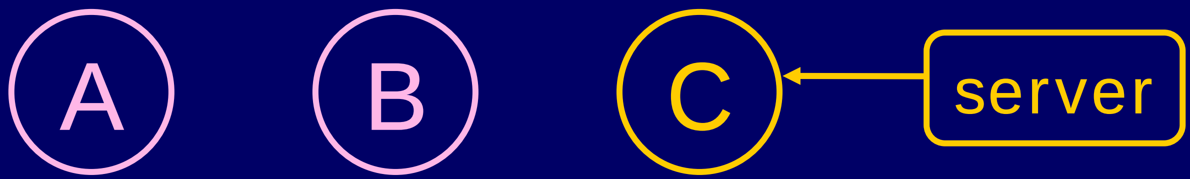


- Each agent initially shares a pairwise key with the server
  - K_{as}, K_{bs}, K_{cs}
- Goal: establish pairwise session keys K_{ab} and K_{bc} with minimal communication

Request Phase



Key Distribution (1)



A, B, $Encrypt_{hash(K_{as},N_a)}(K_{ab})$, $Encrypt_{K_{ab}}(A,B,N_a)$,

B, A, $Encrypt_{hash(K_{bs},N_b)}(K_{ab})$, $Encrypt_{K_{ab}}(B,A,N_b)$,

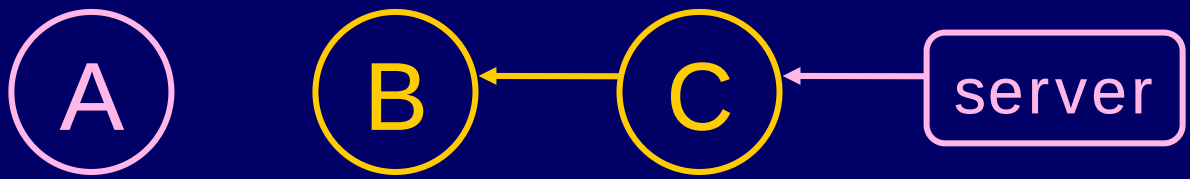
B, C, $Encrypt_{hash(K_{bs},N_b)}(K_{bc})$, $Encrypt_{K_{bc}}(B,C,N_b)$,

C, B, $Encrypt_{hash(K_{cs},N_c)}(K_{bc})$, $Encrypt_{K_{bc}}(C,B,N_c)$

C decrypts and learns K_{bc} because only C knows both K_{cs} and N_c

C uses newly learned K_{bc} to decrypt and verify

Key Distribution (2)



$A, B, Encrypt_{hash(K_{as},N_a)}(K_{ab}), Encrypt_{K_{ab}}(A,B,N_a),$

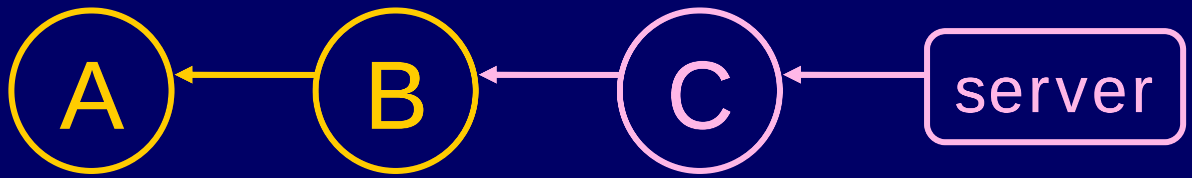
$B, A, Encrypt_{hash(K_{bs},N_b)}(K_{ab}), Encrypt_{K_{ab}}(B,A,N_b),$

$B, C, Encrypt_{hash(K_{bs},N_b)}(K_{bc}), Encrypt_{K_{bc}}(B,C,N_b)$

B decrypts and learns K_{ab} and K_{bc} because only B knows both K_{bs} and N_b

B uses newly learned K_{ab} and K_{bc} to decrypt and verify

Key Distribution (3)



A, B, $\text{Encrypt}_{\text{hash}(K_{as},N_a)}(K_{ab})$, $\text{Encrypt}_{K_{ab}}(A,B,N_a)$

A decrypts and learns K_{ab} because only A knows both K_{as} and N_a

A uses newly learned K_{ab} to decrypt and verify

Abstract encryption: design of the protocol can be verified without modeling details of the underlying crypto system…

… proved correct by Paulson in his CSFW '97 paper using higher-order logic (in particular, malicious C cannot learn K_{ab})

Two Views of Encryption

abstraction

$\text{Encrypt}_k(M)$

Message M encrypted with key k in some symmetric cipher

$k \oplus M$

implementation

Specific implementation from Bull's recursive authentication paper (perfectly reasonable: block ciphers are implemented like this)

Key Distribution "Refined"

A, B, $hash(K_{as},N_a) \oplus K_{ab}$, ...

B, A, $hash(K_{bs},N_b) \oplus K_{ab}$, ...

B, C, $hash(K_{bs},N_b) \oplus K_{bc}$, ...

C, B, $hash(K_{cs},N_c) \oplus K_{bc}$

C computes
$K_{bc} \oplus$
$(hash(K_{bs},N_b) \oplus K_{ab}) \oplus$
$(hash(K_{bs},N_b) \oplus K_{bc}) =$
$K_{bc} \oplus K_{ab} \oplus K_{bc} =$
K_{ab}

C knows K_{cs} and N_c, computes $hash(K_{cs},N_c)$ and learns K_{bc}

Oops!

When abstraction is refined in a "provably secure" protocol, C learns secret key K_{ab}

Abstraction Gap

- Formal models pretend that the output of a cryptographic primitive is an abstract data type
  - Can only access values through the type interface
    - E.g., apply “decrypt” to a ciphertext and a key
  - Cannot access values in any other way
    - This does not follow directly from cryptographic definitions of security
  - Ignore possibility of partial information leakage
    - In the Dolev-Yao model, there is no way to say “adversary learns 7^{th} bit with probability 0.55”
- Goal: sound “abstraction” of cryptography that can be used by higher protocol levels

Typical Pattern for a Definition

- Define cryptographic functionalities as oracles
- Define a game between adversary and the oracles
  - The goal of the adversary is to “break” security
  - For example, adversary against an encryption scheme succeeds if he learns even a single bit of plaintext
- Computational security: probabilistic poly-time adversary succeeds only with negligible probability
  - < 1/poly(n) for any polynomial of security parameter n
- Information-theoretic security: computationally unbounded adversary cannot succeed

Cryptographic “Oracles”

- Formal representation of cryptographic operations available to the adversary
  - E.g., adversary may use the protocol to obtain ciphertexts corresponding to plaintexts of his choice; we model this by giving adversary access to an encryption oracle
  - Similar for decryption oracles, signing oracles, etc.
- The rules of the game constrain how adversary may interact with the oracles
  - Different types of attacks (CPA, CCA, etc.) depending on what the adversary is permitted to do

Symmetric Encryption

- A symmetric encryption scheme SE consists in three algorithms K, E, D
- Key generation algorithm K returns a string from some set Keys(SE)
  - Key generation algorithm is randomized
- Encryption algorithm E takes $k \in \text{Keys(SE)}$ and $m \in \{0,1\}^*$ and returns ciphertext $c \in \{0,1\}^* \cup \{\perp\}$
  - Encryption algorithm may be randomized or stateful
- Decryption algorithm D takes $k \in \text{Keys(SE)}$ and $c \in \{0,1\}^*$ and returns some $m \in \{0,1\}^* \cup \{\perp\}$
  - Decryption algorithm is deterministic

What Does "Security" Mean?

- Hard to recover the key?
  - What if the adversary can learn plaintext without learning the key?
- Hard to recover plaintext from ciphertext?
  - What if the adversary learns some bits or some function of bits?
- Fixed mapping from plaintexts to ciphertexts?
  - What if the adversary see two identical ciphertexts and infers that the corresponding plaintexts are identical?
  - Implication: encryption must be randomized or stateful

Left-Right Encryption Oracles

- Idea: adversary should not be able to learn even a single bit
- Define left-right encryption oracle $E_k(LR(m_0,m_1,b))$ where $b \in \{0,1\}$ as

  if $|m_0| \neq |m_1|$ then return $\perp$

  else return $E_k(M_b)$

Given two plaintexts, returns encryption of one of them

- Adversary is given access to $E_k(LR(-,-,b))$
  - Bit b is fixed, but adversary doesn't know its value
  - Adversary can use any plaintexts m_0, m_1 as inputs; one of them will be returned as ciphertext. To learn bit b, adversary must determine which one was returned.

Chosen-Plaintext Indistinguishability

- Consider two experiments
  - A is the adversary with oracle access

$\underline{\mathrm{Exp}_{SE}^{\,0}(A)}$

$k \leftarrow K$ (keygen)

$d \leftarrow A(E_k(LR(-,-,0)))$

return d

$\underline{\mathrm{Exp}_{SE}^{\,1}(A)}$

$k \leftarrow K$ (keygen)

$d \leftarrow A(E_k(LR(-,-,1)))$

return d

- The IND-CPA advantage of A is

$$\mathrm{Adv}(A) = |\Pr(\mathrm{Exp}_{SE}^{\,0}(A)=1) - \Pr(\mathrm{Exp}_{SE}^{\,1}(A)=1)|$$

"Measures" A's ability to make his output depend on oracle's bit

- Encryption scheme is chosen-plaintext secure if advantage is negligible for any prob polytime A

CPA Game

1. Security parameter is given to all algorithms, including the adversary
2. The key is generated and given to all oracles
   - Adversary does not learn the key
3. Adversary makes as many queries as he wants to encryption oracles, obtaining encryption of any message of his choice
   - Number of queries must be polynomial in security parameter
4. When adversary is ready, he outputs m_0 and m_1 of his choice. The "test oracle" picks a random bit b and returns encryption of m_b to the adversary.
5. Adversary may continue asking for encryptions of any plaintexts, including m_0 and m_1
6. Adversary outputs b', which is his judgement about what bit b is
7. The scheme is secure if the probability that b'=b is at most negligibly better than a random coin toss, i.e. 1/2

Simple Example

- <u>Any</u> deterministic, stateless symmetric encryption scheme is insecure
  - Adversary can easily distinguish encryptions of different plaintexts from encryptions of identical plaintexts

<u>Adversary $A(E_k(LR(-,-,b))$</u>

Let X,Y be distinct strings in plaintext space

$C_1 \leftarrow E_k(LR(X,Y,b))$

$C_2 \leftarrow E_k(LR(Y,Y,b))$

If $C_1=C_2$ then return 1 else return 0

- The IND-CPA advantage of A is 1

$$\Pr(\mathrm{Exp}_{SE}^{0}(A)=1)=0 \quad \Pr(\mathrm{Exp}_{SE}^{1}(A)=1)=1$$

CBC Mode: Encryption

- CBC (cipherblock chaining) is a common mode for using block ciphers such as DES and Rijndael
- Let $E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ be the n-bit block cipher

Pseudo-random permutation family with fixed block length

Algorithm CBC-encrypt$_k$(M)

if $|M| \neq 0 \bmod n$ or $|M|=0$ then return $\perp$

break M into n-bit blocks M[1]...M[m]

IV $\leftarrow$ random $\{0,1\}^n$

C[0] $\leftarrow$ IV

for i=1 to m do C[i] $\leftarrow E_k(C[i-1] \oplus M[i])$

C $\leftarrow$ C[1] ... C[m]

return (IV,C)

Randomly generate initialization vector

XOR each plaintext with previous cipherblock and encrypt using block cipher to produce next cipherblock

CBC Mode: Decryption

Algorithm CBC-decrypt$_k$(IV,C)

if $|C| \neq 0 \bmod n$ or $|C|=0$ then return $\perp$

break C into n-bit blocks C[1]...C[m]

$C[0] \leftarrow IV$

for i=1 to m do $M[i] \leftarrow E_k^{-1}(C[i]) \oplus C[i-1]$

$M \leftarrow M[1] \ldots M[m]$

return M

- CBC with random IV is IND-CPA secure
  - [Proof omitted]

CBCC: Use Counters for IV

- Replace random initialization vectors with counters

Algorithm CBCC-encrypt$_k$(M)

static ctr $\leftarrow$ 0

Values of ctr are persistent across multiple invocations of CBCC-encrypt

if $|M| \neq 0 \bmod n$ or $|M|=0$ then return $\perp$

break M into n-bit blocks M[1]...M[m]

if $ctr \geq 2^n$ then return $\perp$

IV $\leftarrow [ctr]_n$

Use current counter value as initialization vector

$C[0] \leftarrow IV$

for i=1 to m do $C[i] \leftarrow E_k(C[i-1] \oplus M[i])$

$C \leftarrow C[1] \ldots C[m]$

ctr $\leftarrow$ ctr+1

Increase counter on each invocation of CBCC-encrypt

return (IV,C)

Chosen-Plaintext Attack on CBCC

- Problem: adversary can predict counter value

Adversary $A(E_k(LR(-,-,b))$

$M_0 \leftarrow 0^n$, $M_1 \leftarrow 0^n$,

$M'_0 \leftarrow 0^n$, $M'_1 \leftarrow 0^{n-1}1$

$(IV,C) \leftarrow E_k(LR(M_0,M_1,b))$

$(IV',C') \leftarrow E_k(LR(M'_0,M'_1,b))$

If $C=C'$ then return 1 else return 0

$IV=0$, $IV'=1$

If $b=0$ then $C=E_k(IV \oplus M_0)=E_K(0 \oplus 0)=E_K(0)$

$C'= E_k(IV' \oplus M'_0)=E_K(1 \oplus 0)=E_K(1) \neq C$

If $b=1$, then $C=E_k(IV \oplus M_0)=E_K(0 \oplus 0)=E_K(0)$

$C'= E_k(IV' \oplus M'_0)=E_K(1 \oplus 1)=E_K(0) = C$

- The IND-CPA advantage of A is 1

$\Pr(Exp_{SE}{}^0(A)=1)=0 \quad \Pr(Exp_{SE}{}^1(A)=1)=1$

From CPA to CCA

- A stronger form of security than chosen-plaintext indistinguishability is chosen-ciphertext indistinguishability
- Suppose that in addition to encryption oracles, adversary also has access to decryption oracles
  - A decryption oracle is simply an algorithm that decrypts any ciphertext (or anything that looks like ciphertext) on adversary's request
  - For example, in many protocols participants are expected to decrypt random challenges. This may give the adversary an opportunity to obtain a decryption of a ciphertext of his choice.

“Lunchtime” CCA Game (CCA-1)

1. Security parameter is given to all algorithms, including the adversary
2. The key is generated and given to all oracles
   - Adversary does not learn the key
3. Adversary makes as many queries as he wants to encryption oracles, obtaining encryption of any message of his choice. Adversary also obtains decryptions of as many ciphertexts as he wants by querying decryption oracles.
   - Number of queries must be polynomial in security parameter
4. When adversary is ready, he outputs m_0 and m_1 of his choice. The “test oracle” picks a random bit b and returns encryption of m_b to the adversary.
5. Adversary may continue asking for encryptions of any plaintexts, including m_0 and m_1
6. Adversary outputs b’, which is his judgement about what bit b is
7. The scheme is secure if the probability that b’=b is at most negligibly better than a random coin toss, i.e. 1/2

CCA-2 Game

1. Security parameter is given to all algorithms, including the adversary
2. The key is generated and given to all oracles
   - Adversary does not learn the key
3. Adversary makes as many queries as he wants to encryption oracles, obtaining encryption of any message of his choice. Adversary also obtains decryptions of as many ciphertexts as he wants by querying decryption oracles.
   - Number of queries must be polynomial in security parameter
4. When adversary is ready, he outputs m_0 and m_1 of his choice. The "test oracle" picks a random bit b and returns encryption of m_b to the adversary.
5. Adversary may continue asking for encryptions of any plaintexts, including m_0 and m_1. Adversary may also continue asking for decryptions of any ciphertext <u>except</u> the one ciphertext returned by the test oracle.
6. Adversary outputs b', which is his judgement about what bit b is
7. The scheme is secure if the probability that b'=b is at most negligibly better than a random coin toss, i.e. 1/2