Defining Encryption

Lecture 2

Defining Encryption

Lecture 2

Secrecy when Computationally Bounded

Roadmap

Roadmap

- First, Symmetric Key Encryption

Roadmap

- First, Symmetric Key Encryption
- Defining the problem
  - We'll do it elaborately, so that it will be easy to see different levels of security

Roadmap

- First, Symmetric Key Encryption
- Defining the problem
  - We'll do it elaborately, so that it will be easy to see different levels of security
- Solving the problem
  - In theory and in practice

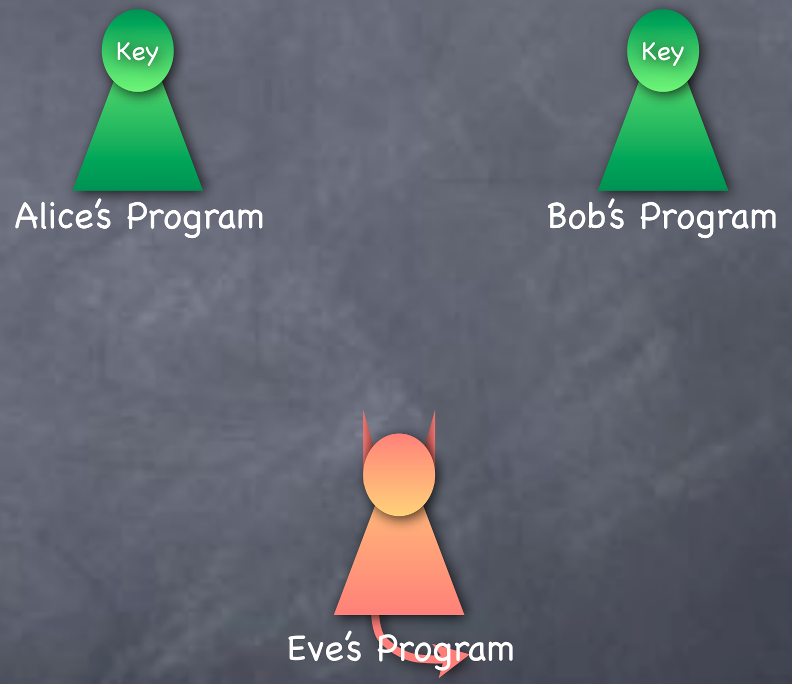
Roadmap

- First, Symmetric Key Encryption
- Defining the problem
  - We'll do it elaborately, so that it will be easy to see different levels of security
- Solving the problem
  - In theory and in practice
- Today: defining symmetric-key encryption

Building the Model

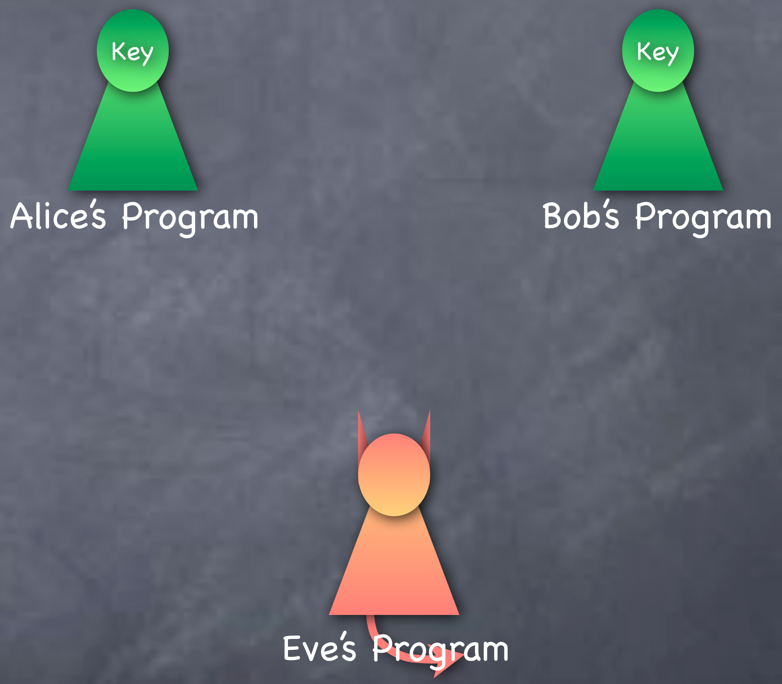
Building the Model

- Alice, Bob and Eve. Alice and Bob share a key (a bit string)



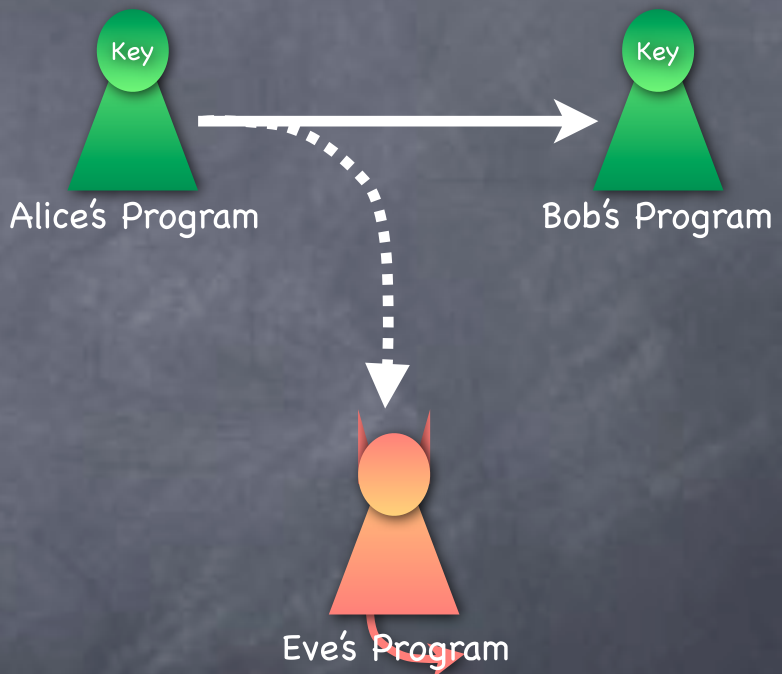
Building the Model

- Alice, Bob and Eve. Alice and Bob share a key (a bit string)
- Alice wants Bob to learn a message, "without Eve learning it"

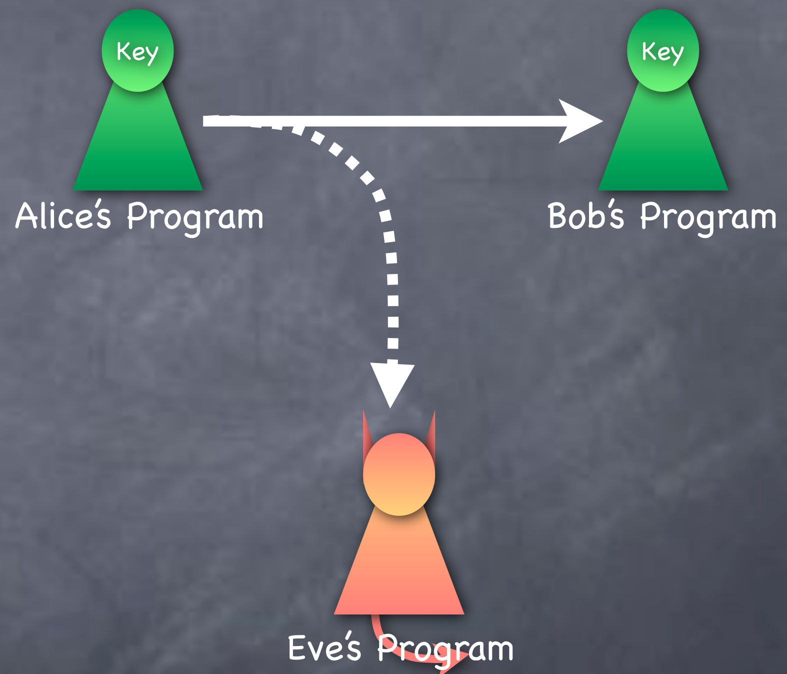


Building the Model

- Alice, Bob and Eve. Alice and Bob share a key (a bit string)
- Alice wants Bob to learn a message, "without Eve learning it"
- Alice can send out a bit string on the channel. Bob and Eve both get it

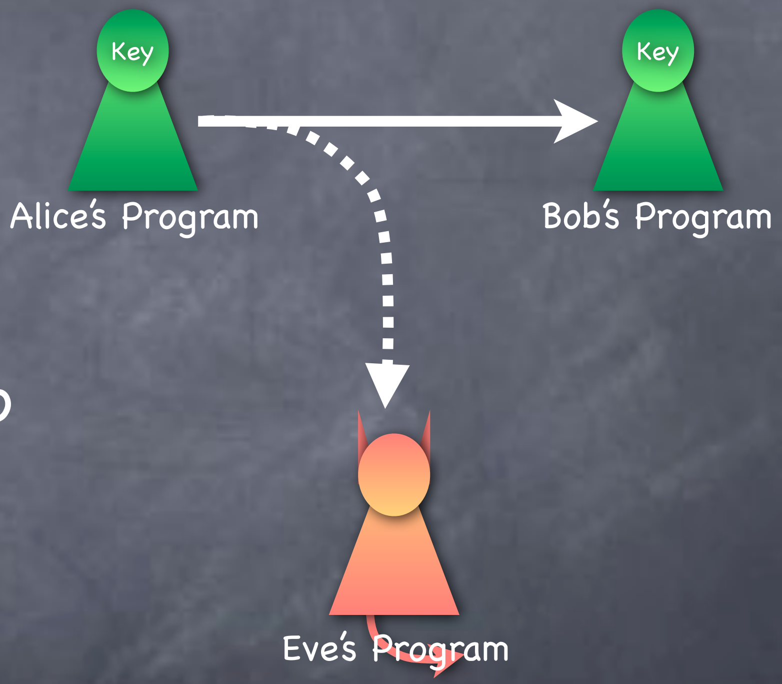


Encryption: Syntax



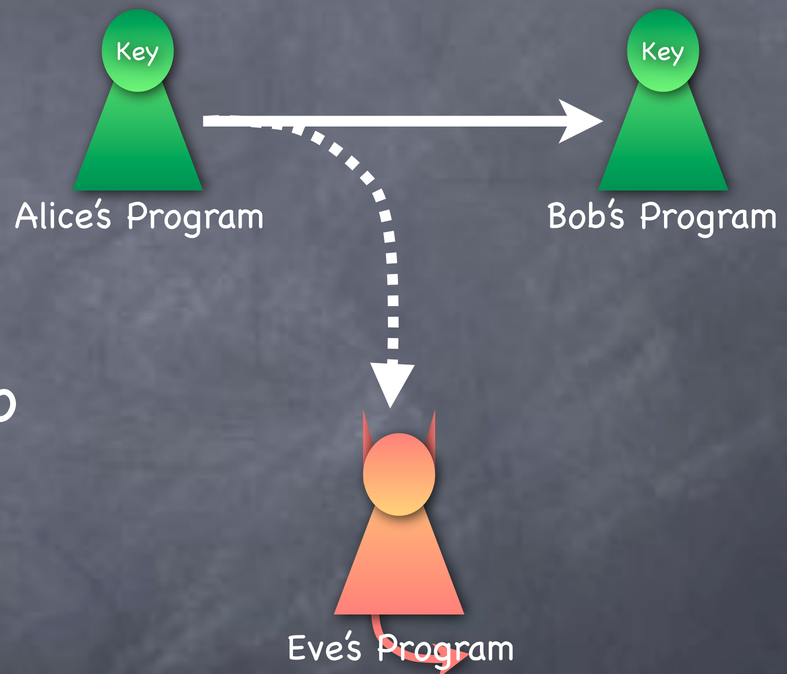
Encryption: Syntax

- Three algorithms
  - Key Generation: What Alice and Bob do a priori, for creating the shared secret key
  - Encryption: What Alice does with the message and the key to obtain a "ciphertext"
  - Decryption: What Bob does with the ciphertext and the key to get the message out of it

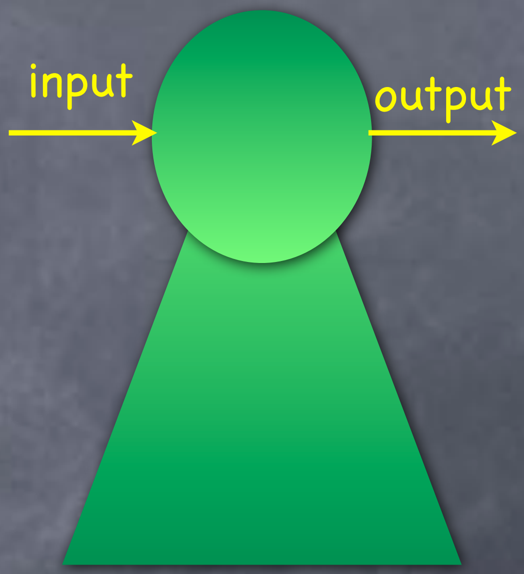


Encryption: Syntax

- Three algorithms
  - Key Generation: What Alice and Bob do a priori, for creating the shared secret key
  - Encryption: What Alice does with the message and the key to obtain a "ciphertext"
  - Decryption: What Bob does with the ciphertext and the key to get the message out of it
- All of these are (probabilistic) computations

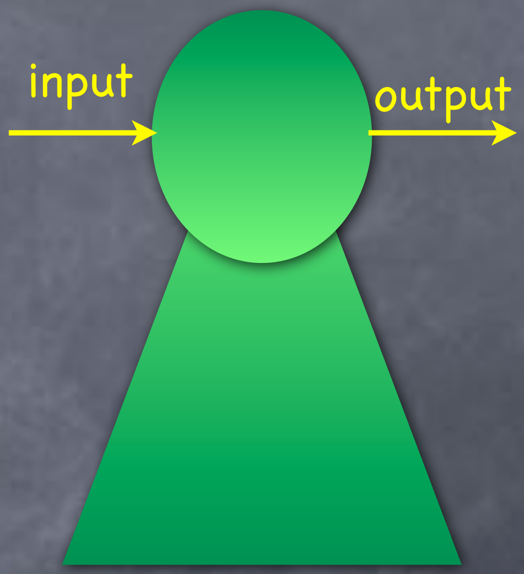


Modeling Computation



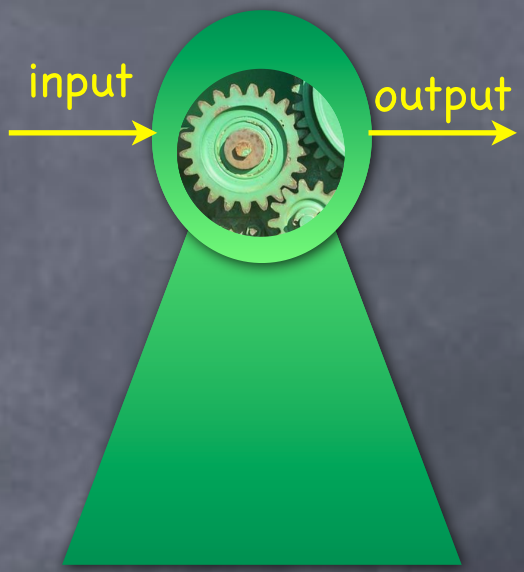
Modeling Computation

- In our model (standard model) parties are programs (computations, say Turing Machines)



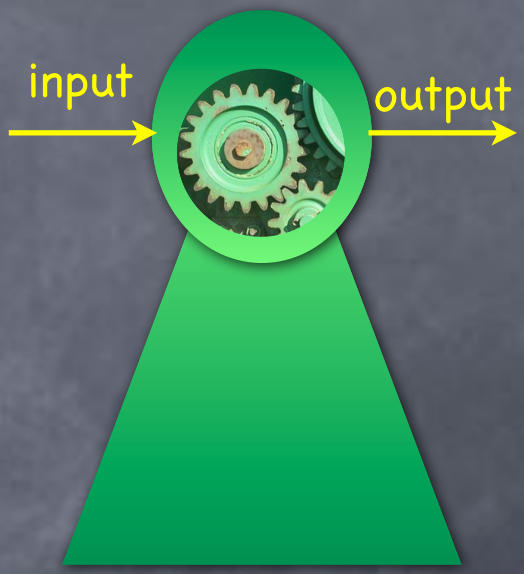
Modeling Computation

- In our model (standard model) parties are programs (computations, say Turing Machines)



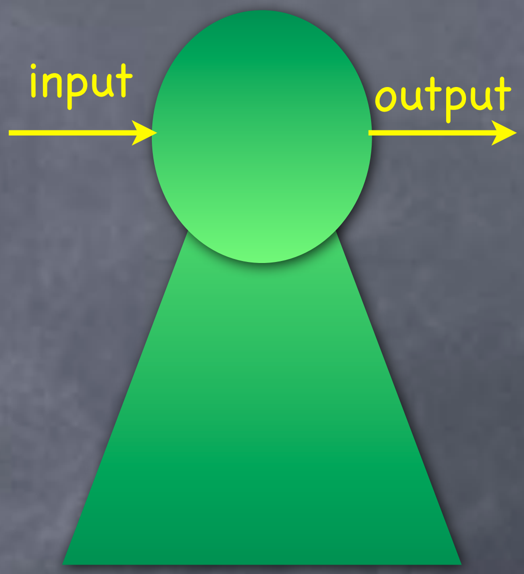
Modeling Computation

- In our model (standard model) parties are programs (computations, say Turing Machines)
- Effect of computation limited to be in a blackbox manner (only through input/output functionality)



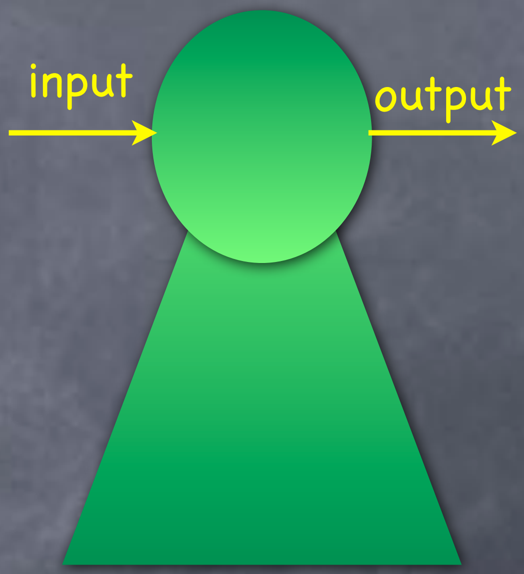
Modeling Computation

- In our model (standard model) parties are programs (computations, say Turing Machines)
- Effect of computation limited to be in a blackbox manner (only through input/output functionality)



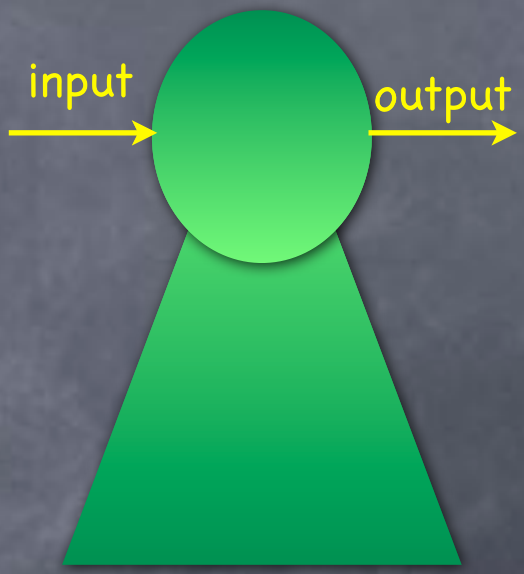
Modeling Computation

- In our model (standard model) parties are programs (computations, say Turing Machines)
- Effect of computation limited to be in a blackbox manner (only through input/output functionality)
  - No side-information (timing, electric signals, ...) unless explicitly modeled



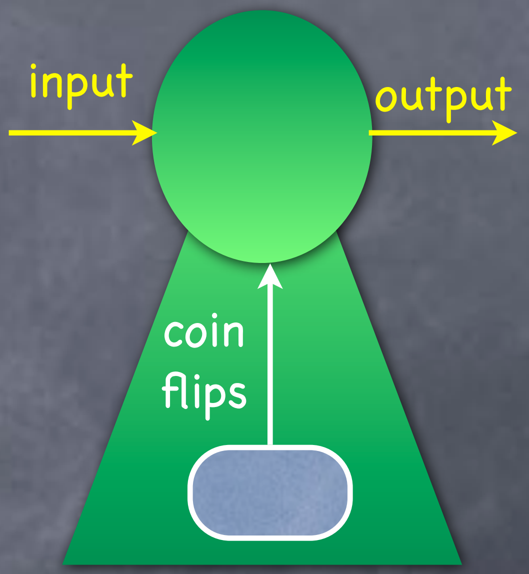
Modeling Computation

- In our model (standard model) parties are programs (computations, say Turing Machines)
- Effect of computation limited to be in a blackbox manner (only through input/output functionality)
  - No side-information (timing, electric signals, ...) unless explicitly modeled
  - Can be probabilistic



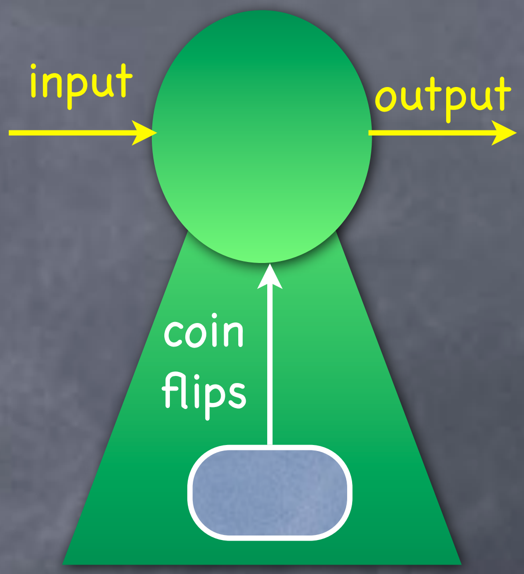
Modeling Computation

- In our model (standard model) parties are programs (computations, say Turing Machines)
- Effect of computation limited to be in a blackbox manner (only through input/output functionality)
  - No side-information (timing, electric signals, ...) unless explicitly modeled
  - Can be probabilistic



Modeling Computation

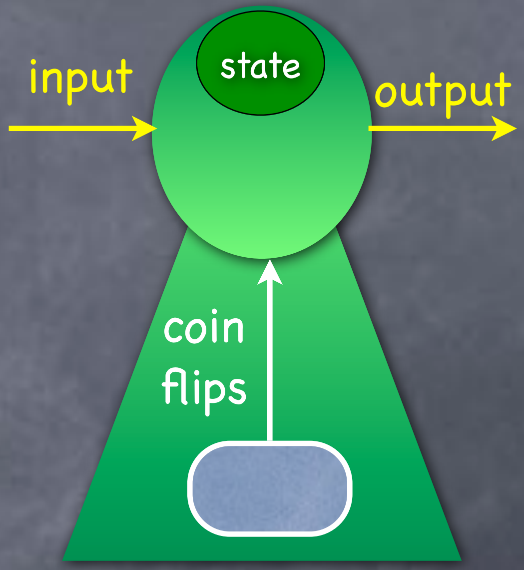
- In our model (standard model) parties are programs (computations, say Turing Machines)
- Effect of computation limited to be in a blackbox manner (only through input/output functionality)
  - No side-information (timing, electric signals, ...) unless explicitly modeled
  - Can be probabilistic



Ideal coin flips: If n coins flipped, each outcome has probability 2^{-n}

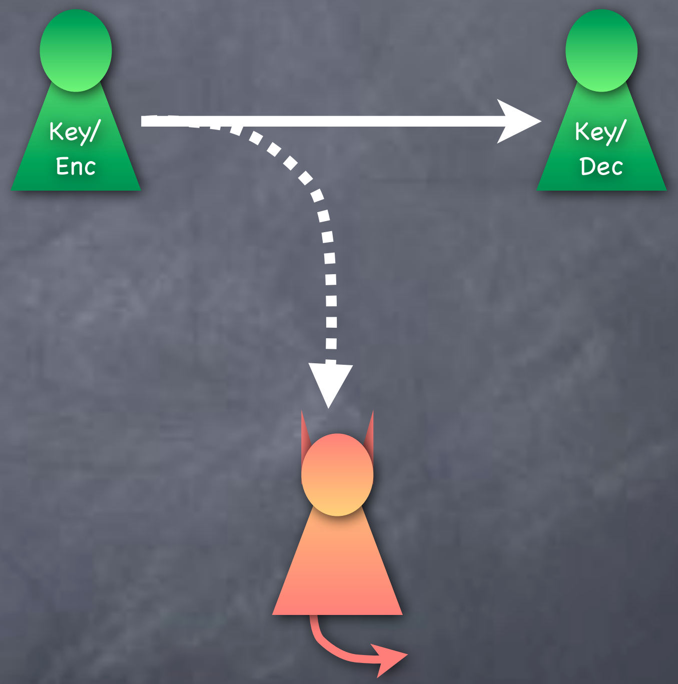
Modeling Computation

- In our model (standard model) parties are programs (computations, say Turing Machines)
- Effect of computation limited to be in a blackbox manner (only through input/output functionality)
  - No side-information (timing, electric signals, ...) unless explicitly modeled
  - Can be probabilistic
  - Sometimes stateful



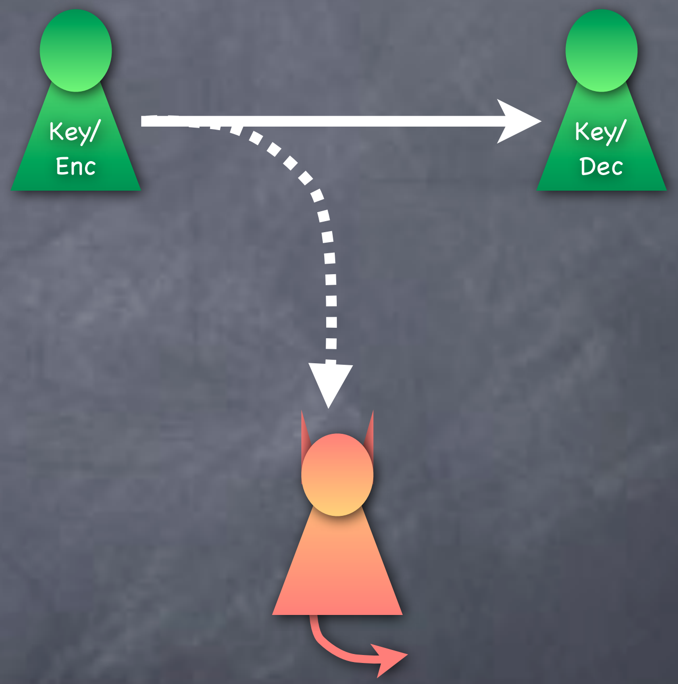
Ideal coin flips: If n coins flipped, each outcome has probability 2^{-n}

The Environment



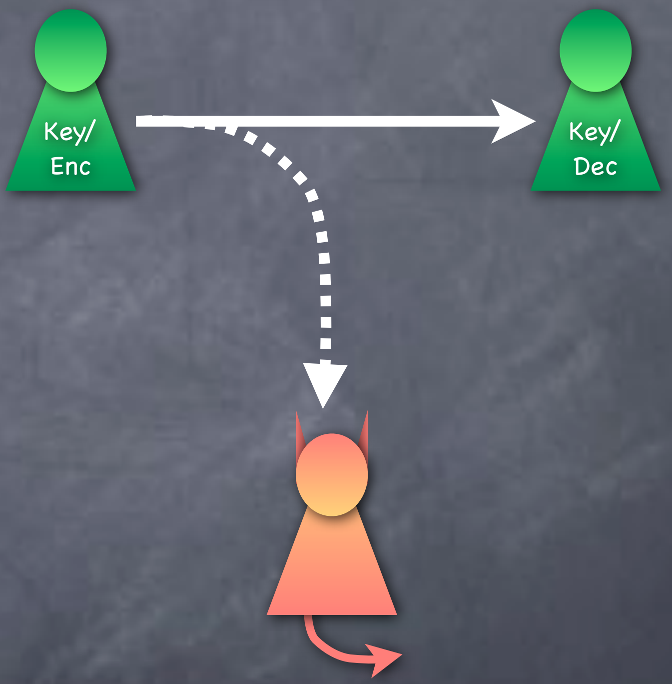
The Environment

- Where does the message come from?



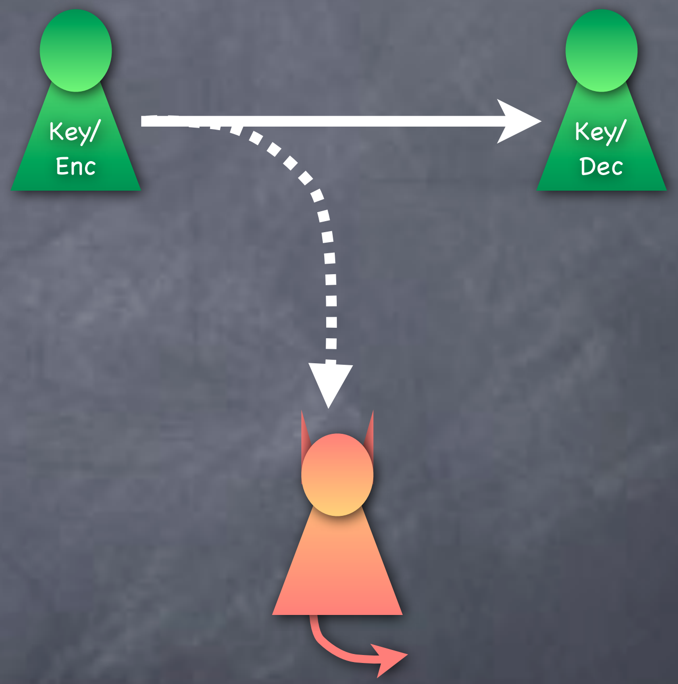
The Environment

- Where does the message come from?
  - Eve might already have partial information about the message, or might receive such information later



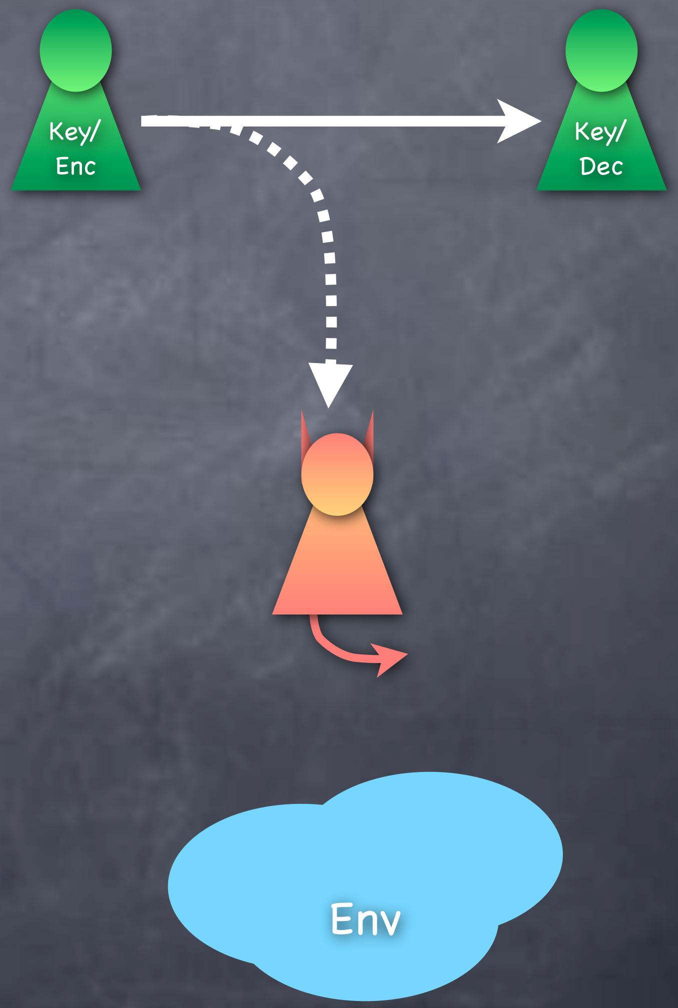
The Environment

- Where does the message come from?
  - Eve might already have partial information about the message, or might receive such information later
  - In fact, Eve might influence the choice of the message



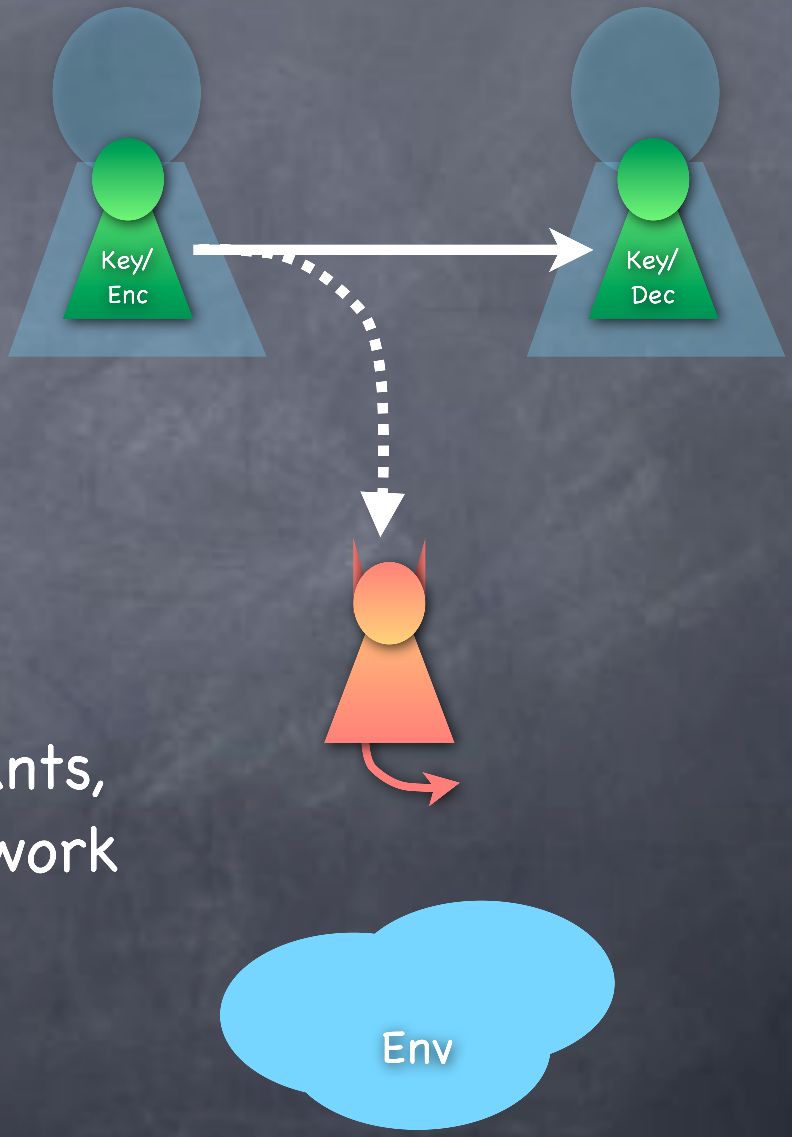
The Environment

- Where does the message come from?
  - Eve might already have partial information about the message, or might receive such information later
  - In fact, Eve might influence the choice of the message
- The environment



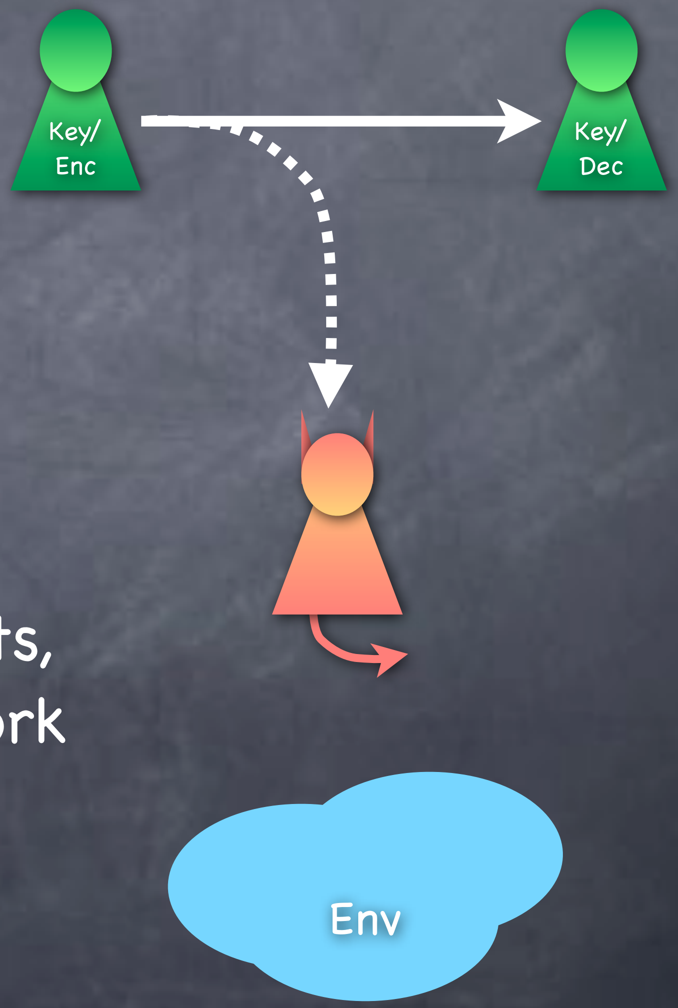
The Environment

- Where does the message come from?
  - Eve might already have partial information about the message, or might receive such information later
  - In fact, Eve might influence the choice of the message
- The environment
  - Includes the operating systems and other programs run by the participants, as well as other parties, if in a network



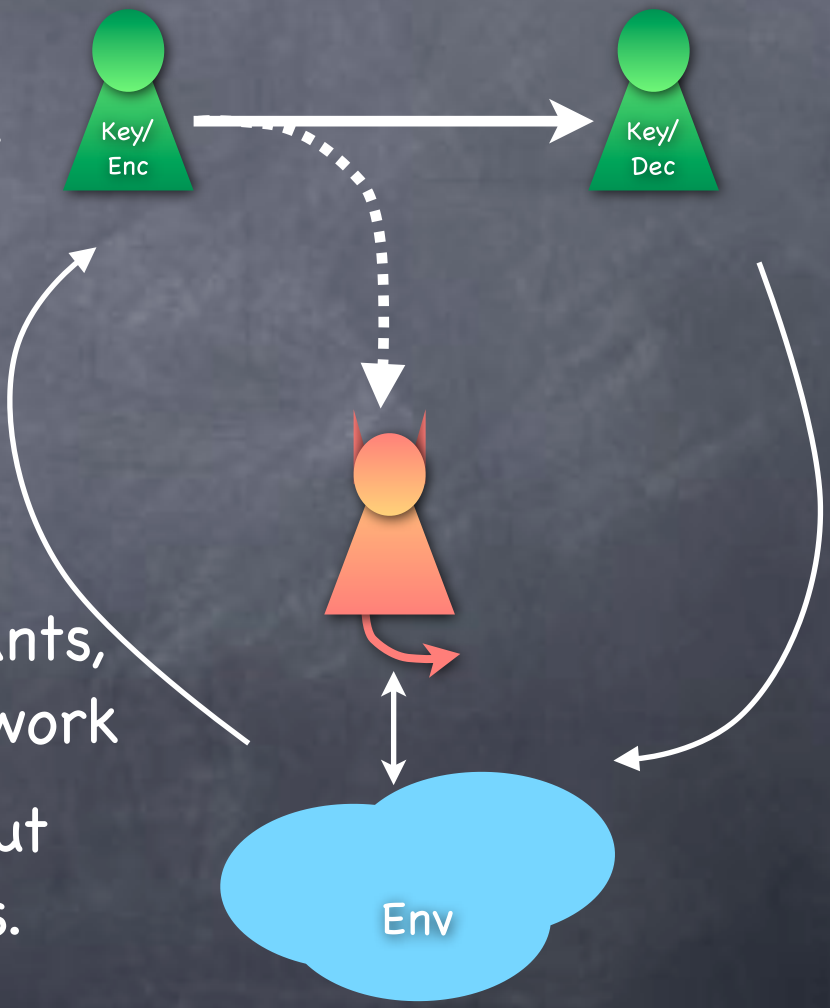
The Environment

- Where does the message come from?
  - Eve might already have partial information about the message, or might receive such information later
  - In fact, Eve might influence the choice of the message
- The environment
  - Includes the operating systems and other programs run by the participants, as well as other parties, if in a network

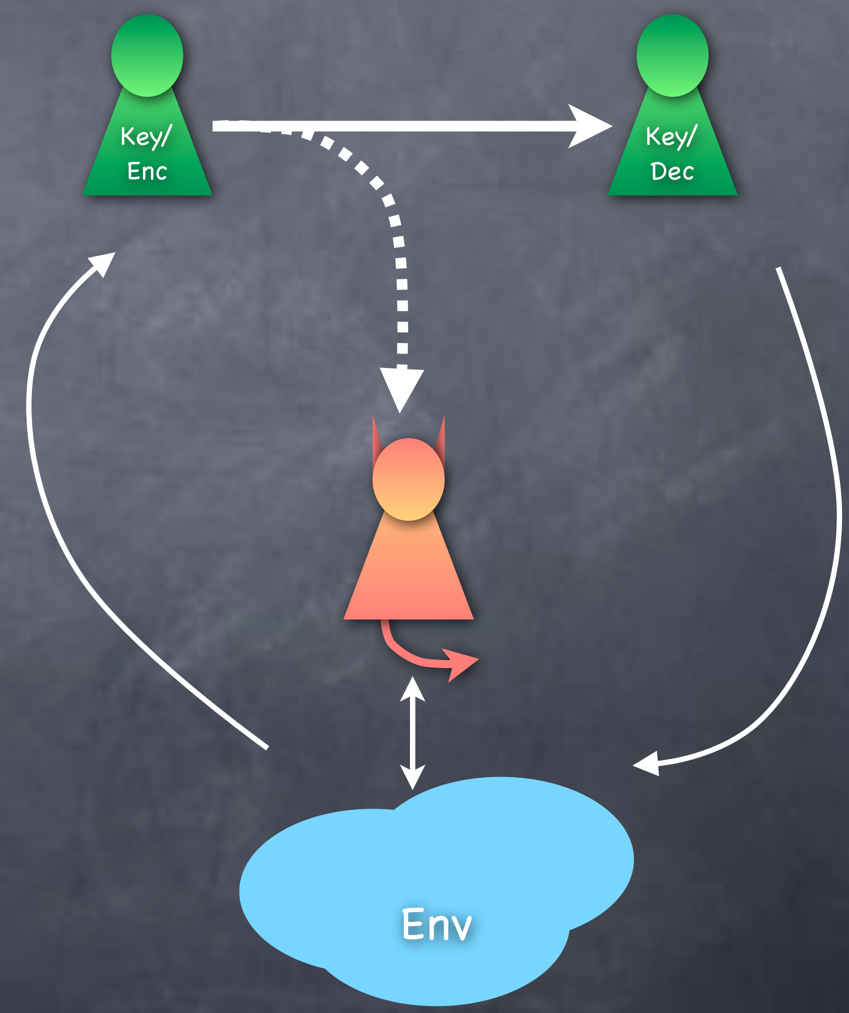


The Environment

- Where does the message come from?
  - Eve might already have partial information about the message, or might receive such information later
  - In fact, Eve might influence the choice of the message
- The environment
  - Includes the operating systems and other programs run by the participants, as well as other parties, if in a network
  - Abstract entity from which the input comes and to which the output goes. Arbitrarily influenced by Eve

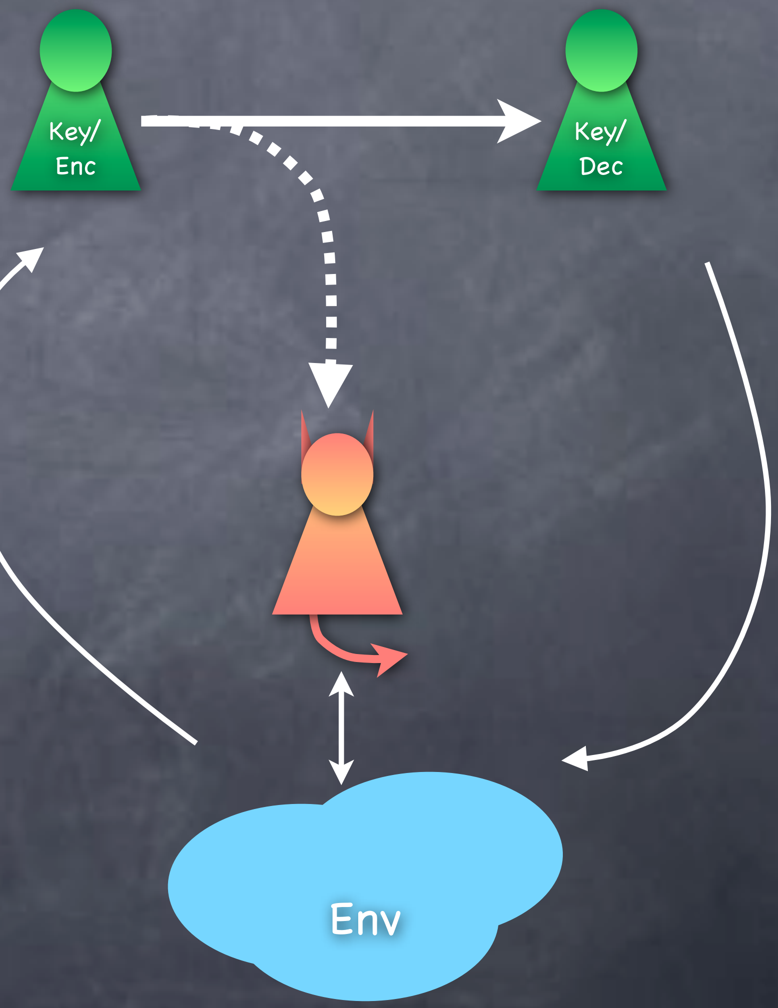


Defining Security



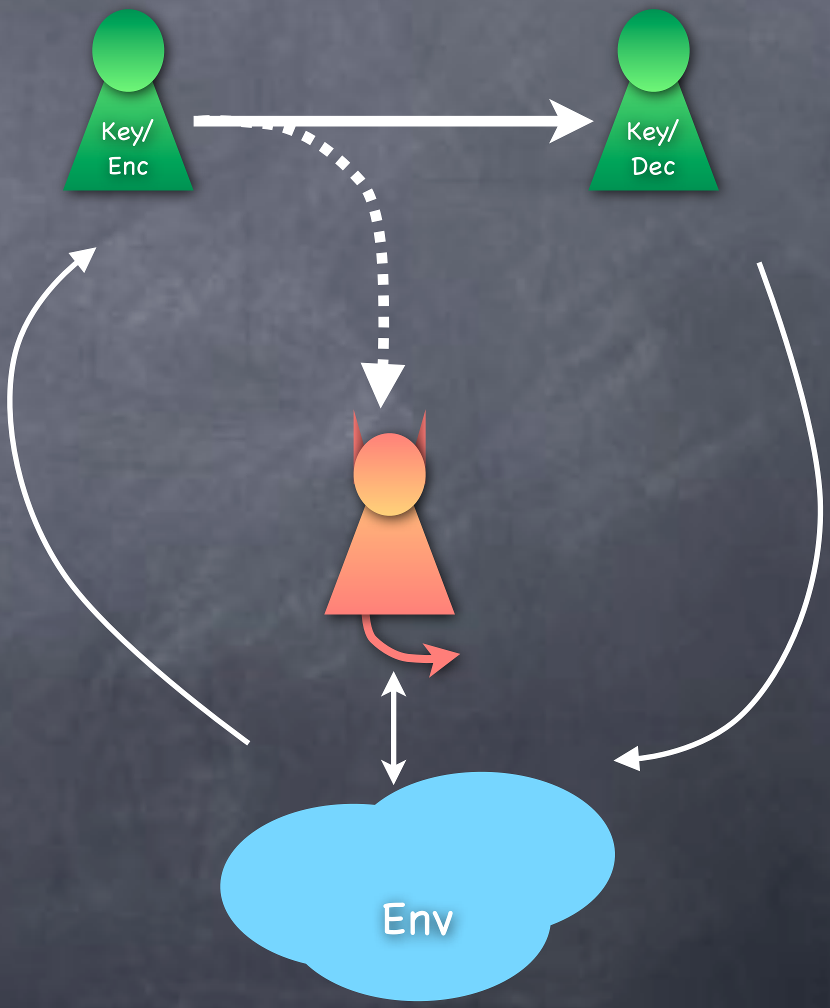
Defining Security

- Eve shouldn't be able to produce any "bad effects" in any environment



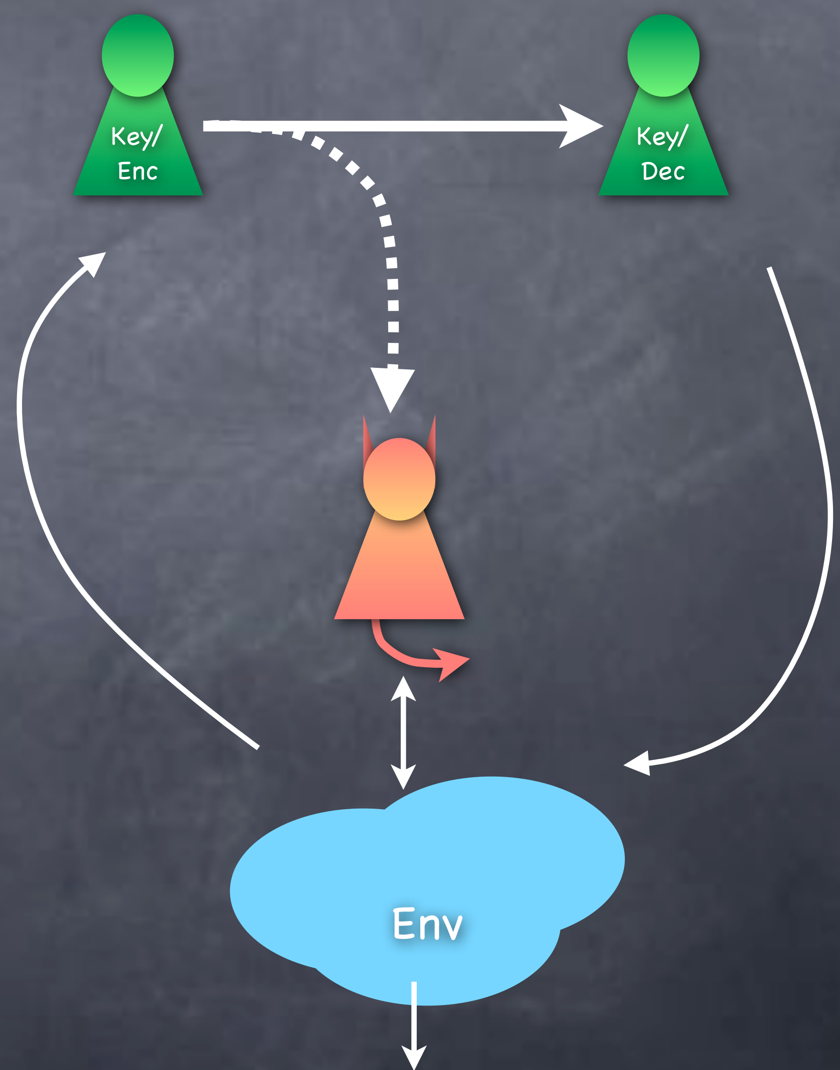
Defining Security

- Eve shouldn't be able to produce any "bad effects" in any environment
  - Or increase the probability of "bad effects"



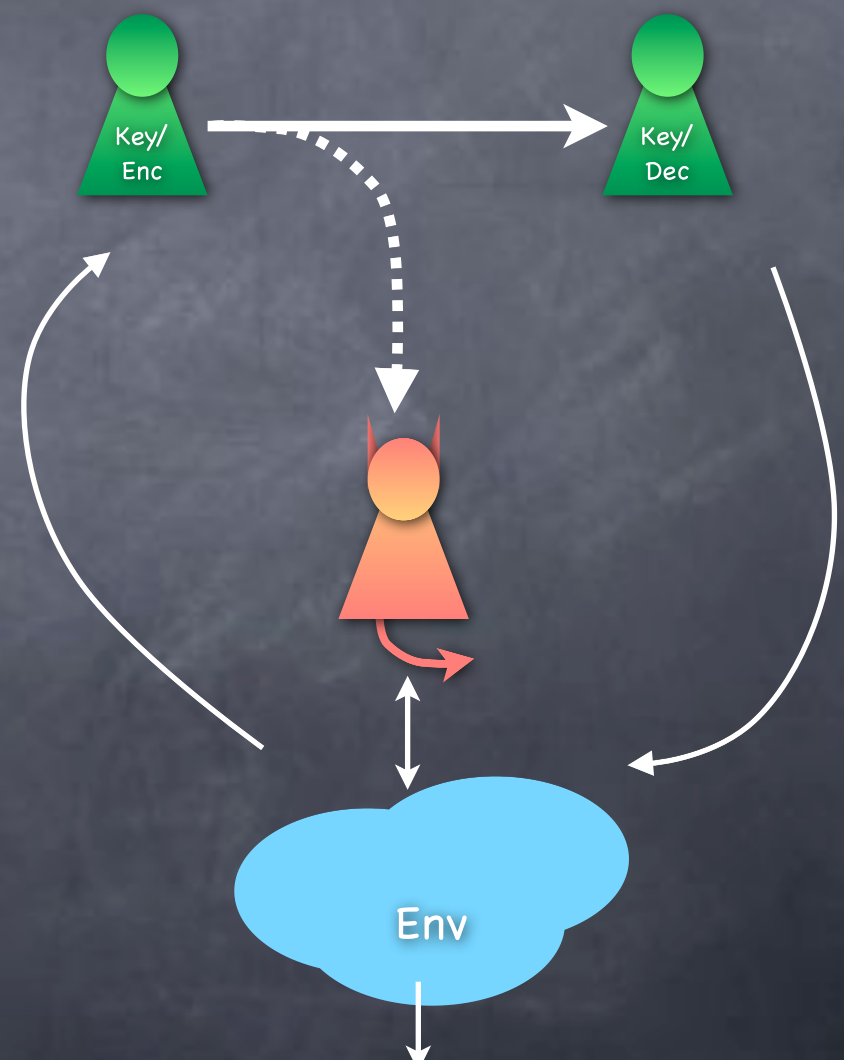
Defining Security

- Eve shouldn't be able to produce any "bad effects" in any environment
    - Or increase the probability of "bad effects"
- Effects in the environment: modeled as a bit in the environment (called the output bit)



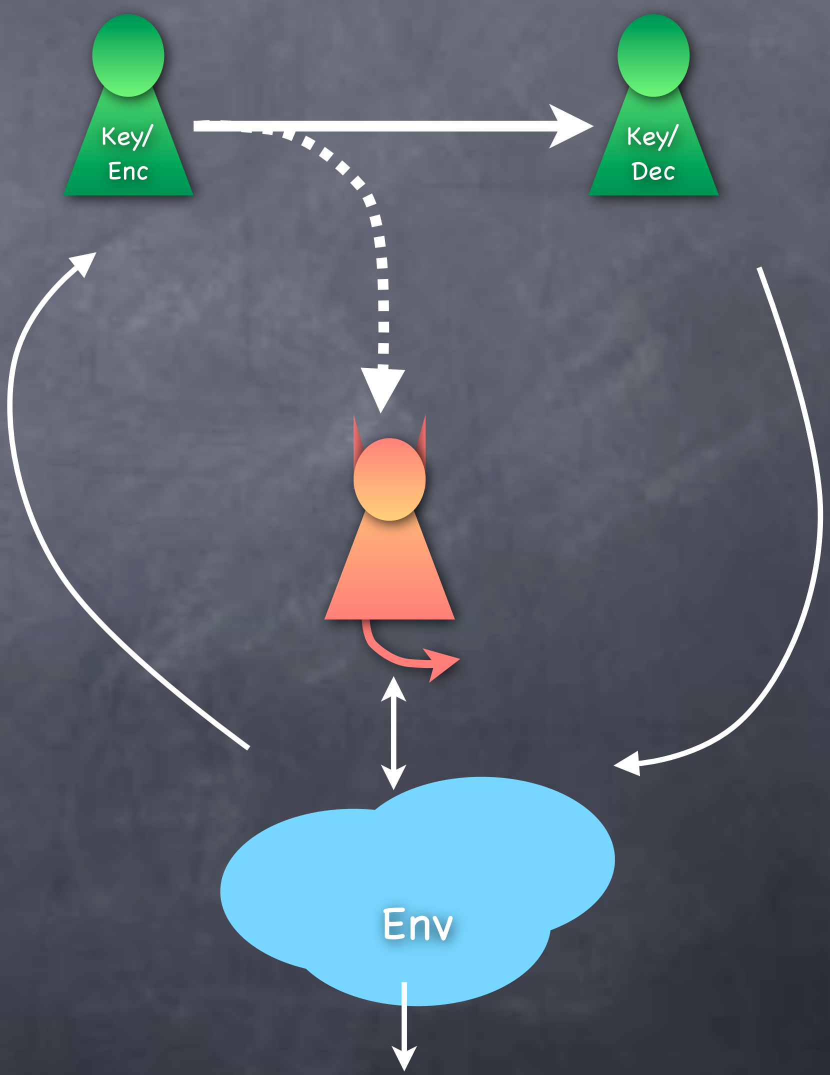
Defining Security

- Eve shouldn't be able to produce any "bad effects" in any environment
  - Or increase the probability of "bad effects"
- Effects in the environment: modeled as a bit in the environment (called the output bit)
- What is bad?



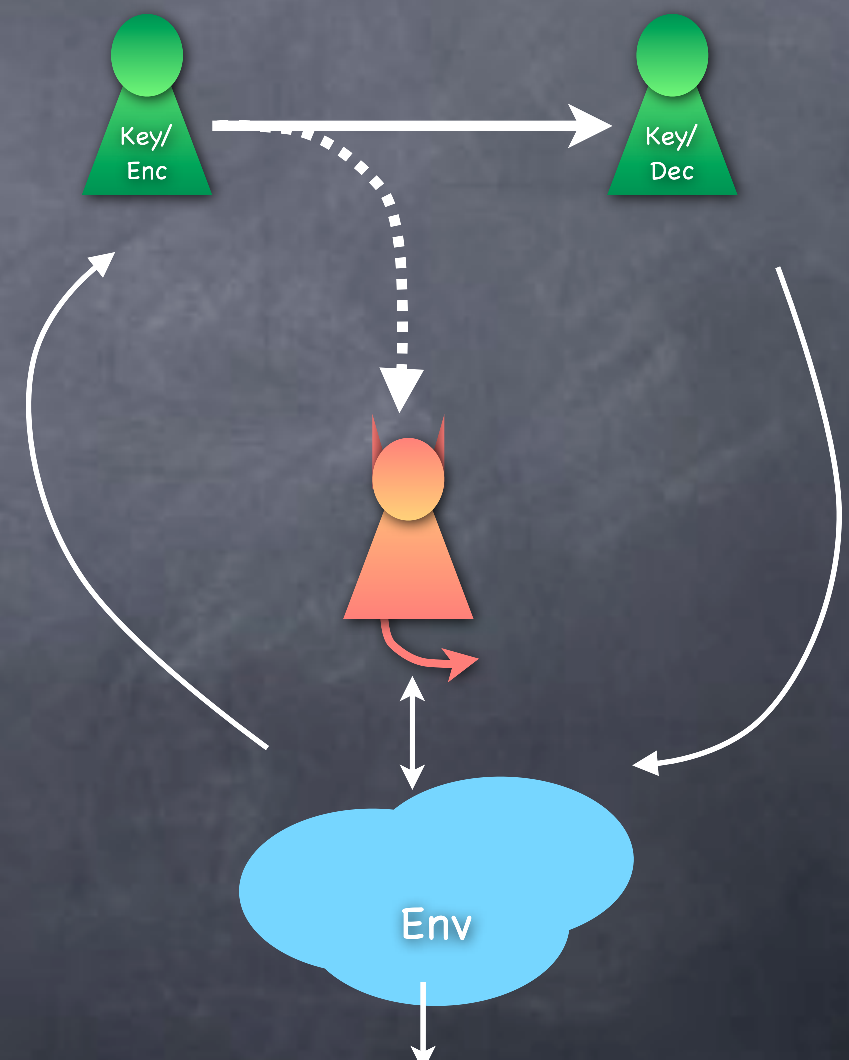
Defining Security

- Eve shouldn't be able to produce any "bad effects" in any environment
  - Or increase the probability of "bad effects"
- Effects in the environment: modeled as a bit in the environment (called the output bit)
- What is bad?
  - Anything that Eve couldn't have caused if an "ideal channel" was used



Defining Security

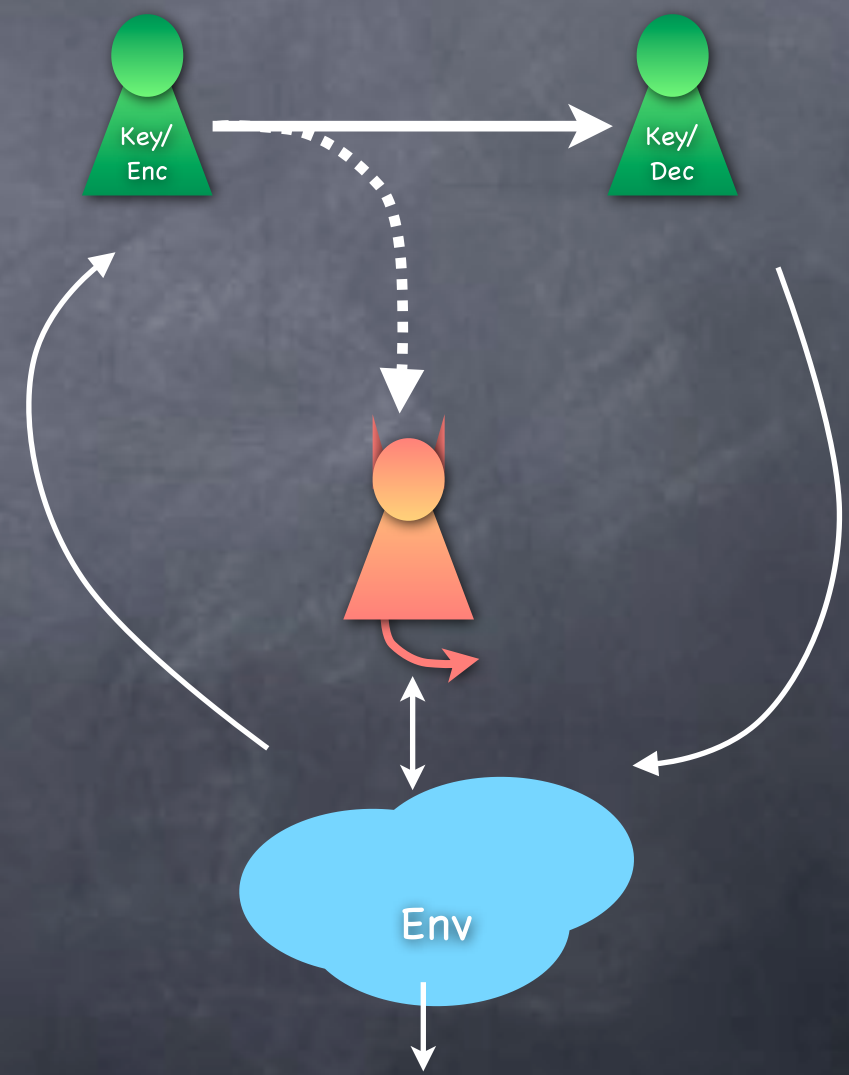
The REAL/IDEAL Paradigm



Defining Security

The REAL/IDEAL Paradigm

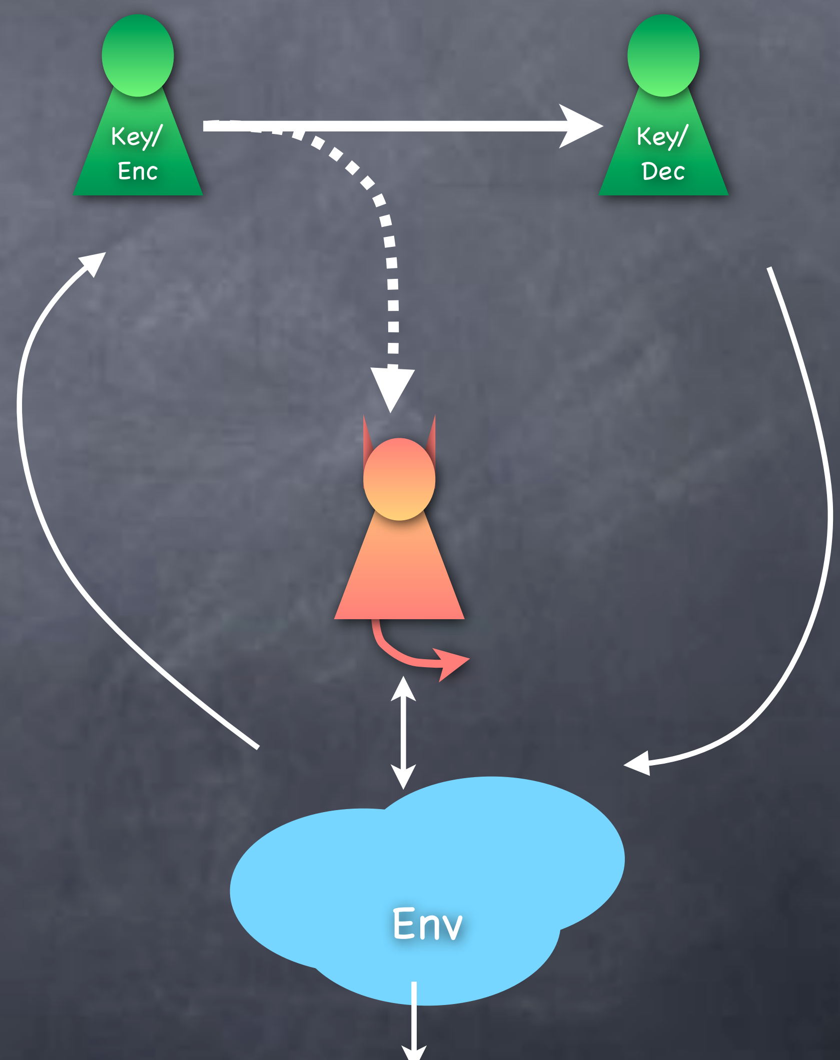
- Eve shouldn't produce any more effects than she could have in the ideal world



Defining Security

The REAL/IDEAL Paradigm

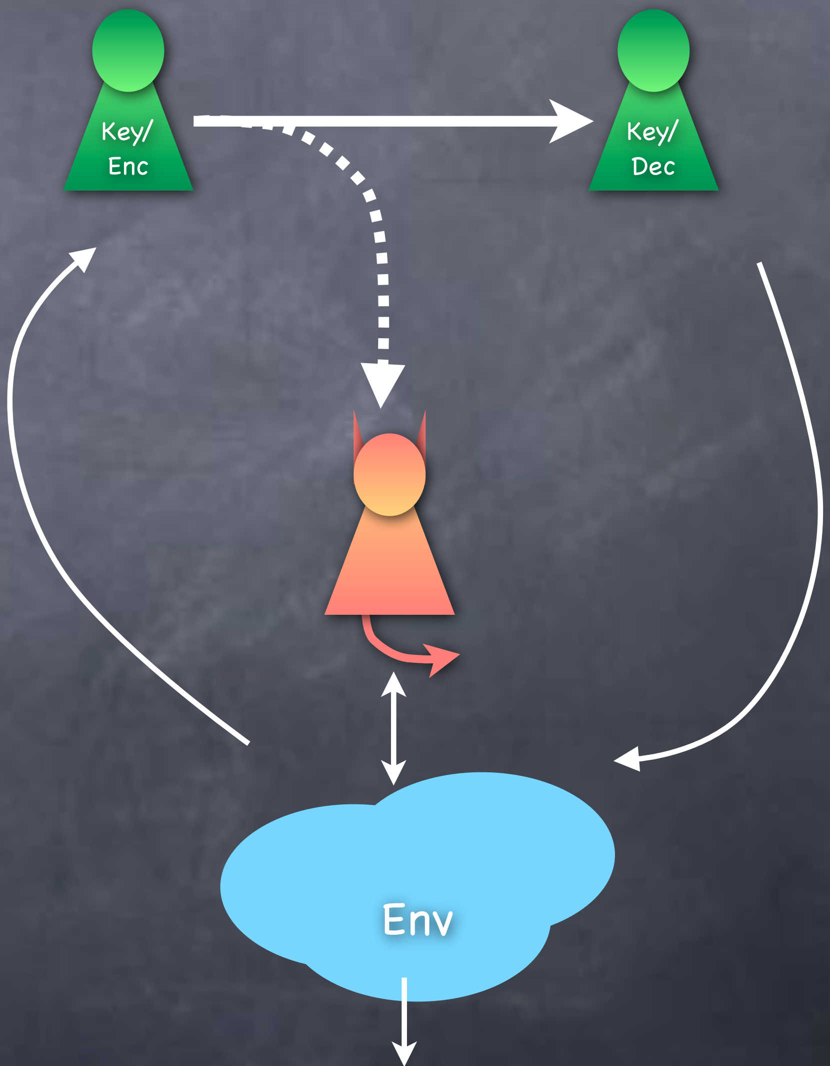
- Eve shouldn't produce any more effects than she could have in the ideal world
  - IDEAL world: Message sent over a (physically) secure channel. No encryption in this world.



Defining Security

The REAL/IDEAL Paradigm

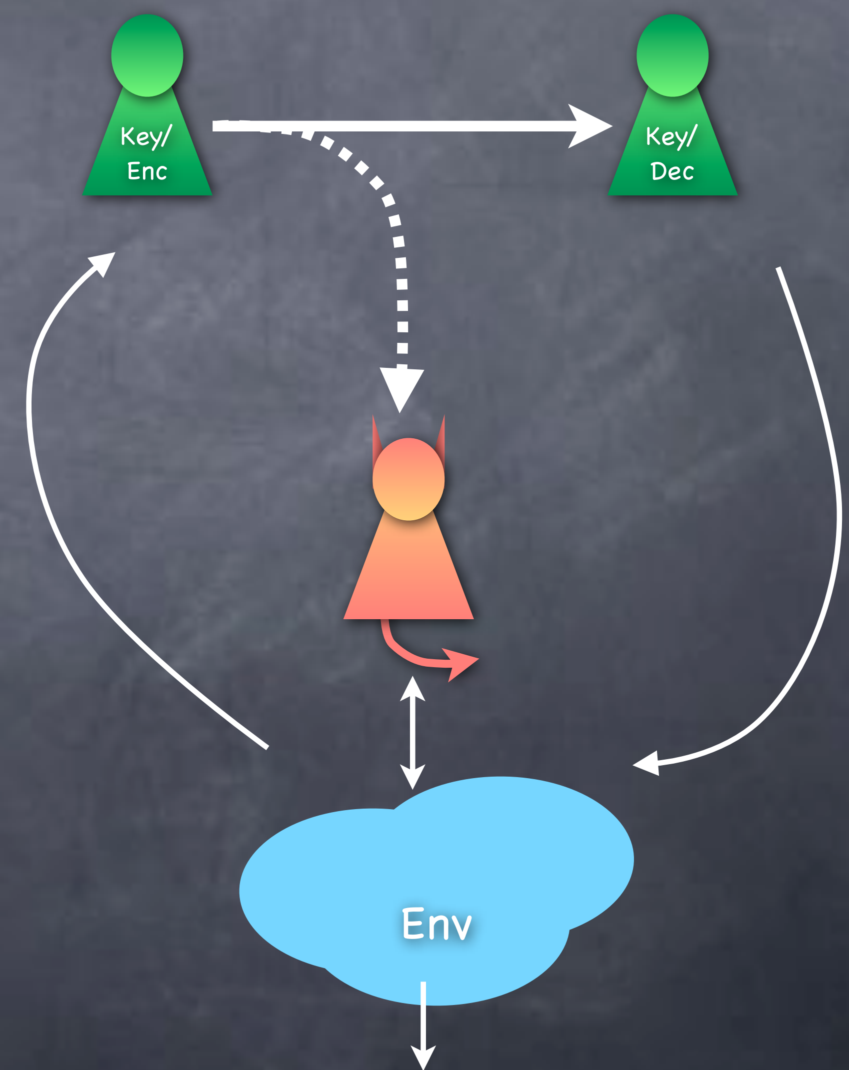
- Eve shouldn't produce any more effects than she could have in the ideal world
  - IDEAL world: Message sent over a (physically) secure channel. No encryption in this world.
  - REAL world: Using encryption



Defining Security

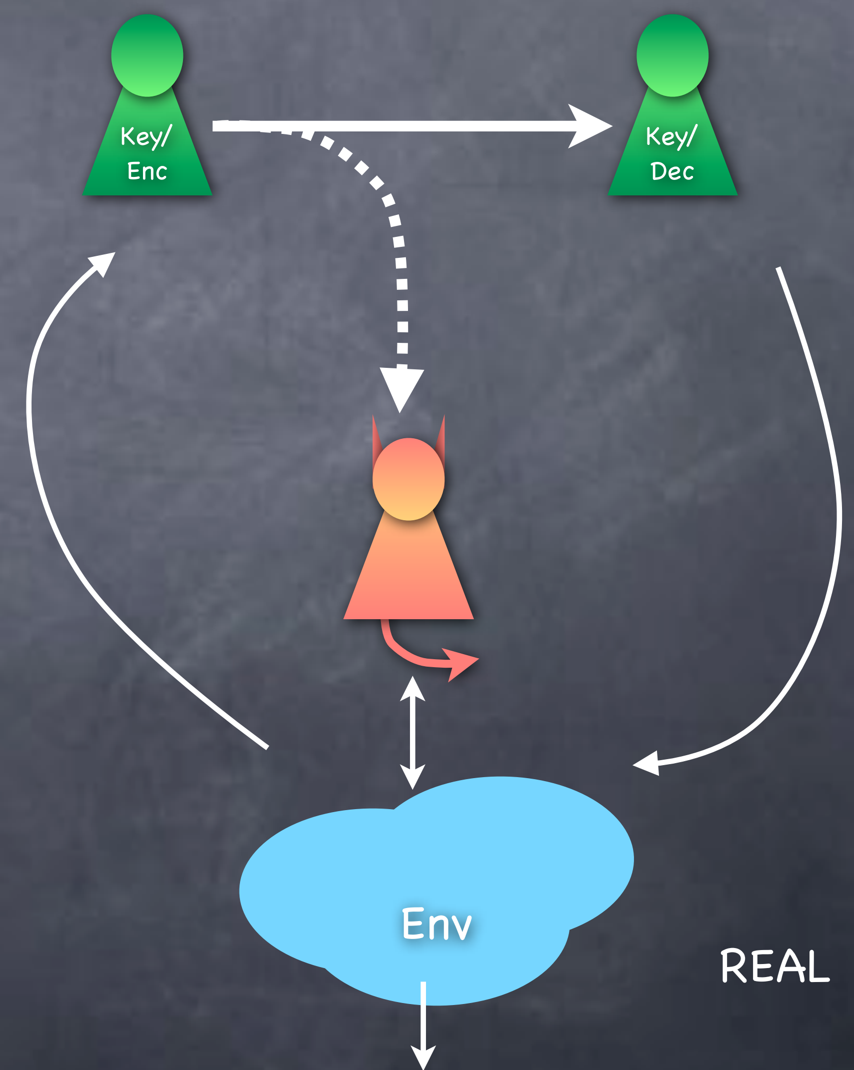
The REAL/IDEAL Paradigm

- Eve shouldn't produce any more effects than she could have in the ideal world
  - IDEAL world: Message sent over a (physically) secure channel. No encryption in this world.
  - REAL world: Using encryption
  - Encryption is secure if whatever Eve can do in the REAL world (using some strategy), she can do in the IDEAL world too (using an appropriate strategy)



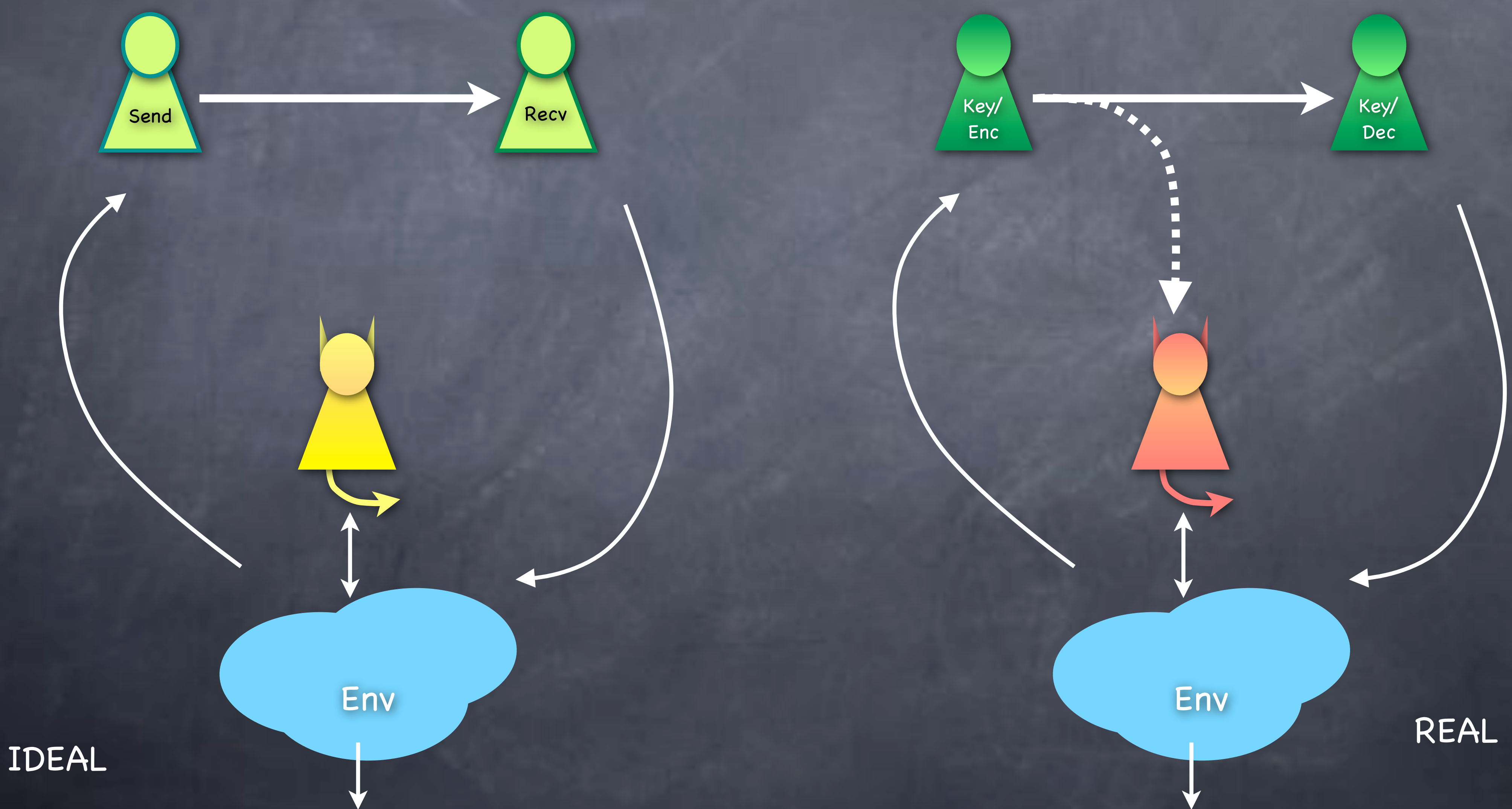
Defining Security

The REAL/IDEAL Paradigm



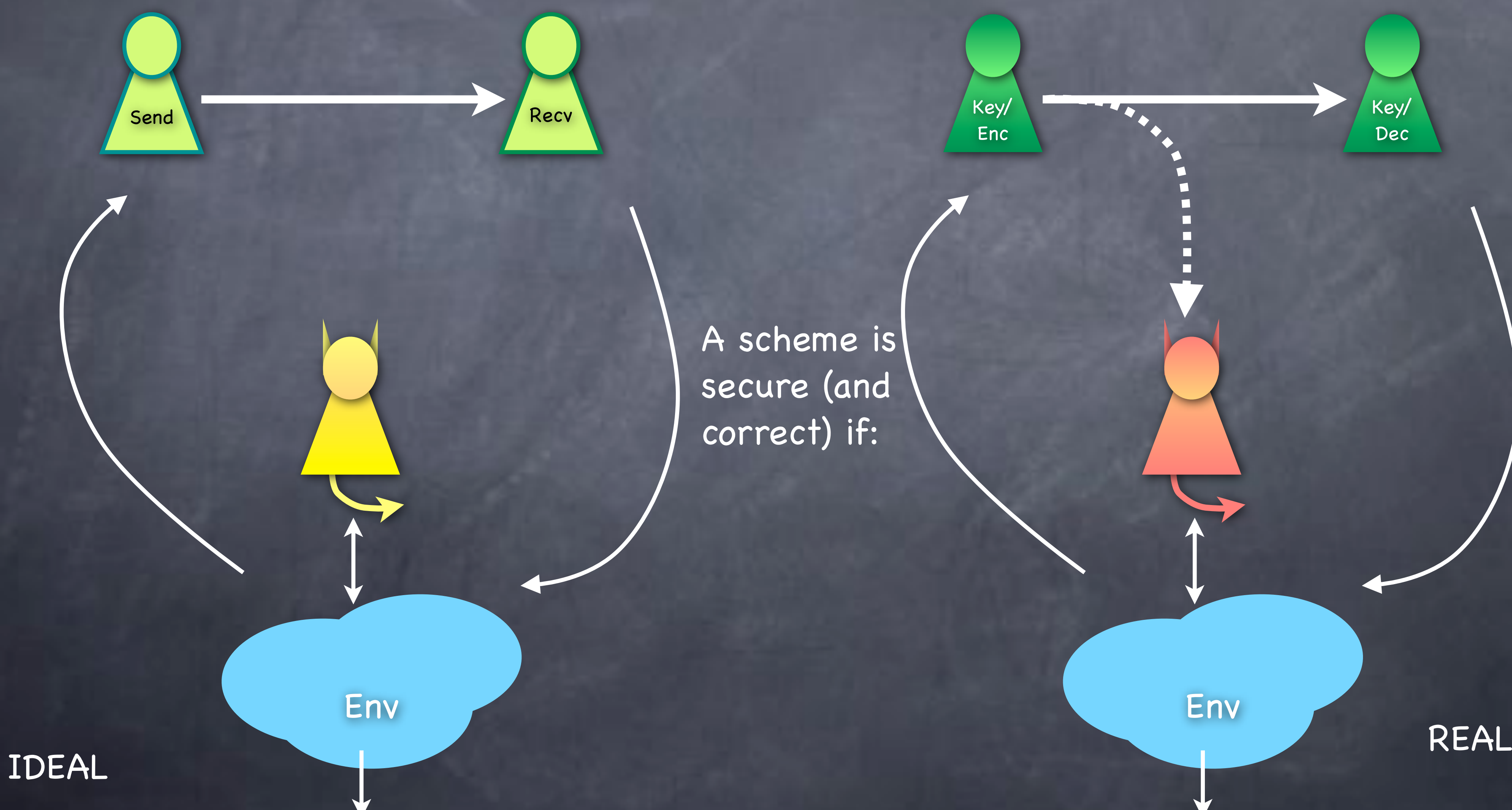
Defining Security

The REAL/IDEAL Paradigm



Defining Security

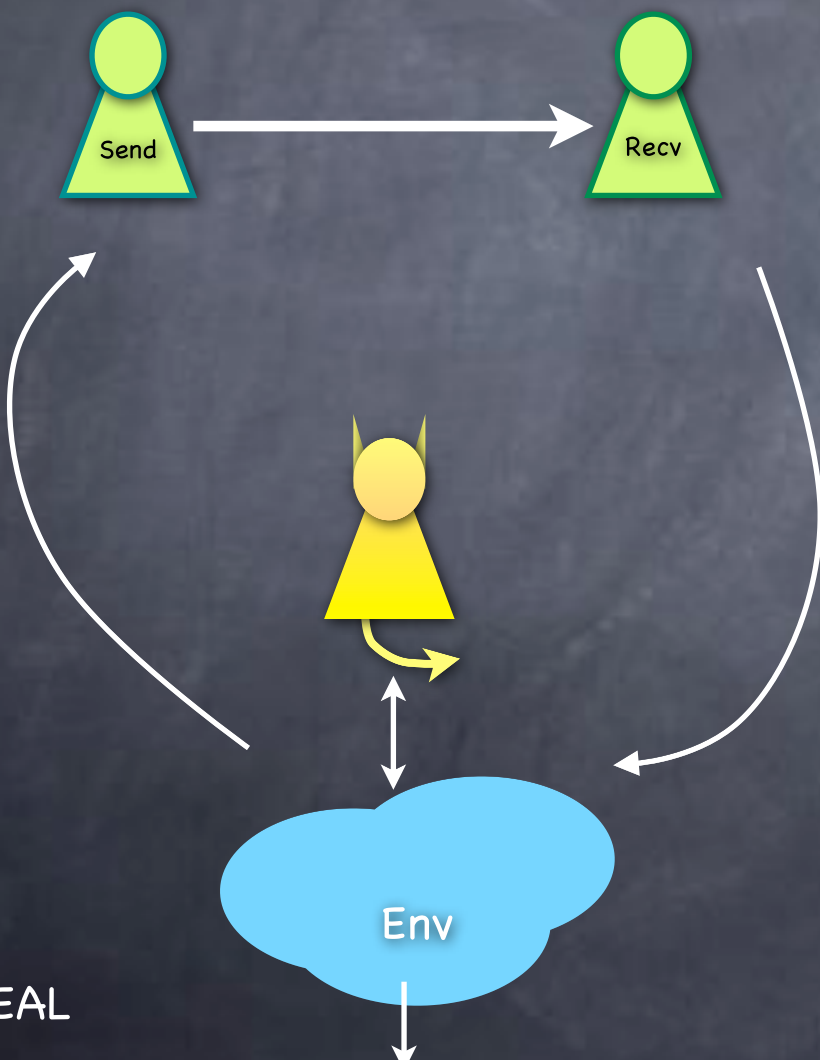
The REAL/IDEAL Paradigm



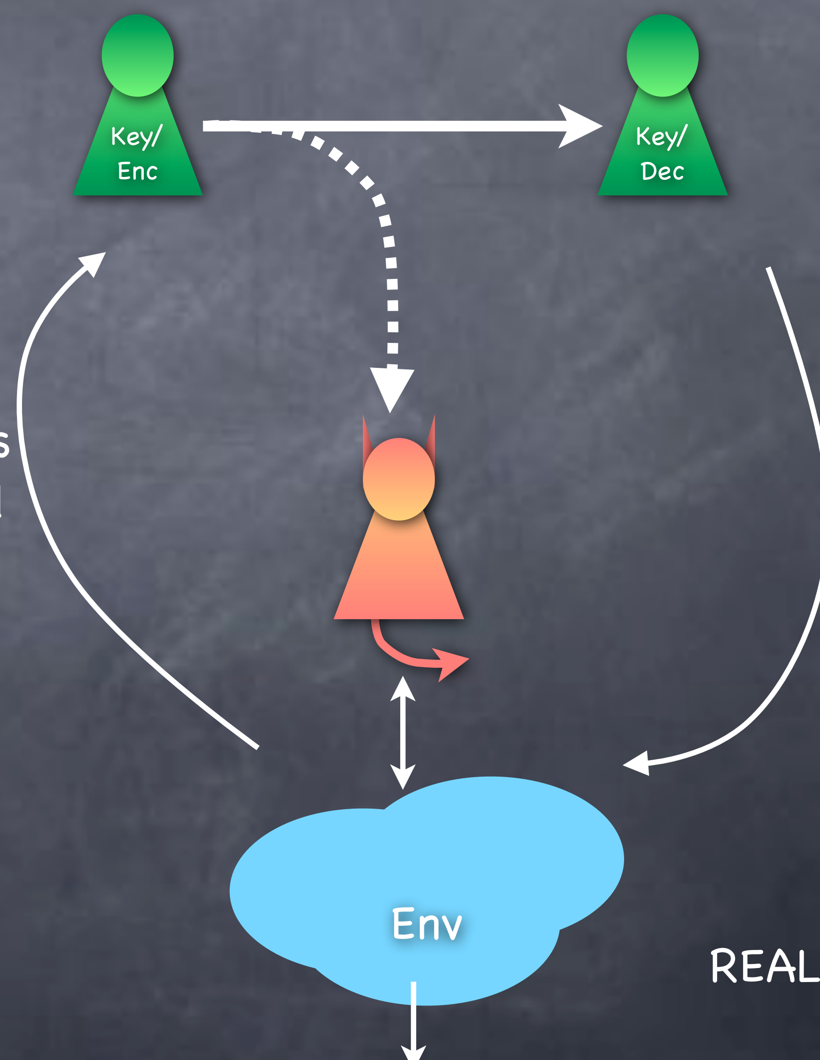
A scheme is secure (and correct) if:

Defining Security

The REAL/IDEAL Paradigm

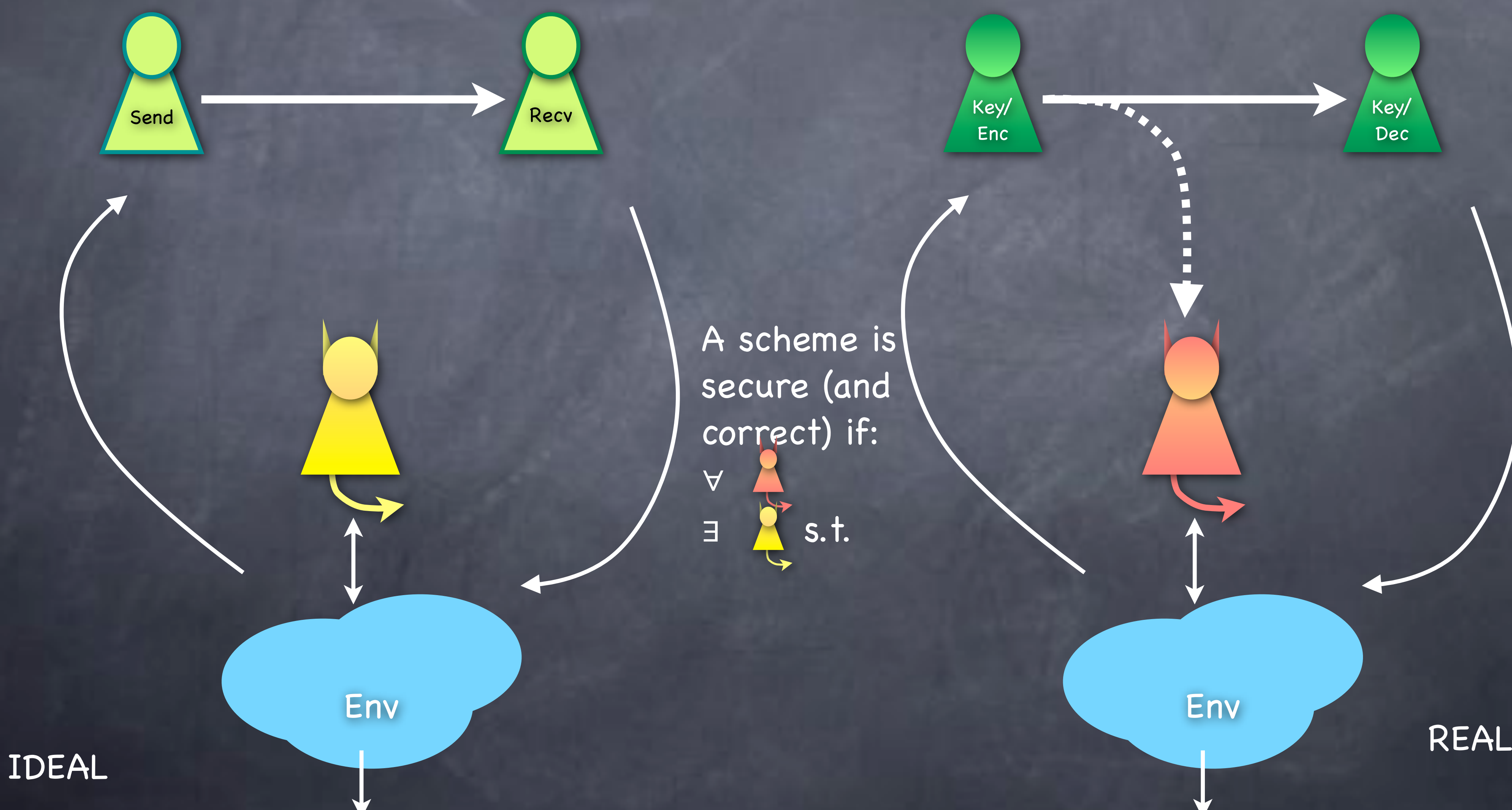


A scheme is secure (and correct) if:
∀



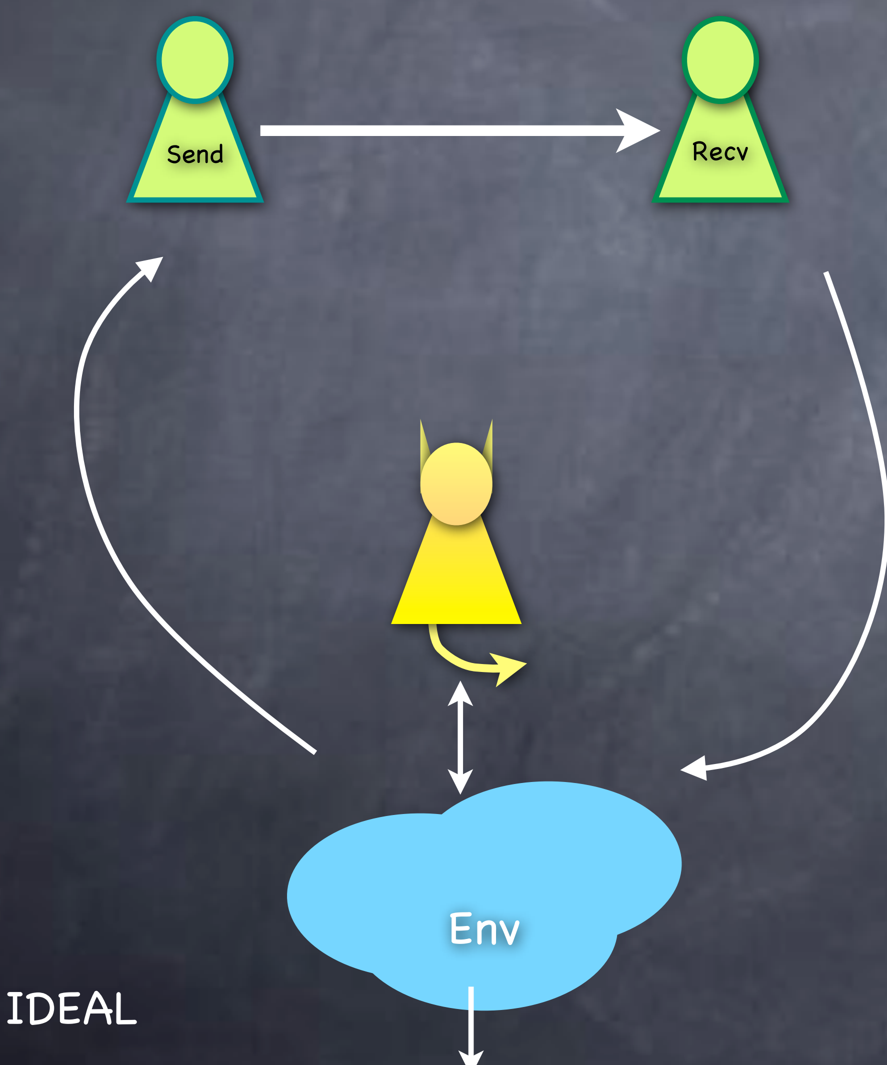
Defining Security

The REAL/IDEAL Paradigm

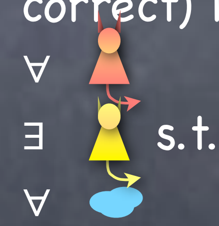


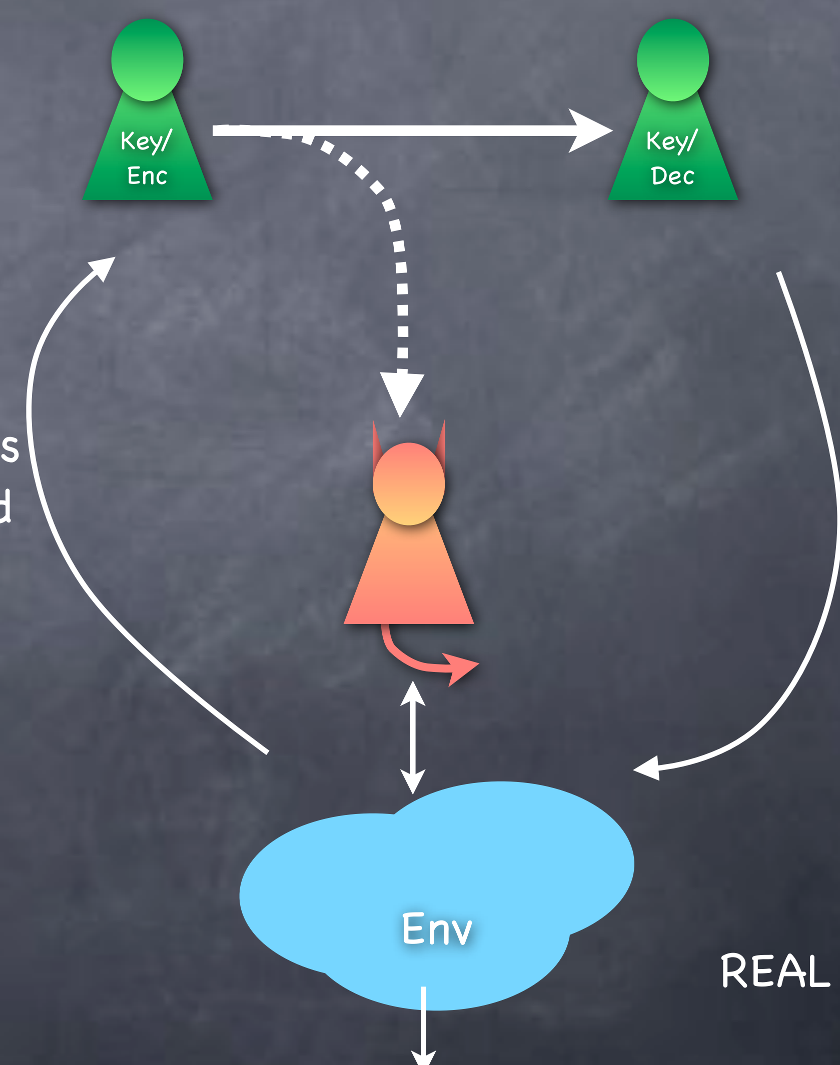
Defining Security

The REAL/IDEAL Paradigm



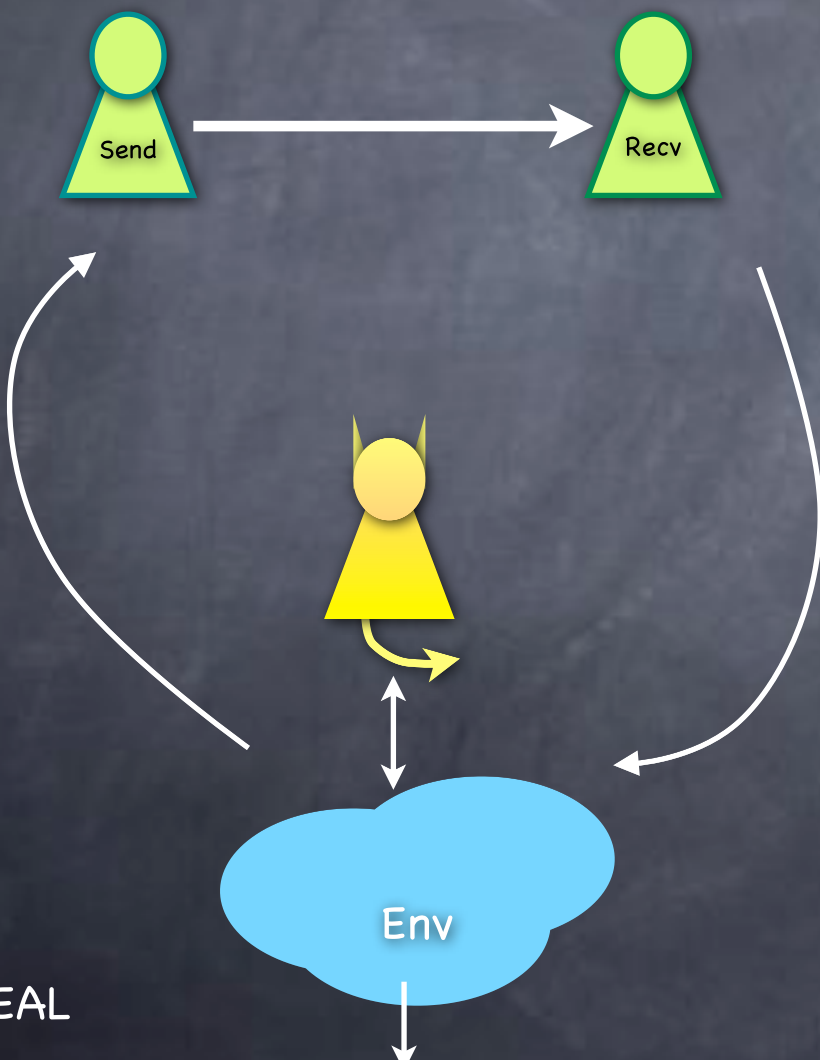
A scheme is secure (and correct) if:





Defining Security

The REAL/IDEAL Paradigm



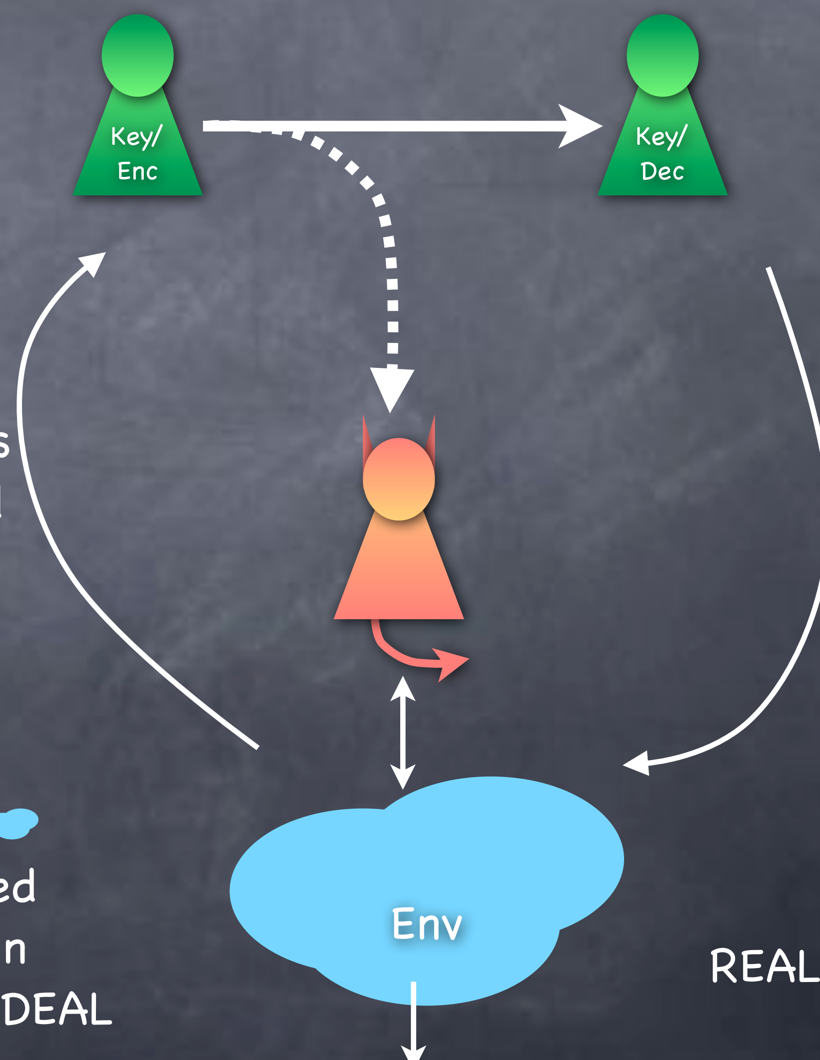
A scheme is secure (and correct) if:

$\forall$

$\exists$ s.t.

$\forall$

output of is distributed identically in REAL and IDEAL



Ready to go...

Ready to go...

- REAL/IDEAL (a.k.a simulation-based) security forms the basic template for a large variety of security definitions

Ready to go...

- REAL/IDEAL (a.k.a simulation-based) security forms the basic template for a large variety of security definitions
- We will see three definitions of symmetric-key encryption

Ready to go...

- REAL/IDEAL (a.k.a simulation-based) security forms the basic template for a large variety of security definitions
- We will see three definitions of symmetric-key encryption
  - Security of "one-time encryption"

Ready to go...

- REAL/IDEAL (a.k.a simulation-based) security forms the basic template for a large variety of security definitions
- We will see three definitions of symmetric-key encryption
  - Security of “one-time encryption”
  - Security of (muti-message) encryption

Ready to go...

- REAL/IDEAL (a.k.a simulation-based) security forms the basic template for a large variety of security definitions
- We will see three definitions of symmetric-key encryption
  - Security of "one-time encryption"
  - Security of (muti-message) encryption
  - Security against "active attacks"

Ready to go…

- REAL/IDEAL (a.k.a simulation-based) security forms the basic template for a large variety of security definitions
- We will see three definitions of symmetric-key encryption
  - Security of “one-time encryption”
  - Security of (muti-message) encryption
  - Security against “active attacks”
- Will also see alternate (but essentially equivalent) security definitions

Onetime Encryption

Onetime Encryption
The Syntax

- Shared-key (Private-key) Encryption
  - Key Generation: Randomized
    - K ← $\mathcal{K}$, uniformly randomly drawn from the key-space (or according to a key-distribution)
  - Encryption: Deterministic
    - Enc: $\mathcal{M} \times \mathcal{K} \to \mathcal{C}$
  - Decryption: Deterministic
    - Dec: $\mathcal{C} \times \mathcal{K} \to \mathcal{M}$

Onetime Encryption

Perfect Secrecy

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m′ $\in \mathcal{M}$
  - $\{\mathrm{Enc}(m,K)\}_{K \leftarrow \mathrm{KeyGen}} = \{\mathrm{Enc}(m',K)\}_{K \leftarrow \mathrm{KeyGen}}$

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m′ $\in \mathcal{M}$
  - $\{\mathrm{Enc}(m,K)\}_{K\leftarrow \mathrm{KeyGen}} = \{\mathrm{Enc}(m',K)\}_{K\leftarrow \mathrm{KeyGen}}$

Distribution of the ciphertext

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m′ $\in \mathcal{M}$
  - $\{\mathrm{Enc}(m,K)\}_{K \leftarrow \mathrm{KeyGen}} = \{\mathrm{Enc}(m',K)\}_{K \leftarrow \mathrm{KeyGen}}$

Distribution of the ciphertext

$\mathcal{M}$ \ $\mathcal{K}$	0	1	2	3
a	x	y	y	z
b	y	x	z	y

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m′ $\in \mathcal{M}$
  - $\{Enc(m,K)\}_{K\leftarrow KeyGen} = \{Enc(m',K)\}_{K\leftarrow KeyGen}$
- Distribution of the ciphertext is defined by the randomness in the key

$\mathcal{M}$ \ $\mathcal{K}$	0	1	2	3
a	x	y	y	z
b	y	x	z	y

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m′ $\in \mathcal{M}$
  - $\{\text{Enc}(m,K)\}_{K \leftarrow \text{KeyGen}} = \{\text{Enc}(m',K)\}_{K \leftarrow \text{KeyGen}}$
- Distribution of the ciphertext is defined by the randomness in the key

$\mathcal{M}$ \ $\mathcal{K}$	0	1	2	3
a	x	y	y	z
b	y	x	z	y

Assuming K uniformly drawn from $\mathcal{K}$

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m' $\in \mathcal{M}$
  - $\{\text{Enc}(m,K)\}_{K\leftarrow\text{KeyGen}} = \{\text{Enc}(m',K)\}_{K\leftarrow\text{KeyGen}}$
- Distribution of the ciphertext is defined by the randomness in the key

$\mathcal{M}$ \ $\mathcal{K}$	0	1	2	3
a	x	y	y	z
b	y	x	z	y

Assuming K uniformly drawn from $\mathcal{K}$

Pr[Enc(a,K)=x] = ¼,
Pr[Enc(a,K)=y] = ½,
Pr[Enc(a,K)=z] = ¼

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m′ $\in \mathcal{M}$
  - $\{Enc(m,K)\}_{K \leftarrow KeyGen} = \{Enc(m',K)\}_{K \leftarrow KeyGen}$
- Distribution of the ciphertext is defined by the randomness in the key

$\mathcal{M} \backslash \mathcal{K}$	0	1	2	3
a	x	y	y	z
b	y	x	z	y

Assuming K uniformly drawn from $\mathcal{K}$

Pr[Enc(a,K)=x] = ¼,
Pr[Enc(a,K)=y] = ½,
Pr[Enc(a,K)=z] = ¼

Same for Enc(b,K).

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m′ $\in \mathcal{M}$
  - $\{Enc(m,K)\}_{K \leftarrow KeyGen} = \{Enc(m',K)\}_{K \leftarrow KeyGen}$
- Distribution of the ciphertext is defined by the randomness in the key
- In addition, require correctness
  - $\forall$ m, K,   Dec(Enc(m,K), K) = m

$\mathcal{M}$ \ $\mathcal{K}$	0	1	2	3
a	x	y	y	z
b	y	x	z	y

Assuming K uniformly drawn from $\mathcal{K}$

Pr[Enc(a,K)=x] = ¼,

Pr[Enc(a,K)=y] = ½,

Pr[Enc(a,K)=z] = ¼

Same for Enc(b,K).

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m′ $\in \mathcal{M}$
  - $\{Enc(m,K)\}_{K \leftarrow KeyGen} = \{Enc(m',K)\}_{K \leftarrow KeyGen}$
- Distribution of the ciphertext is defined by the randomness in the key
- In addition, require correctness
  - $\forall$ m, K, Dec(Enc(m,K), K) = m
- E.g. One-time pad: $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^n$ and Enc(m,K) = m$\oplus$K, Dec(c,K) = c$\oplus$K

$\mathcal{M}$ \ $\mathcal{K}$	0	1	2	3
a	x	y	y	z
b	y	x	z	y

Assuming K uniformly drawn from $\mathcal{K}$

Pr[Enc(a,K)=x] = ¼,

Pr[Enc(a,K)=y] = ½,

Pr[Enc(a,K)=z] = ¼

Same for Enc(b,K).

Onetime Encryption

Perfect Secrecy

- Perfect secrecy: $\forall$ m, m′ $\in \mathcal{M}$
  - $\{Enc(m,K)\}_{K \leftarrow KeyGen} = \{Enc(m',K)\}_{K \leftarrow KeyGen}$
- Distribution of the ciphertext is defined by the randomness in the key
- In addition, require correctness
  - $\forall$ m, K, Dec(Enc(m,K), K) = m
- E.g. One-time pad: $\mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^n$ and Enc(m,K) = m$\oplus$K, Dec(c,K) = c$\oplus$K
  - More generally $\mathcal{M} = \mathcal{K} = \mathcal{C} = \mathcal{G}$ (a finite group) and Enc(m,K) = m+K, Dec(c,K) = c-K

$\mathcal{M}$ \ $\mathcal{K}$	0	1	2	3
a	x	y	y	z
b	y	x	z	y

Assuming K uniformly drawn from $\mathcal{K}$

Pr[Enc(a,K)=x] = ¼,

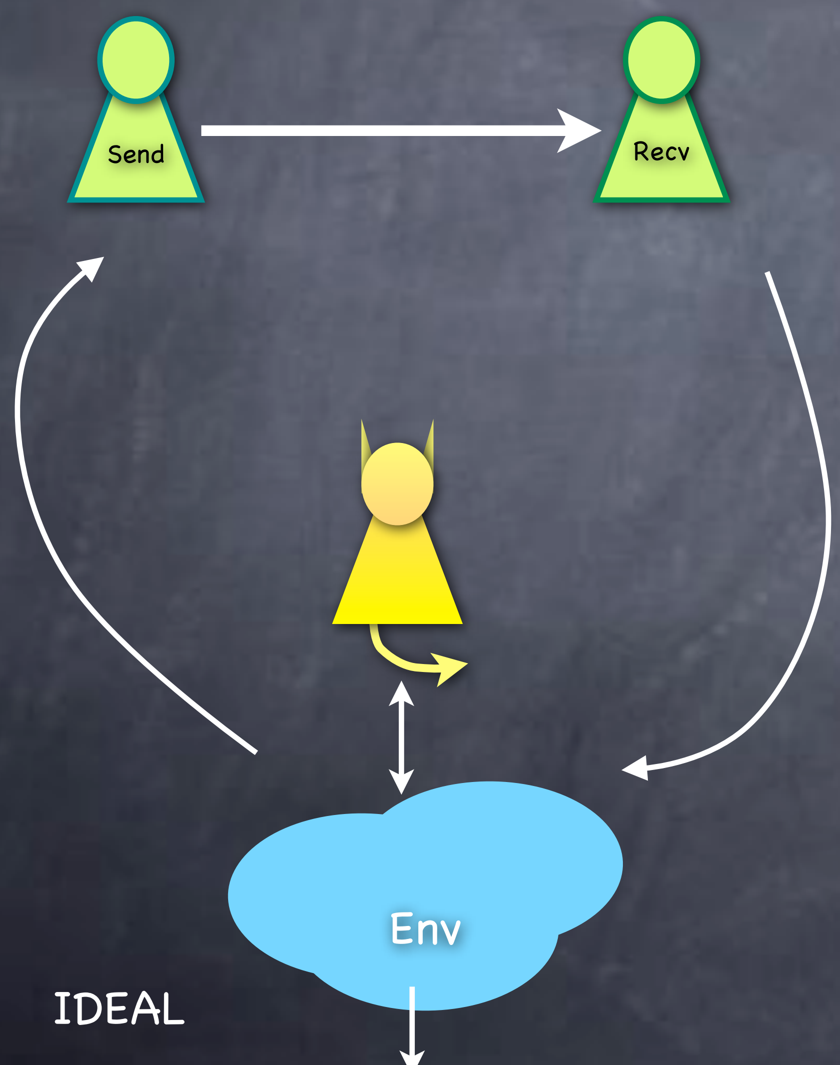
Pr[Enc(a,K)=y] = ½,

Pr[Enc(a,K)=z] = ¼

Same for Enc(b,K).

Onetime Encryption

SIM-Onetime Security



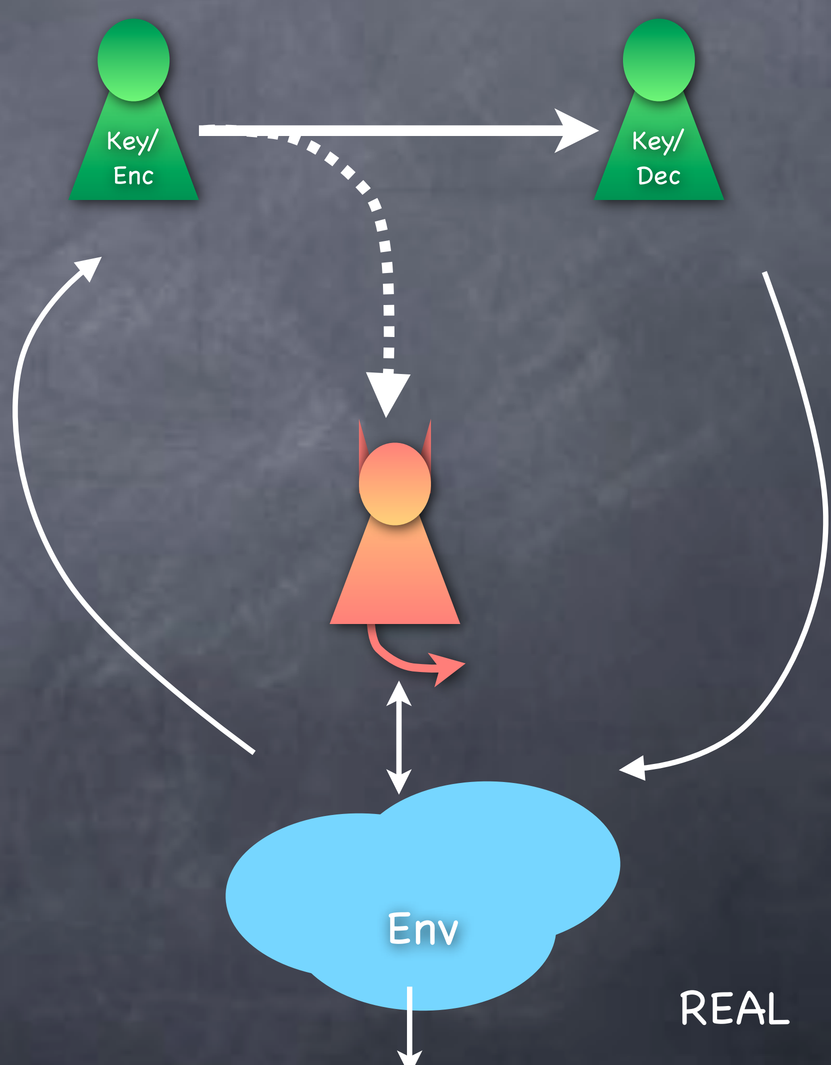
SIM-Onetime secure if:

∀ 

∃  s.t.

∀ 

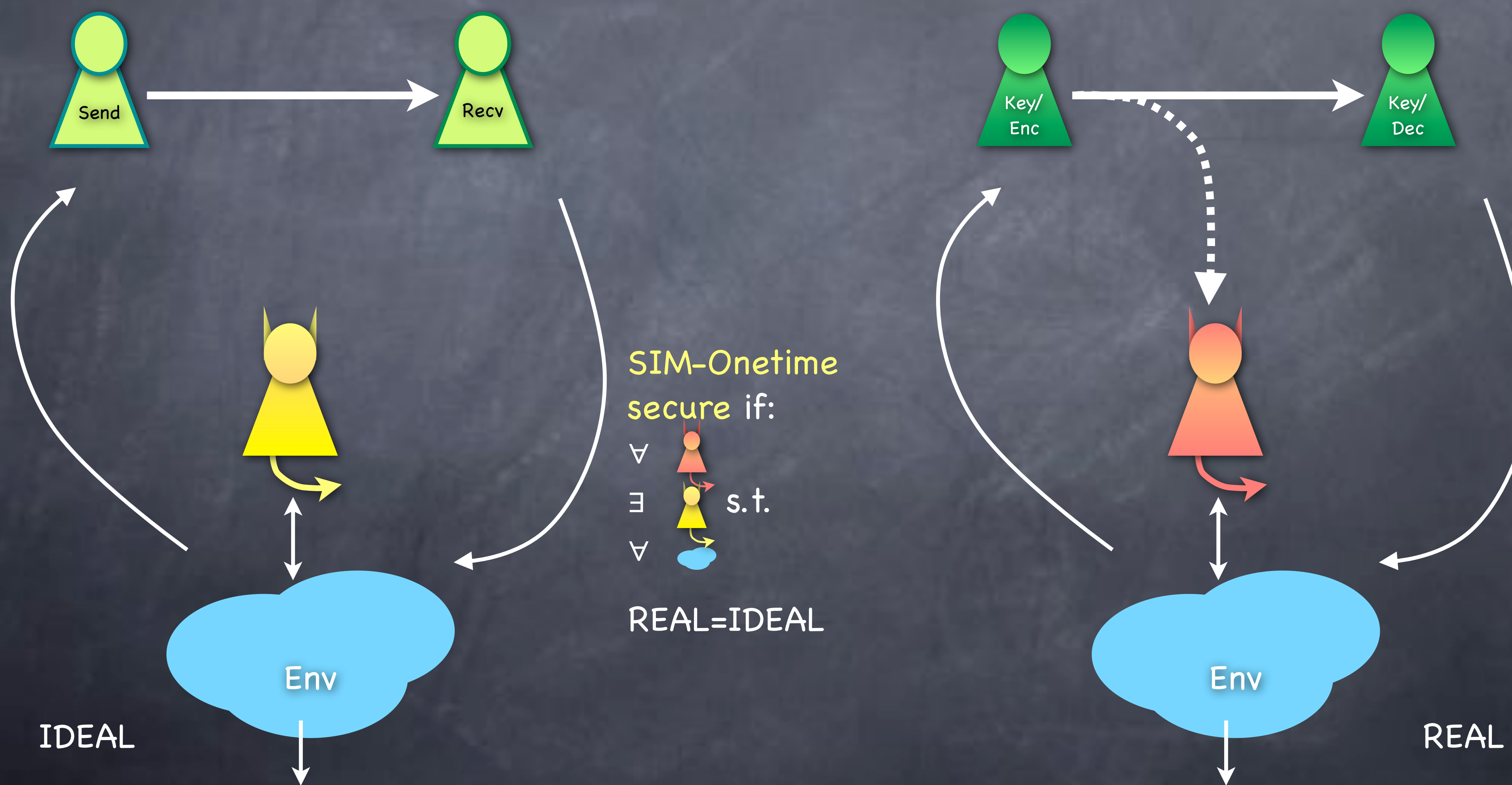
REAL=IDEAL



Onetime Encryption

SIM-Onetime Security

- Class of environments which send only one message

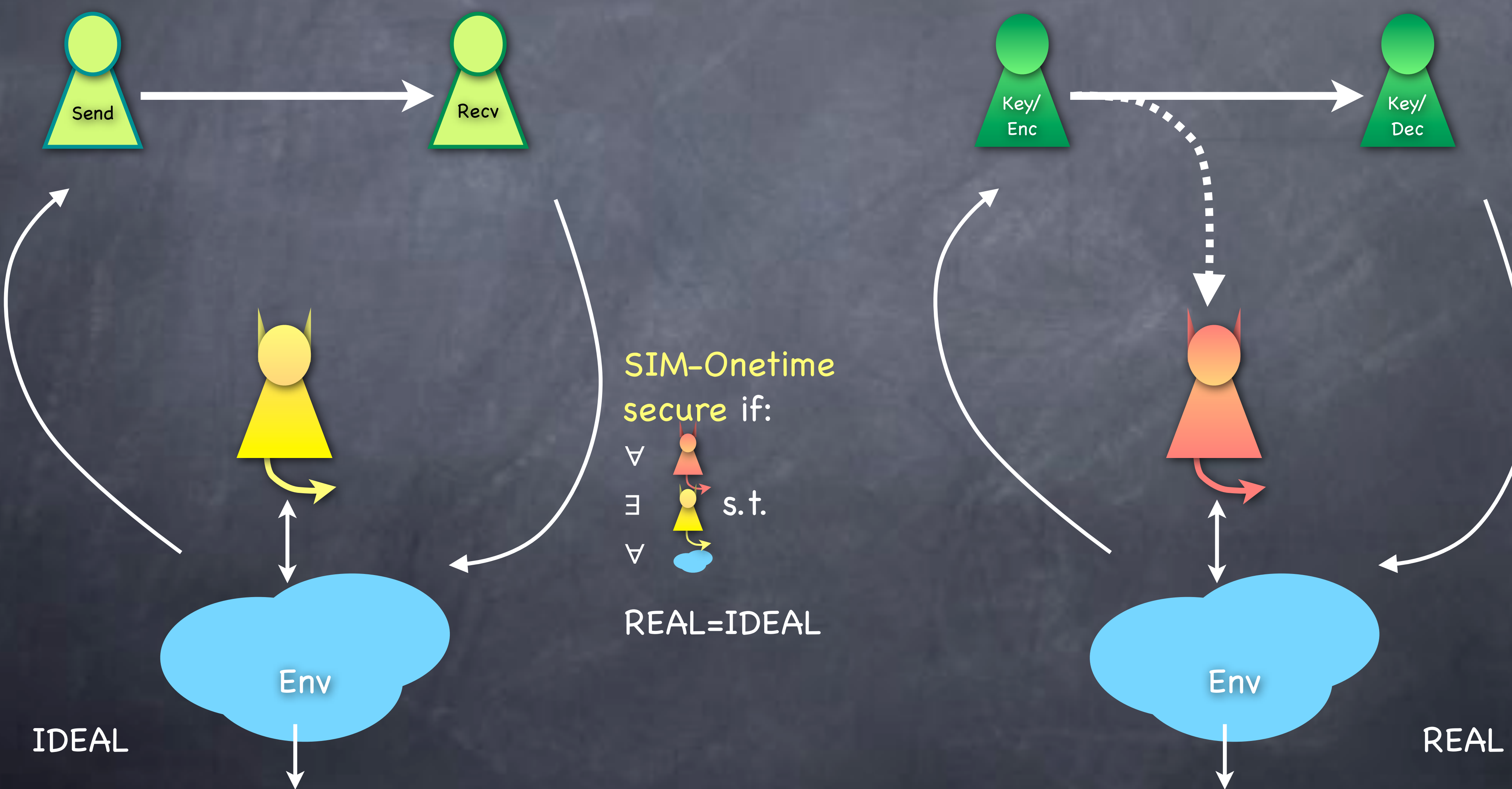


Onetime Encryption

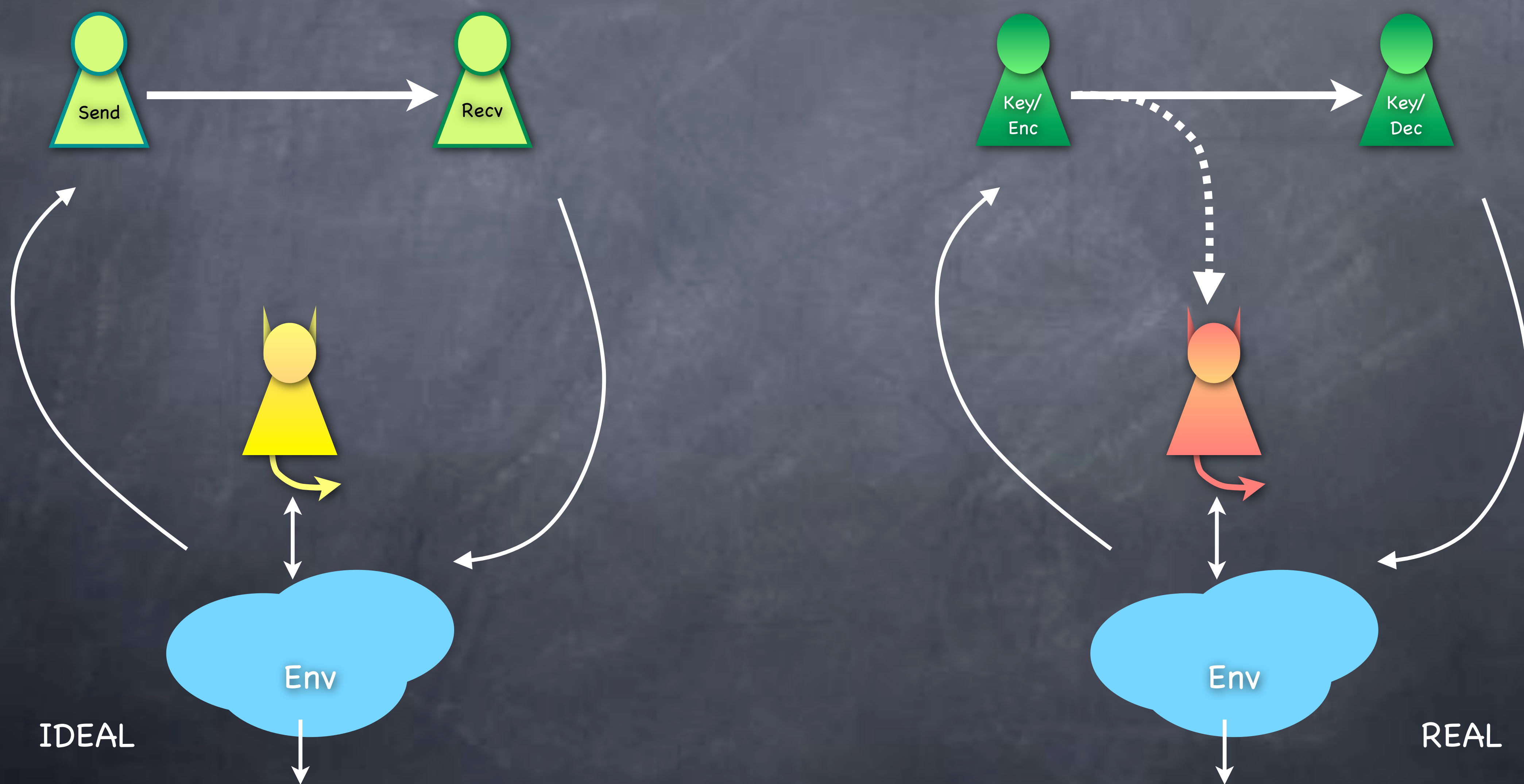
SIM-Onetime Security

Equivalent to perfect secrecy + correctness

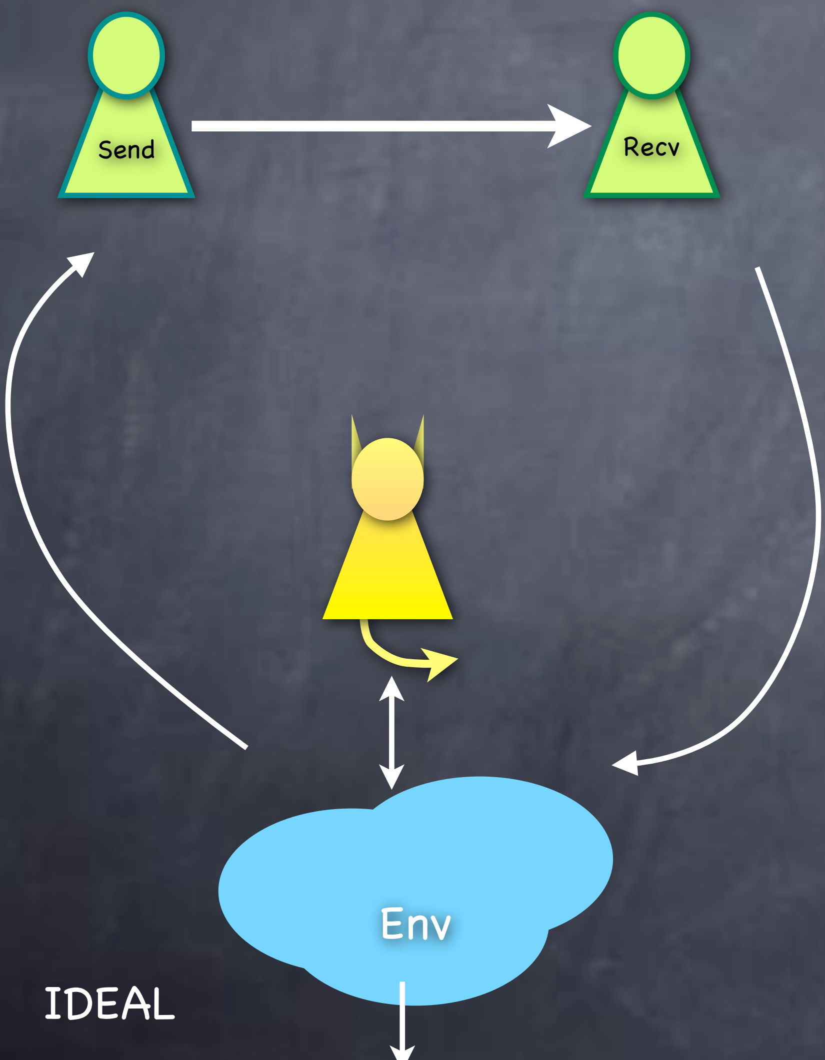
- Class of environments which send only one message



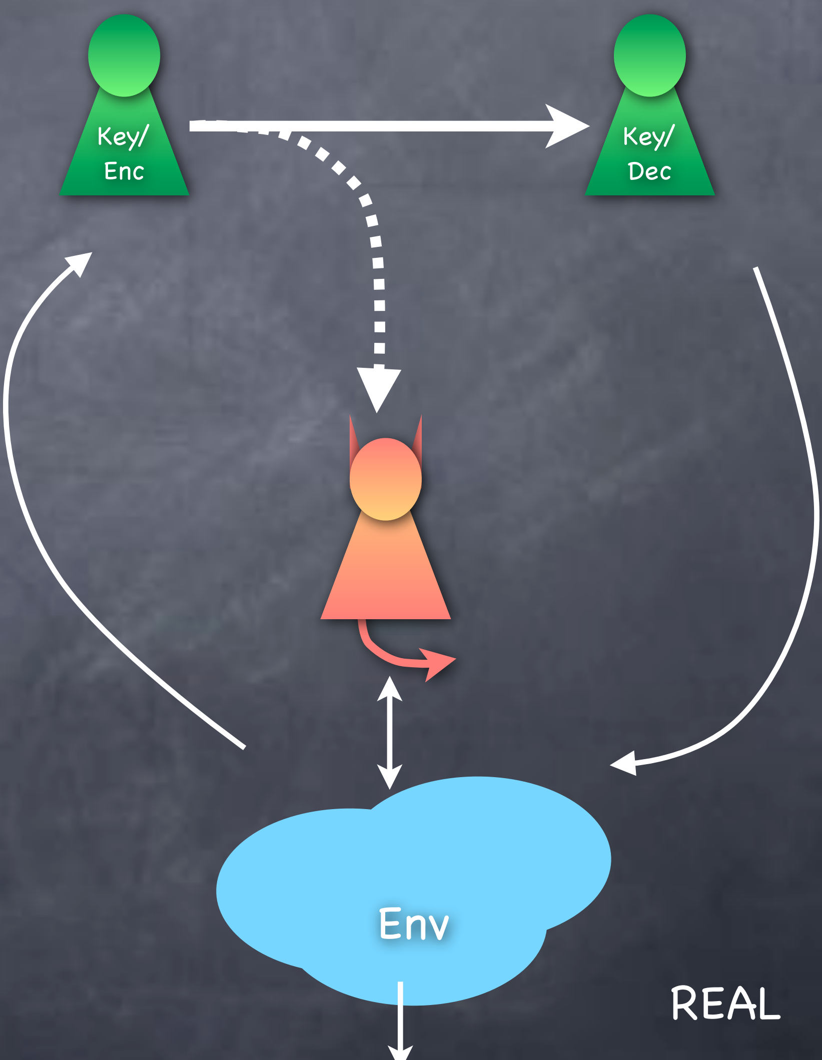
Perfect Secrecy + Correctness ⇒ SIM-Onetime Security



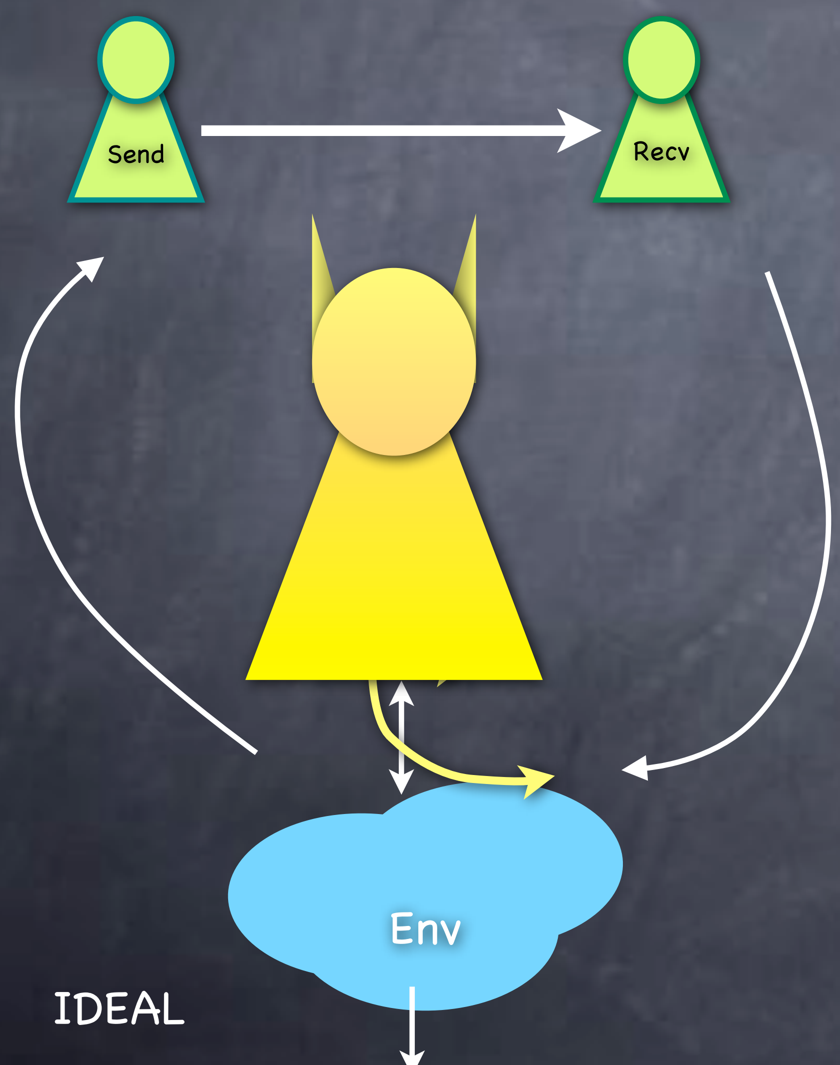
Perfect Secrecy + Correctness ⇒ SIM-Onetime Security



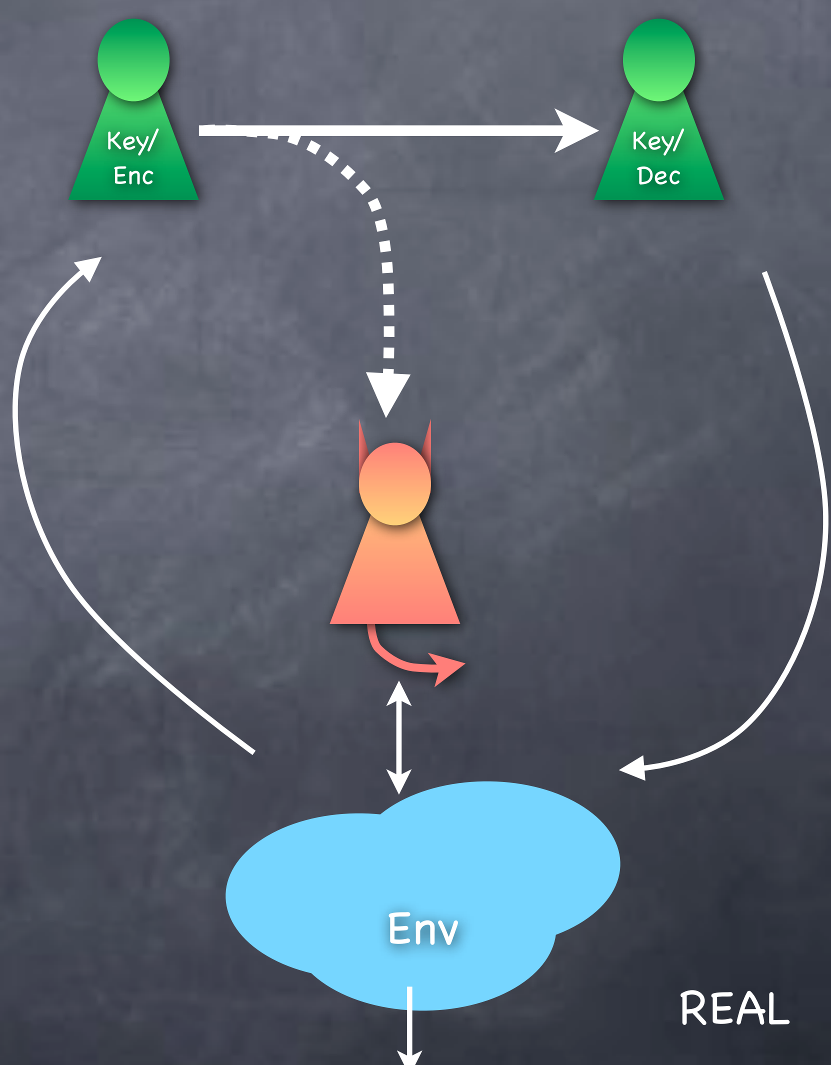
Consider this simulator: Runs adversary internally and lets it talk to the environment directly!



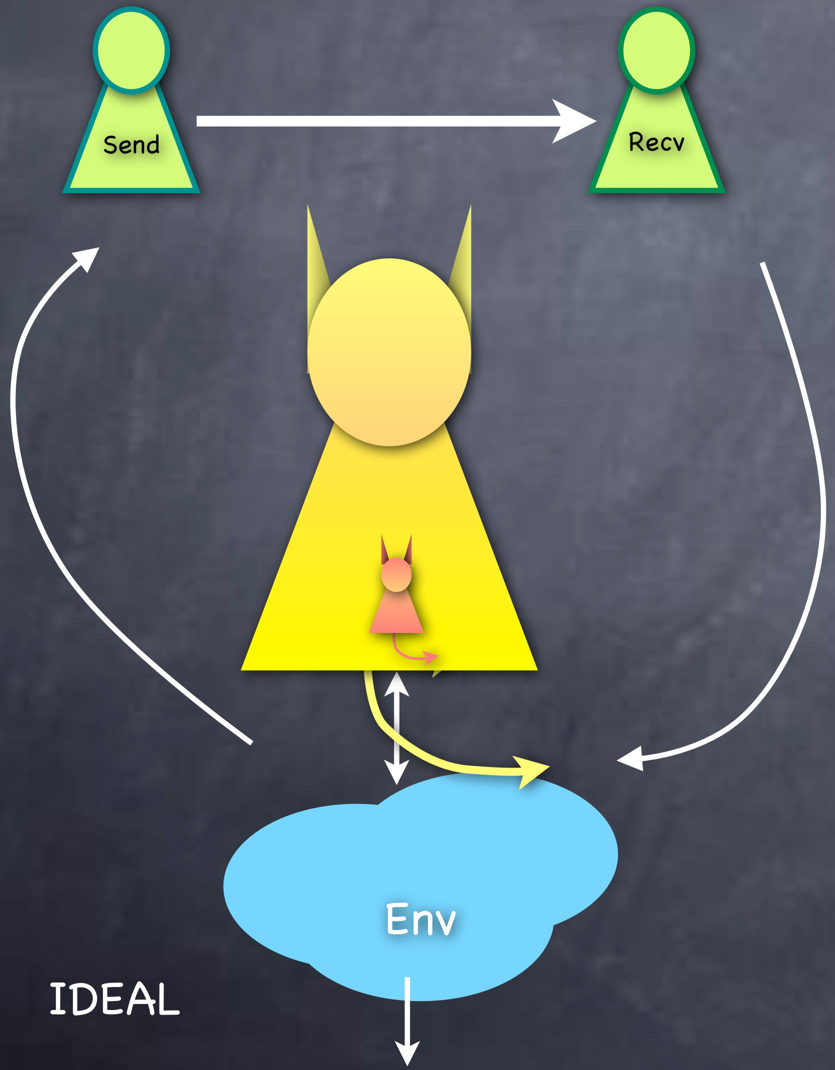
Perfect Secrecy + Correctness ⇒ SIM-Onetime Security



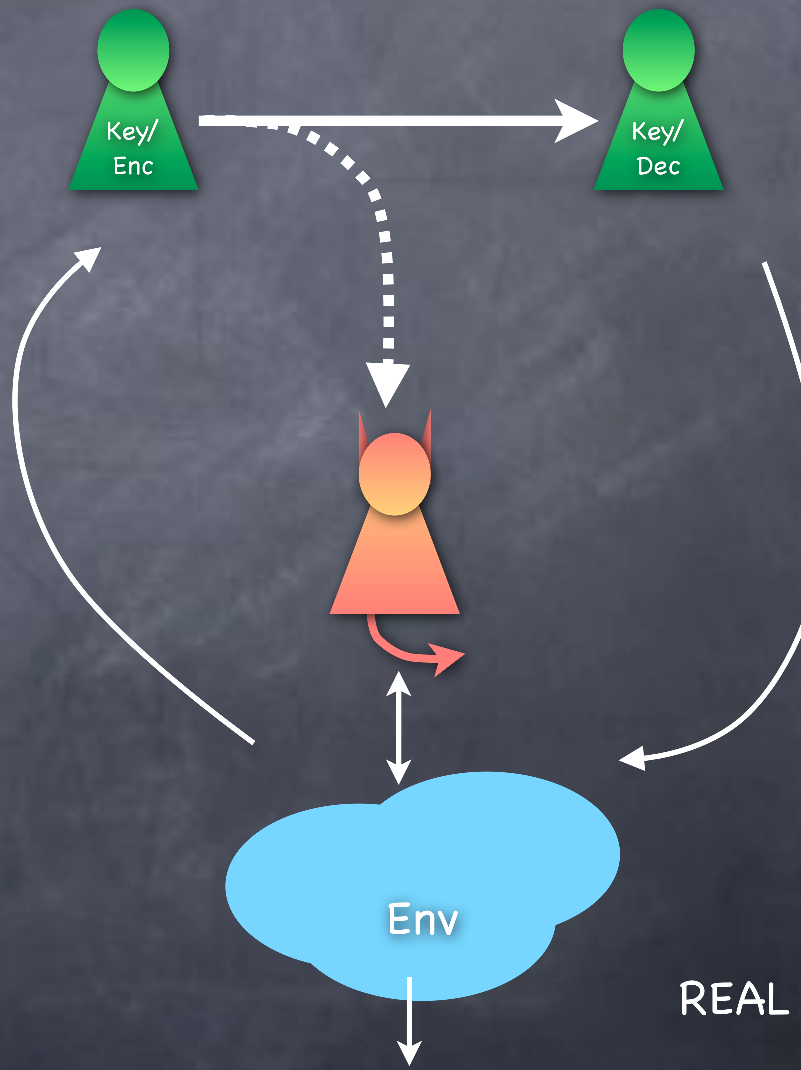
Consider this simulator: Runs adversary internally and lets it talk to the environment directly!



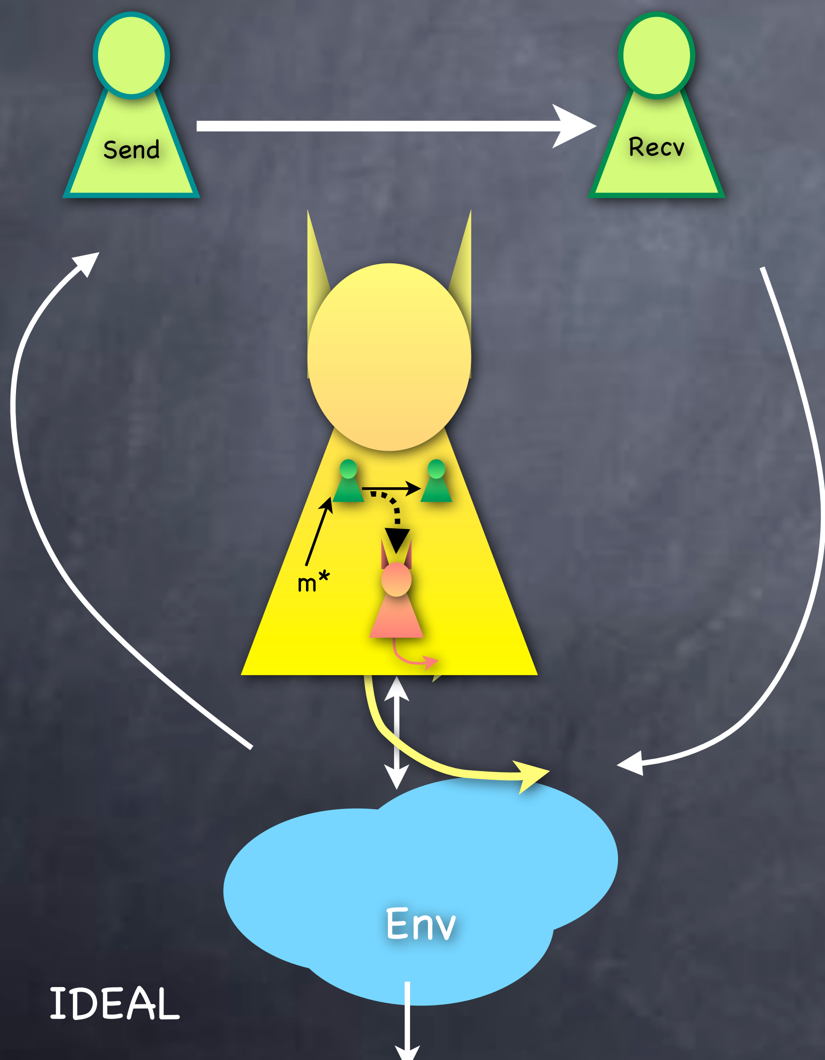
Perfect Secrecy + Correctness ⇒ SIM-Onetime Security



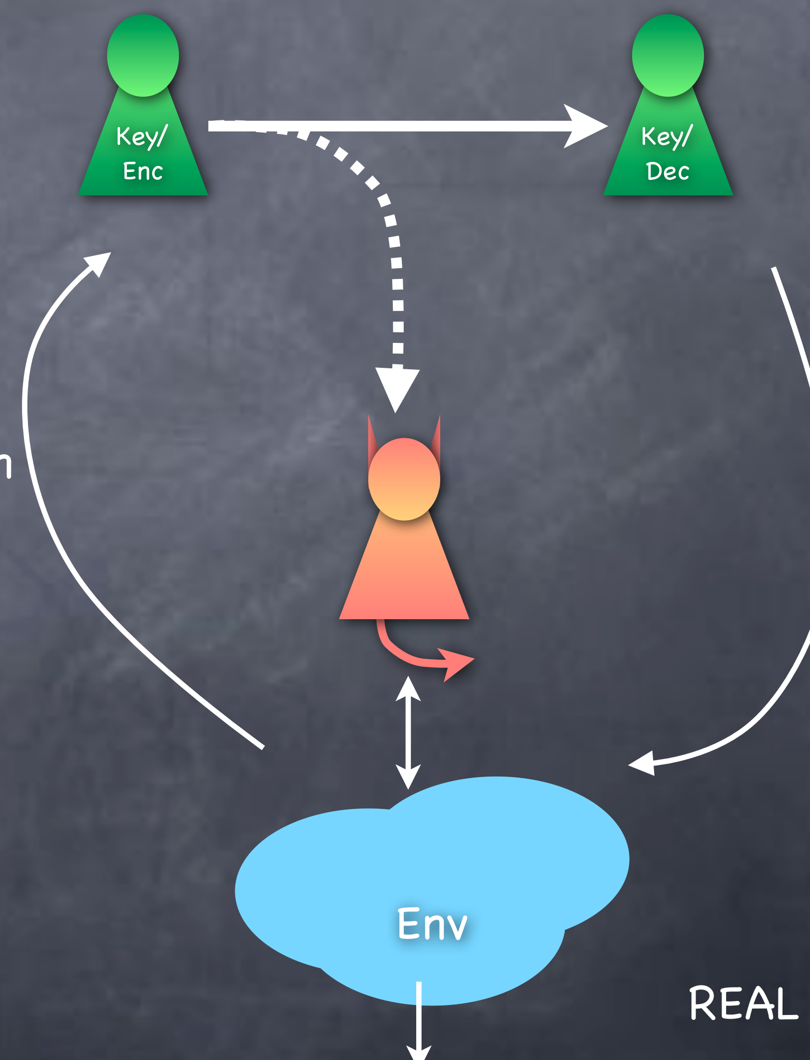
Consider this simulator: Runs adversary internally and lets it talk to the environment directly!



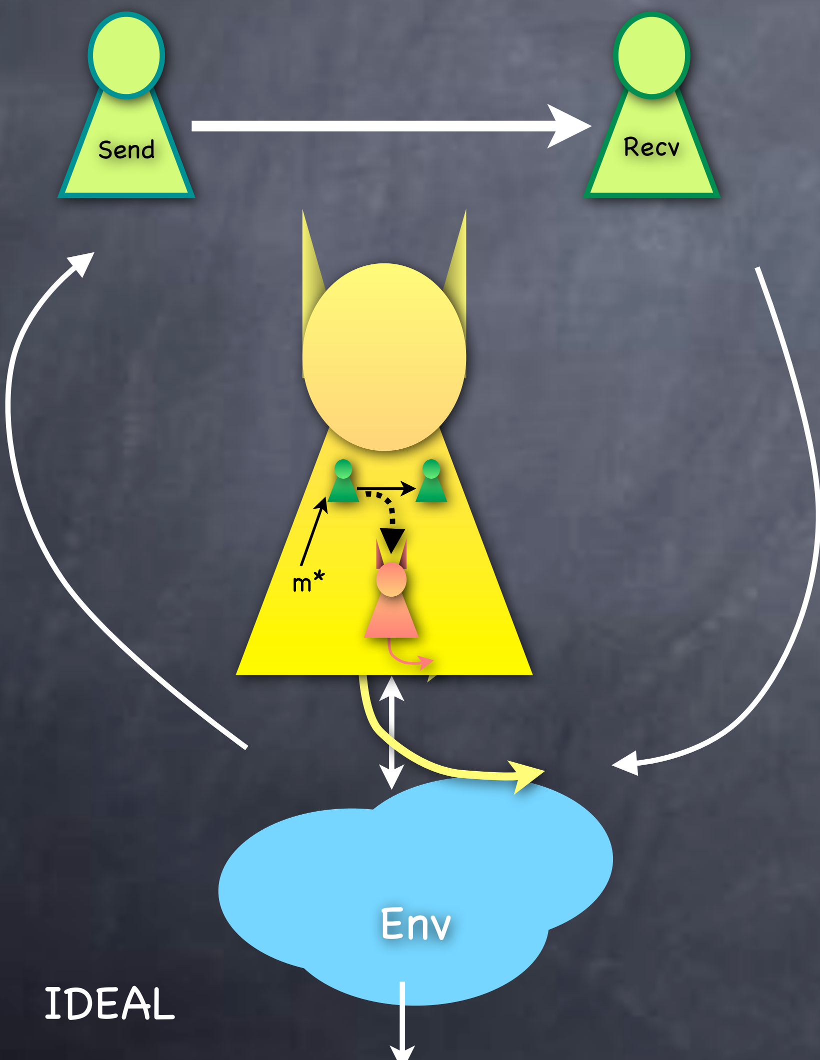
Perfect Secrecy + Correctness ⇒ SIM-Onetime Security



Consider this simulator: Runs adversary internally and lets it talk to the environment directly!
Feeds it encryption of a dummy message



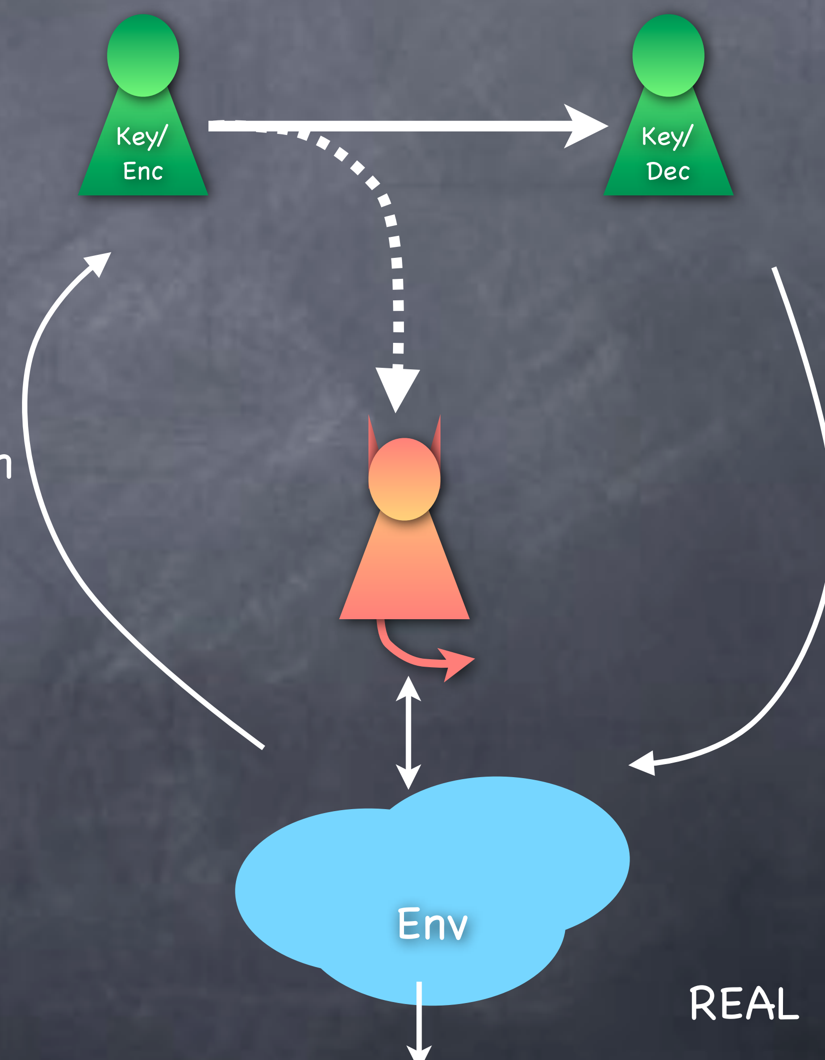
Perfect Secrecy + Correctness ⇒ SIM-Onetime Security



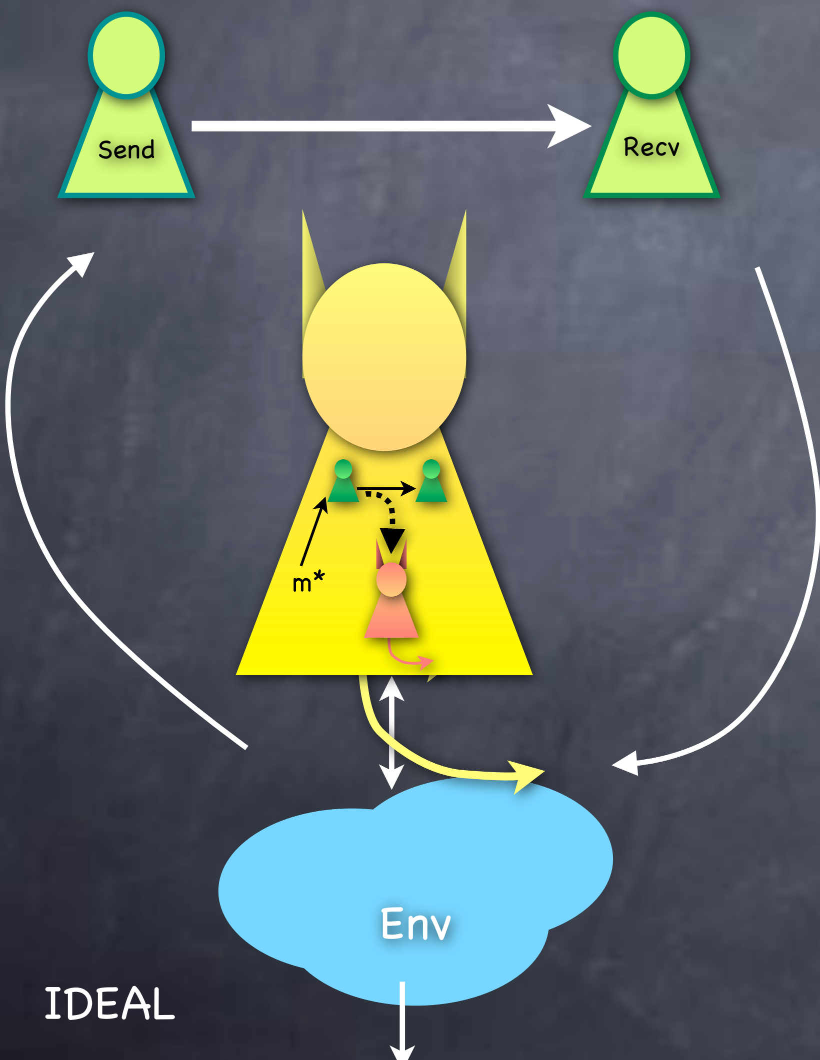
Consider this simulator: Runs adversary internally and lets it talk to the environment directly!

Feeds it encryption of a dummy message

Can show that REAL=IDEAL

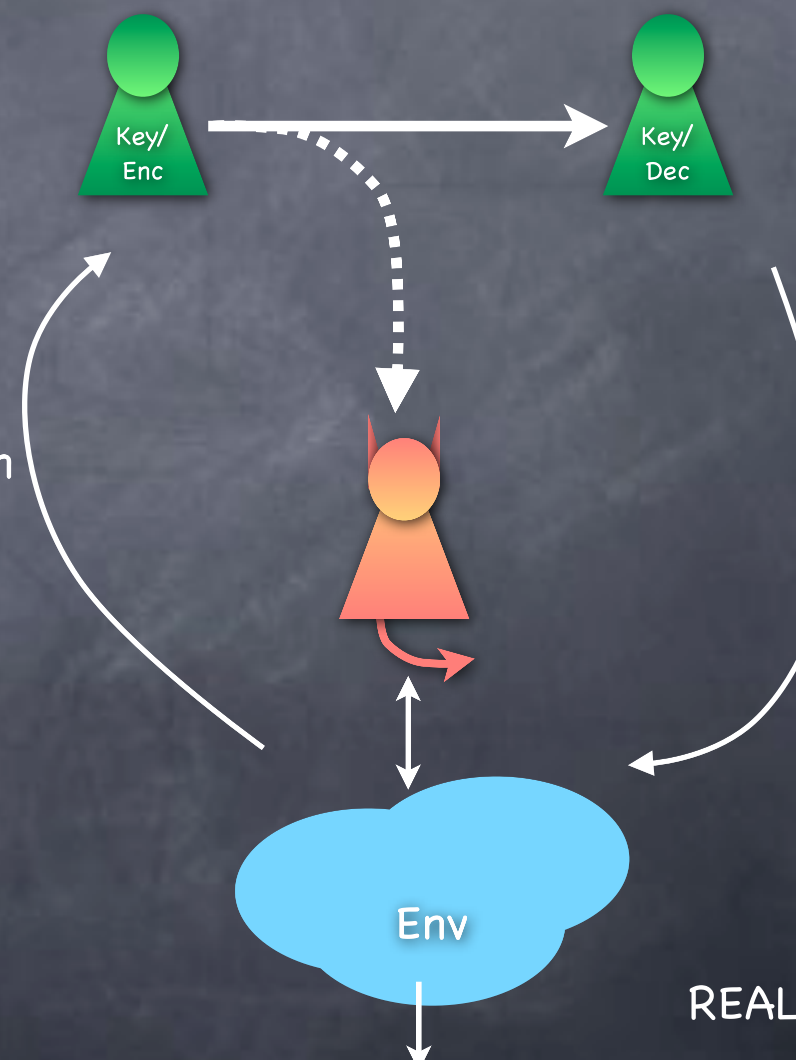


Perfect Secrecy + Correctness ⇒ SIM-Onetime Security



Consider this simulator: Runs adversary internally and lets it talk to the environment directly!
Feeds it encryption of a dummy message

Can show that REAL=IDEAL
(Consider view of + for both)



Implicit Details

Implicit Details

- Random coins used by the encryption scheme is kept private within the programs of the scheme (KeyGen, Enc, Dec)

Implicit Details

- Random coins used by the encryption scheme is kept private within the programs of the scheme (KeyGen, Enc, Dec)
  - If key is used for anything else (i.e., leaked to the environment) no more guarantees

Implicit Details

- Random coins used by the encryption scheme is kept private within the programs of the scheme (KeyGen, Enc, Dec)
  - If key is used for anything else (i.e., leaked to the environment) no more guarantees
- In REAL, Eve+Env's only inputs are ciphertext and Bob's output

Implicit Details

- Random coins used by the encryption scheme is kept private within the programs of the scheme (KeyGen, Enc, Dec)
  - If key is used for anything else (i.e., leaked to the environment) no more guarantees
- In REAL, Eve+Env's only inputs are ciphertext and Bob's output
  - In particular no timing attacks

Implicit Details

- Random coins used by the encryption scheme is kept private within the programs of the scheme (KeyGen, Enc, Dec)
    - If key is used for anything else (i.e., leaked to the environment) no more guarantees
- In REAL, Eve+Env's only inputs are ciphertext and Bob's output
    - In particular no timing attacks
- Message space is finite and known to Eve (and Eve')

Implicit Details

- Random coins used by the encryption scheme is kept private within the programs of the scheme (KeyGen, Enc, Dec)
  - If key is used for anything else (i.e., leaked to the environment) no more guarantees
- In REAL, Eve+Env's only inputs are ciphertext and Bob's output
  - In particular no timing attacks
- Message space is finite and known to Eve (and Eve')
  - Alternately, if message length is variable, it is given out to Eve' in IDEAL as well

Implicit Details

- Random coins used by the encryption scheme is kept private within the programs of the scheme (KeyGen, Enc, Dec)
  - If key is used for anything else (i.e., leaked to the environment) no more guarantees
- In REAL, Eve+Env's only inputs are ciphertext and Bob's output
  - In particular no timing attacks
- Message space is finite and known to Eve (and Eve')
  - Alternately, if message length is variable, it is given out to Eve' in IDEAL as well
  - Also, Eve' allowed to learn the fact that a message is sent

Onetime Encryption

IND-Onetime Security

Onetime Encryption

IND-Onetime Security

- IND-Onetime Experiment

Onetime Encryption

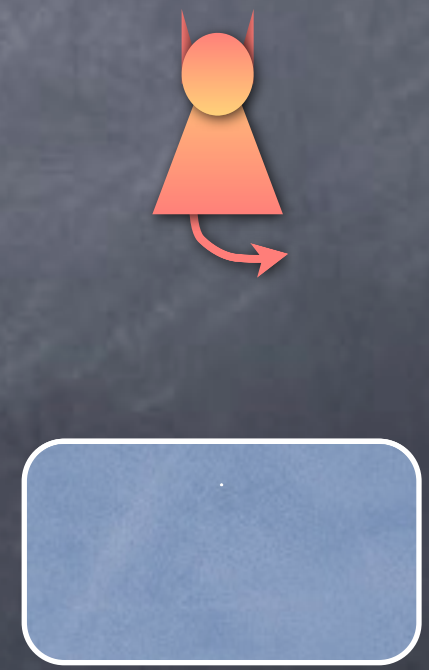
IND-Onetime Security

- IND-Onetime Experiment

Onetime Encryption

IND-Onetime Security

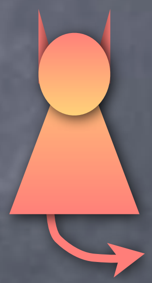
- IND-Onetime Experiment

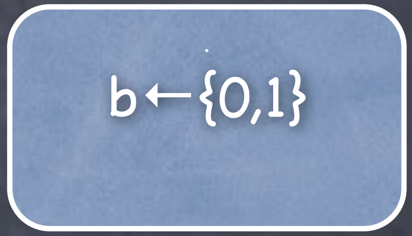


Onetime Encryption

IND-Onetime Security

- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K





Onetime Encryption

IND-Onetime Security

- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K

Onetime Encryption

IND-Onetime Security

- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K
  - Adversary sends two messages m_0, m_1 to the experiment

Onetime Encryption

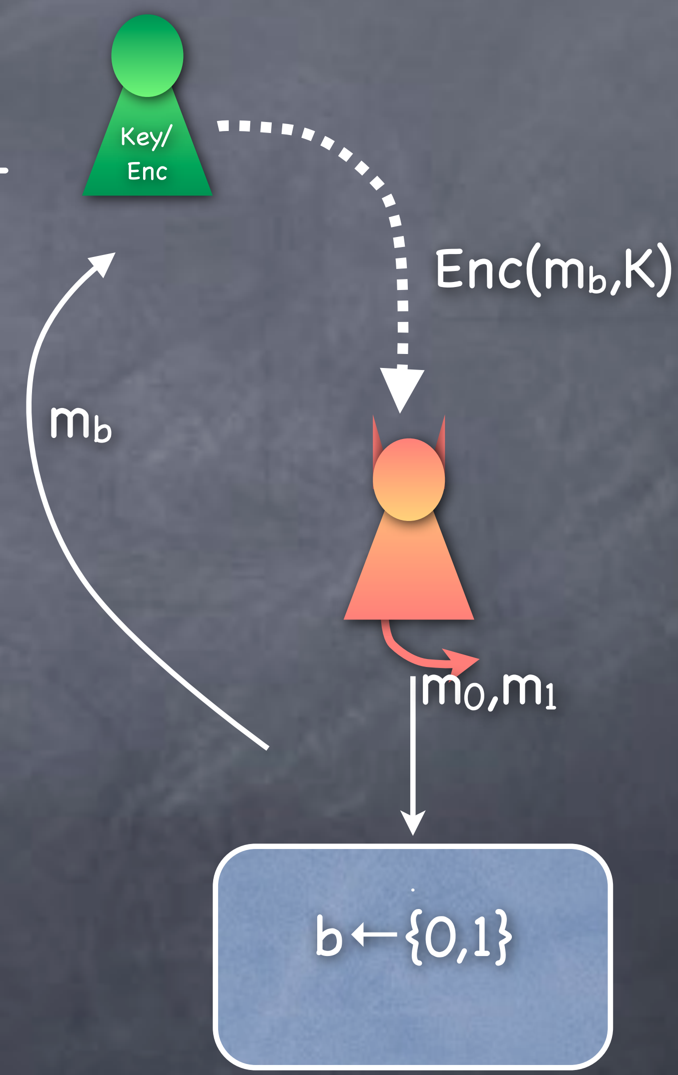
IND-Onetime Security

- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K
  - Adversary sends two messages m_0, m_1 to the experiment
  - Experiment replies with Enc(m_b,K)

Onetime Encryption

IND-Onetime Security

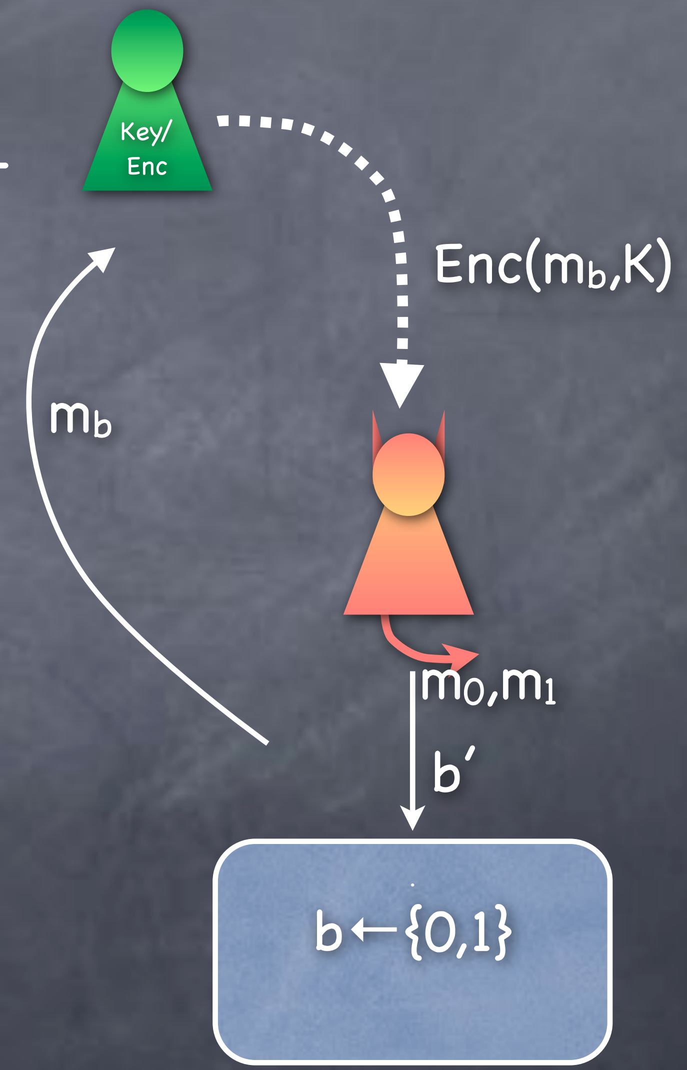
- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K
  - Adversary sends two messages m_0, m_1 to the experiment
  - Experiment replies with Enc(m_b,K)



Onetime Encryption

IND-Onetime Security

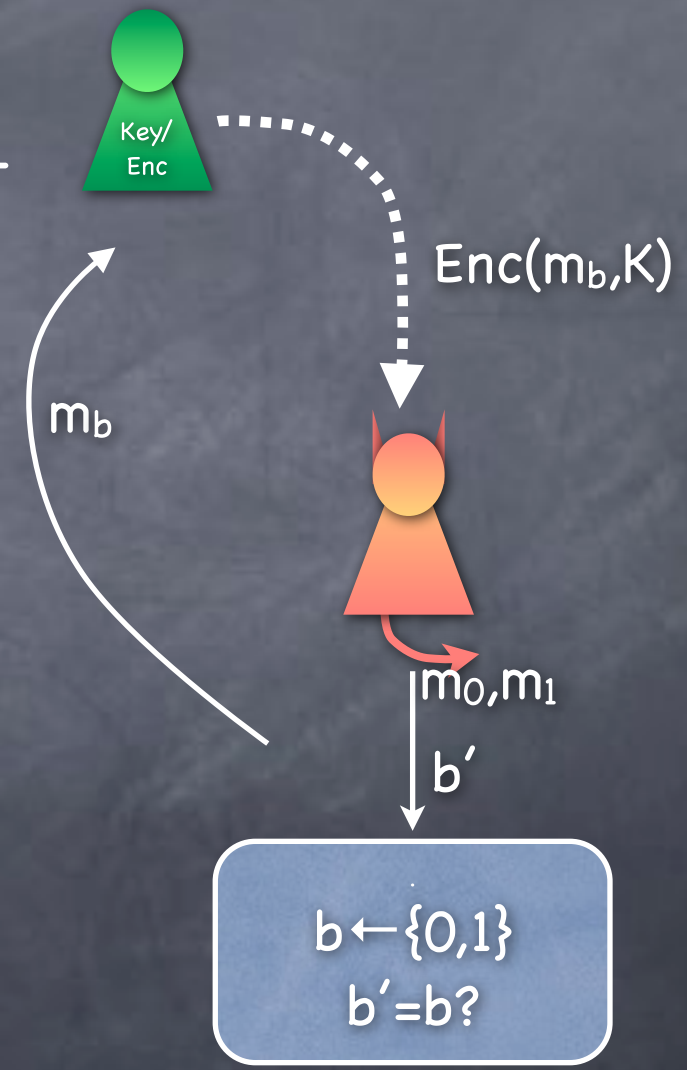
- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K
  - Adversary sends two messages m_0, m_1 to the experiment
  - Experiment replies with Enc(m_b,K)
  - Adversary returns a guess b'



Onetime Encryption

IND-Onetime Security

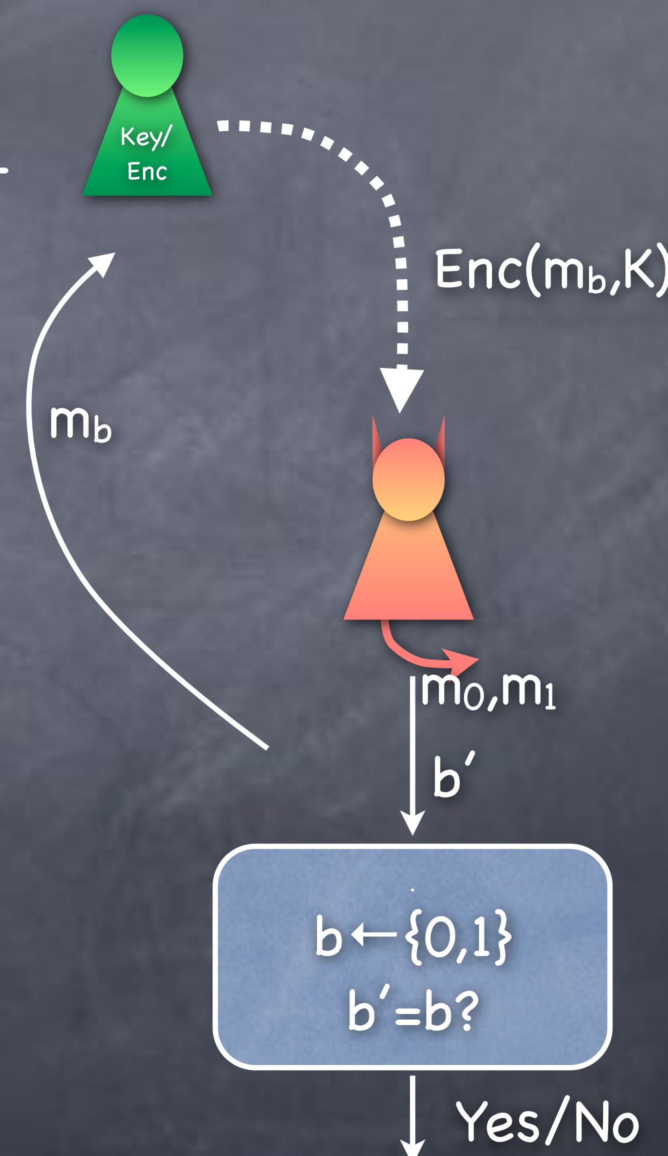
- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K
  - Adversary sends two messages m_0, m_1 to the experiment
  - Experiment replies with Enc(m_b,K)
  - Adversary returns a guess b′



Onetime Encryption

IND-Onetime Security

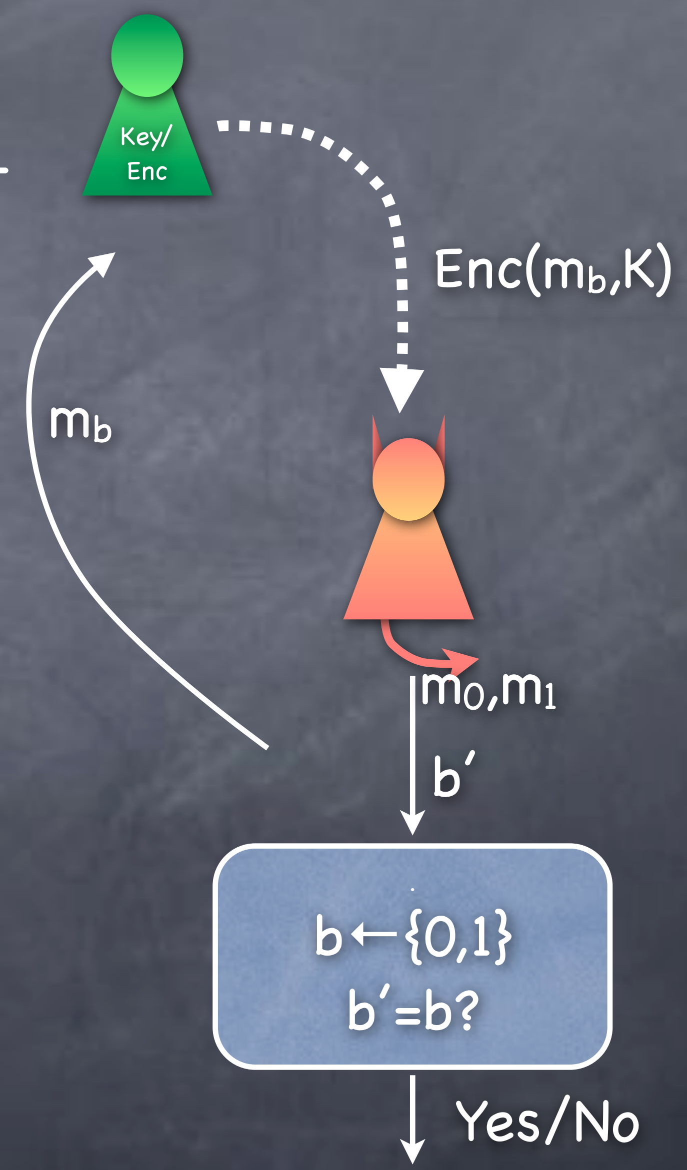
- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K
  - Adversary sends two messages m_0, m_1 to the experiment
  - Experiment replies with Enc(m_b,K)
  - Adversary returns a guess b'
  - Experiments outputs 1 iff b'=b



Onetime Encryption

IND-Onetime Security

- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K
  - Adversary sends two messages m_0, m_1 to the experiment
  - Experiment replies with Enc(m_b,K)
  - Adversary returns a guess b'
  - Experiments outputs 1 iff b'=b
- IND-Onetime secure if for every adversary, Pr[b'=b] = 1/2

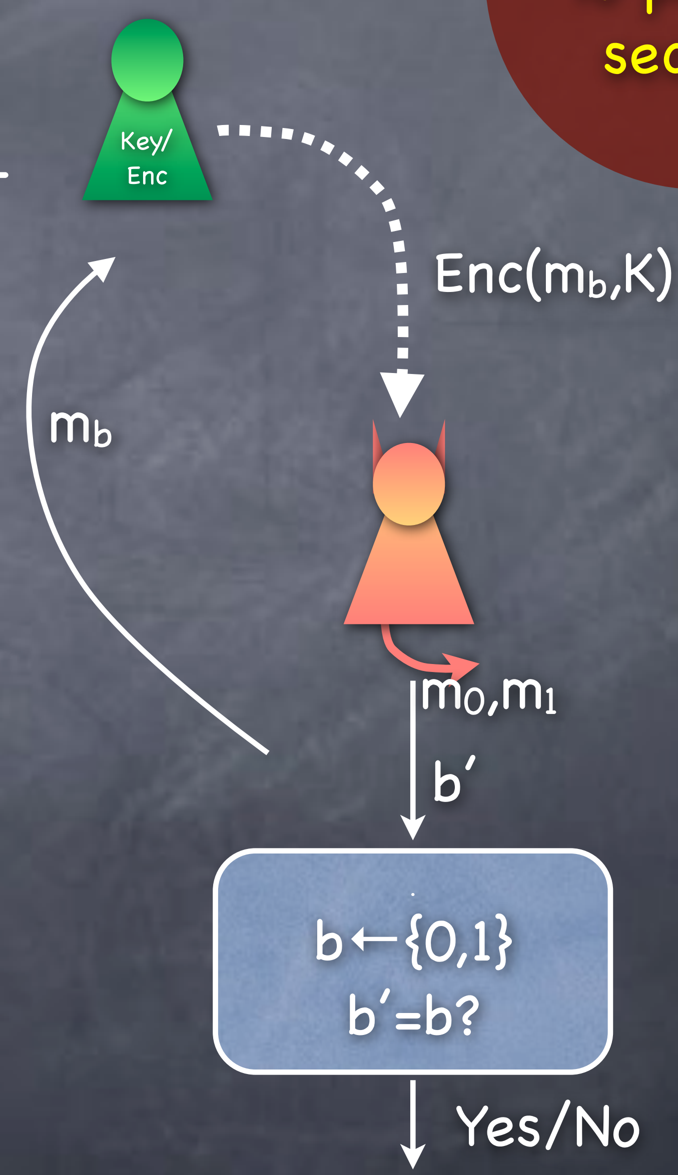


Onetime Encryption

IND-Onetime Security

Equivalent to perfect secrecy

- IND-Onetime Experiment
  - Experiment picks a random bit b. It also runs KeyGen to get a key K
  - Adversary sends two messages m_0, m_1 to the experiment
  - Experiment replies with Enc(m_b,K)
  - Adversary returns a guess b′
  - Experiments outputs 1 iff b′=b
- IND-Onetime secure if for every adversary, Pr[b′=b] = 1/2



Perspective on Definitions

Perspective on Definitions

- "Technical" vs. "Convincing"

Perspective on Definitions

- "Technical" vs. "Convincing"
- For simple scenarios technical definitions could be convincing

Perspective on Definitions

- “Technical” vs. “Convincing”
- For simple scenarios technical definitions could be convincing
  - e.g. Perfect Secrecy

Perspective on Definitions

- "Technical" vs. "Convincing"
- For simple scenarios technical definitions could be convincing
    - e.g. Perfect Secrecy
- IND- definitions tend to be technical: more low-level details, but may not make the big picture clear. Could have "weaknesses"

Perspective on Definitions

- "Technical" vs. "Convincing"
- For simple scenarios technical definitions could be convincing
  - e.g. Perfect Secrecy
- IND- definitions tend to be technical: more low-level details, but may not make the big picture clear. Could have "weaknesses"
- SIM- definitions give the big picture, but may not give details of what is involved in satisfying it. Could be "too strong"

Perspective on Definitions

- "Technical" vs. "Convincing"
- For simple scenarios technical definitions could be convincing
  - e.g. Perfect Secrecy
- IND- definitions tend to be technical: more low-level details, but may not make the big picture clear. Could have "weaknesses"
- SIM- definitions give the big picture, but may not give details of what is involved in satisfying it. Could be "too strong"
- Best of both worlds when they are equivalent:
  use IND- definition while say, proving security of a construction;
  use SIM- definition when low-level details are not important