Defining Encryption

Lecture 2
Defining Encryption

Lecture 2

Towards Defining Secrecy against the 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”
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

Eve's Program

Alice's Program

Bob's Program
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?
Where does the message come from?

Eve might already have partial information about the message, or might receive such information later.
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.
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 includes the operating systems and other programs run by the participants, as well as other parties, if in a network.
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.
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:
A scheme is secure (and correct) if:

\[ \forall \]

**Defining Security**

The REAL/IDEAL Paradigm
A scheme is secure (and correct) if:

\[ \forall \exists \ s.t. \]

The REAL/IDEAL Paradigm

Defining Security
A scheme is secure (and correct) if:

\[ \forall \exists \text{s.t.} \forall \text{Key/Enc} \subseteq \text{Key/Dec} \]

The REAL/IDEAL Paradigm

Defining Security
A scheme is secure (and correct) if:

\[ \forall \exists \text{s.t.} \forall \text{output of is distributed identically in REAL and IDEAL} \]
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.
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.
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”
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 (multi-message) encryption
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”
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”

We will also see alternate (but essentially equivalent) security definitions.
Onetime Encryption
Onetime Encryption

The Syntax

- **Shared-key (Private-key) Encryption**

- **Key Generation**: Randomized
  
  \[ K \leftarrow \mathcal{K}, \text{ uniformly randomly drawn from the key-space} \]
  
  (or according to a key-distribution)

- **Encryption**: Deterministic
  
  \[ \text{Enc}: \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C} \]

- **Decryption**: Deterministic
  
  \[ \text{Dec}: \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M} \]
Onetime Encryption

Perfect Secrecy
Perfect Secrecy

Onetime Encryption

Perfect secrecy: \( \forall m, m' \in M \)

\( \{\text{Enc}(m,K)\}_{K \leftarrow \text{KeyGen}} = \{\text{Enc}(m',K)\}_{K \leftarrow \text{KeyGen}} \)
Perfect Secrecy

Onetime Encryption

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
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
Onetime Encryption
Perfect Secrecy

<table>
<thead>
<tr>
<th></th>
<th>K</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>a</td>
<td>x</td>
<td>y</td>
<td>y</td>
<td>z</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>y</td>
<td>x</td>
<td>z</td>
<td>y</td>
</tr>
</tbody>
</table>

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.
Perfect Secrecy

Onetime Encryption

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

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.

Assuming \( K \) uniformly drawn from \( \mathcal{K} \):

\[
\begin{align*}
\Pr[\text{Enc}(a,K)=x] &= \frac{1}{4}, \\
\Pr[\text{Enc}(a,K)=y] &= \frac{1}{2}, \\
\Pr[\text{Enc}(a,K)=z] &= \frac{1}{4}, \\
\Pr[\text{Enc}(a,K)=z] &= \frac{1}{4}
\end{align*}
\]

Same for \( \text{Enc}(b,K) \).
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.

In addition, require correctness:

\( \forall m, K, \text{Dec( Enc(m,K), K) = m} \)

Assuming K uniformly drawn from \( \mathcal{K} \):

\[
\begin{align*}
\text{Pr}[\text{Enc}(a,K)=x] &= \frac{1}{4}, \\
\text{Pr}[\text{Enc}(a,K)=y] &= \frac{1}{2}, \\
\text{Pr}[\text{Enc}(a,K)=z] &= \frac{1}{4}
\end{align*}
\]

Same for \( \text{Enc}(b,K) \).
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

In addition, require correctness

\( \forall m, K, \quad \text{Dec}(\text{Enc}(m,K), K) = m \)

E.g. **One-time pad:** \( \mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^n \) and

\[ \text{Enc}(m,K) = m \oplus K, \quad \text{Dec}(c,K) = c \oplus K \]

Assuming \( K \) uniformly drawn from \( \mathcal{K} \)

\[
\begin{array}{cccc}
\mathcal{M} & 0 & 1 & 2 & 3 \\
\hline
a & x & y & y & y \\
b & y & x & z & y \\
\end{array}
\]

- \( \Pr[\text{Enc}(a,K)=x] = \frac{1}{4} \)
- \( \Pr[\text{Enc}(a,K)=y] = \frac{1}{2} \)
- \( \Pr[\text{Enc}(a,K)=z] = \frac{1}{4} \)

Same for \( \text{Enc}(b,K) \).
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

- In addition, require correctness
  \( \forall m, K, \quad \text{Dec}(\text{Enc}(m,K), K) = m \)

- E.g. One-time pad: \( \mathcal{M} = \mathcal{K} = \mathcal{C} = \{0,1\}^n \) and \( \text{Enc}(m,K) = m \oplus K, \text{Dec}(c,K) = c \oplus K \)

- More generally \( \mathcal{M} = \mathcal{K} = \mathcal{C} = \mathcal{G} \) (a finite group) and \( \text{Enc}(m,K) = m + K, \text{Dec}(c,K) = c - K \)

**Table:**

<table>
<thead>
<tr>
<th>( \mathcal{M} )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>x</td>
<td>y</td>
<td>y</td>
<td>z</td>
</tr>
<tr>
<td>b</td>
<td>y</td>
<td>x</td>
<td>z</td>
<td>y</td>
</tr>
</tbody>
</table>

Assuming \( K \) uniformly drawn from \( \mathcal{K} \):

\[
\begin{align*}
\Pr[\text{Enc}(a,K)=x] &= \frac{1}{4}, \\
\Pr[\text{Enc}(a,K)=y] &= \frac{1}{2}, \\
\Pr[\text{Enc}(a,K)=z] &= \frac{1}{4}
\end{align*}
\]

Same for \( \text{Enc}(b,K) \).
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

\[ \forall \exists \text{s.t.} \forall \text{Key/Enc} \text{Send} \rightarrow \text{Recv} \]

REAL = IDEAL

SIM-Onetime secure if:

\[ \forall \exists \text{s.t.} \forall \text{REAL} = \text{IDEAL} \]
Onetime Encryption

SIM-Onetime Security

Class of environments which send only one message.

\[
\forall \exists \text{s.t.} \quad \forall \quad \text{Key/Enc} \quad \text{Key/Dec}
\]

REAL=IDEAL

SIM-Onetime secure if:

\[
\forall \quad \exists \quad \text{s.t.} \quad \forall
\]

REAL=IDEAL

Equivalent to perfect secrecy + correctness
Perfect Secrecy + Correctness $\Rightarrow$ SIM-Onetime Security
Consider this simulator: Runs adversary internally and lets it talk to the environment directly!

Perfect Secrecy + Correctness $\Rightarrow$ SIM-Onetime Security
Perfect Secrecy + Correctness $\implies$ SIM-Onetime Security

Consider this simulator: Runs adversary internally and lets it talk to the environment directly!
Perfect Secrecy + Correctness $\Rightarrow$ SIM-Onetime Security

Consider this simulator: Runs adversary internally and lets it talk to the environment directly!
Consider this simulator: Runs adversary internally and lets it talk to the environment directly! Feeds it encryption of a dummy message.
Perfect Secrecy + Correctness $\Rightarrow$ 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 $\text{REAL} = \text{IDEAL}$.
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 $\text{Enc}(m_b, K)$.
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 $\text{Enc}(m_b, K)$
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 $\text{Enc}(m_b, K)$
  - Adversary returns a guess $b'$

$\{0, 1\}$
**Onetime Encryption**

**IND-Onetime Security**

- **IND-Onetime Experiment**
  - The experiment picks a random bit $b$. It also runs KeyGen to get a key $K$.
  - The adversary sends two messages $m_0$, $m_1$ to the experiment.
  - The experiment replies with $\text{Enc}(m_b, K)$.
  - The adversary returns a guess $b'$.

$\text{b} \leftarrow \{0, 1\}$

Does $b' = b$?
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 $\text{Enc}(m_b, K)$
- 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 $\text{Enc}(m_b, K)$
- 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

- **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

“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