

Defining Encryption (ctd.)

Lecture 3
SIM & IND security
Beyond One-Time: CPA security
Computational Indistinguishability

Recall

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**

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

• 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 $\text{Enc}(m, K) = m + K, \text{Dec}(c, K) = c - K$

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

Assuming K uniformly drawn from \mathcal{K}

$$\Pr[\text{Enc}(a, K) = x] = \frac{1}{4},$$

$$\Pr[\text{Enc}(a, K) = y] = \frac{1}{2},$$

$$\Pr[\text{Enc}(a, K) = z] = \frac{1}{4}$$

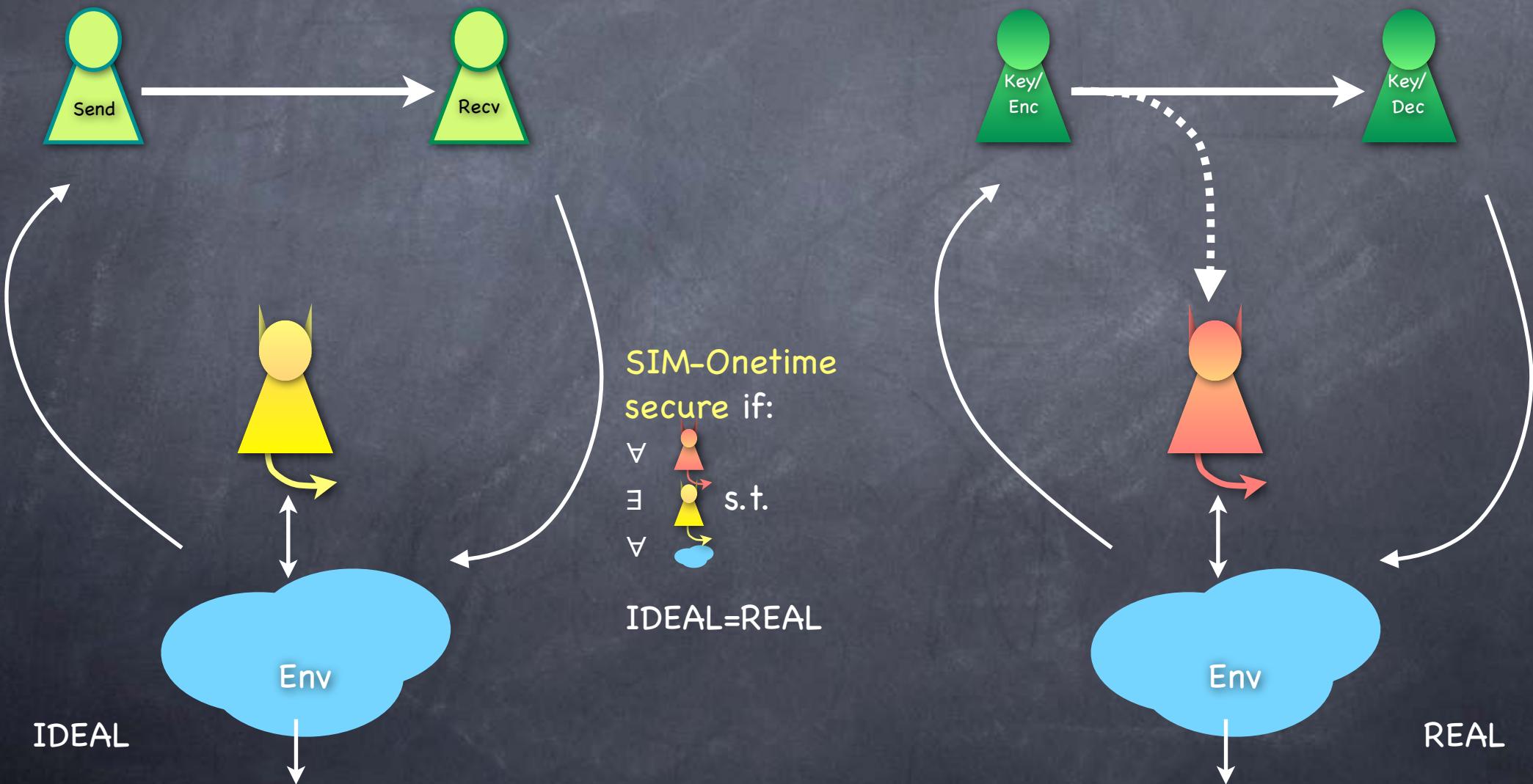
Same for $\text{Enc}(b, K)$.

Recall

Onetime Encryption

SIM-Onetime Security

- Class of environments which send only one message



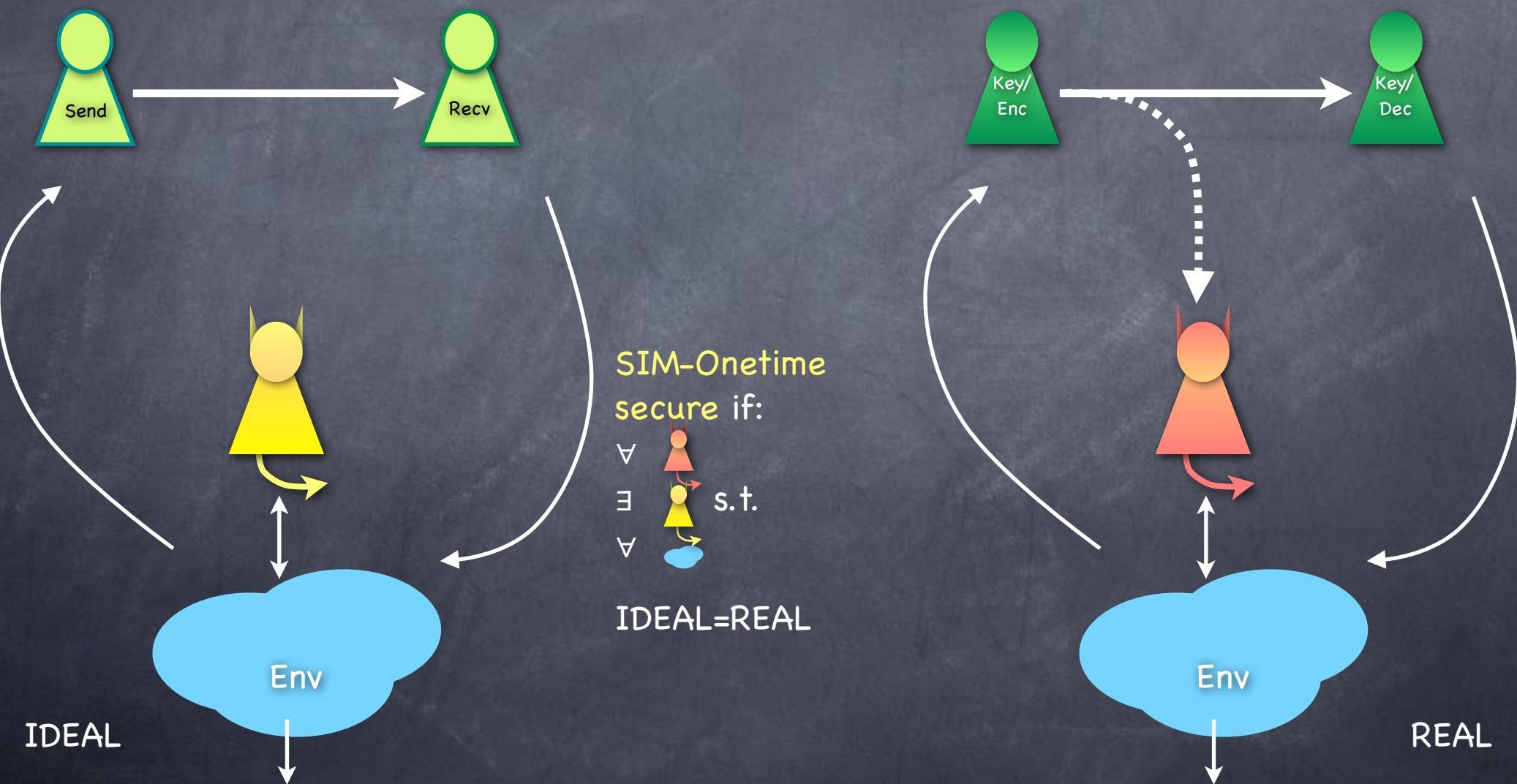
Recall

Onetime Encryption

SIM-Onetime Security

Equivalent to
perfect secrecy
+ correctness

- Class of environments which send only one message



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



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

Onetime Encryption

IND-Onetime Security

- IND-Onetime Experiment

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



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

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



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

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 $\text{Enc}(m_b, K)$



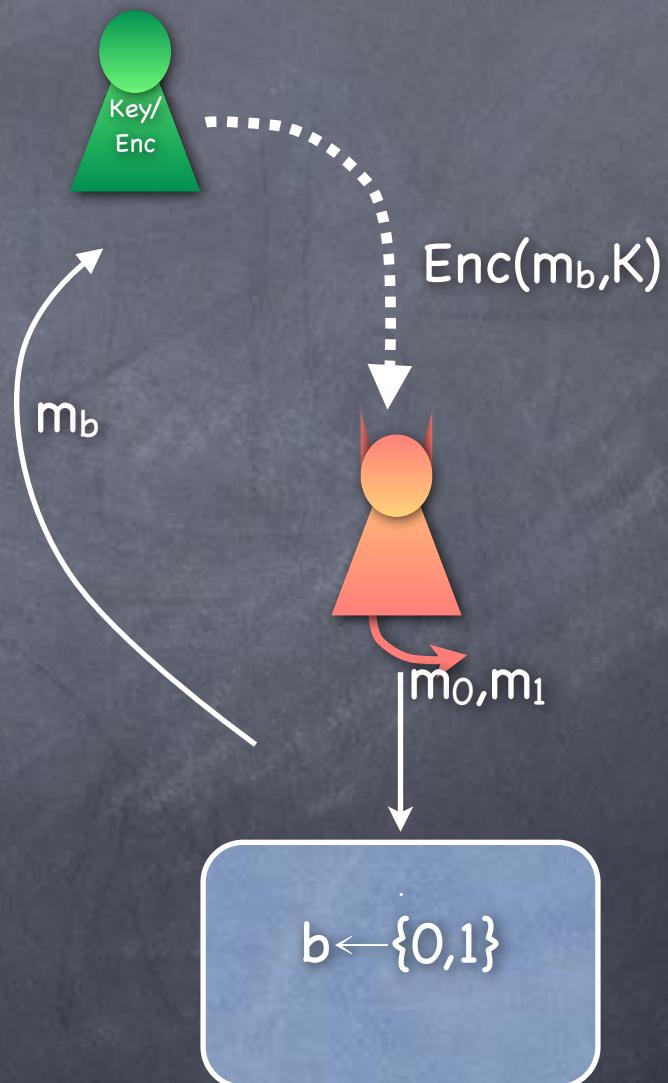
$b \leftarrow \{0,1\}$

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 $\text{Enc}(m_b, K)$

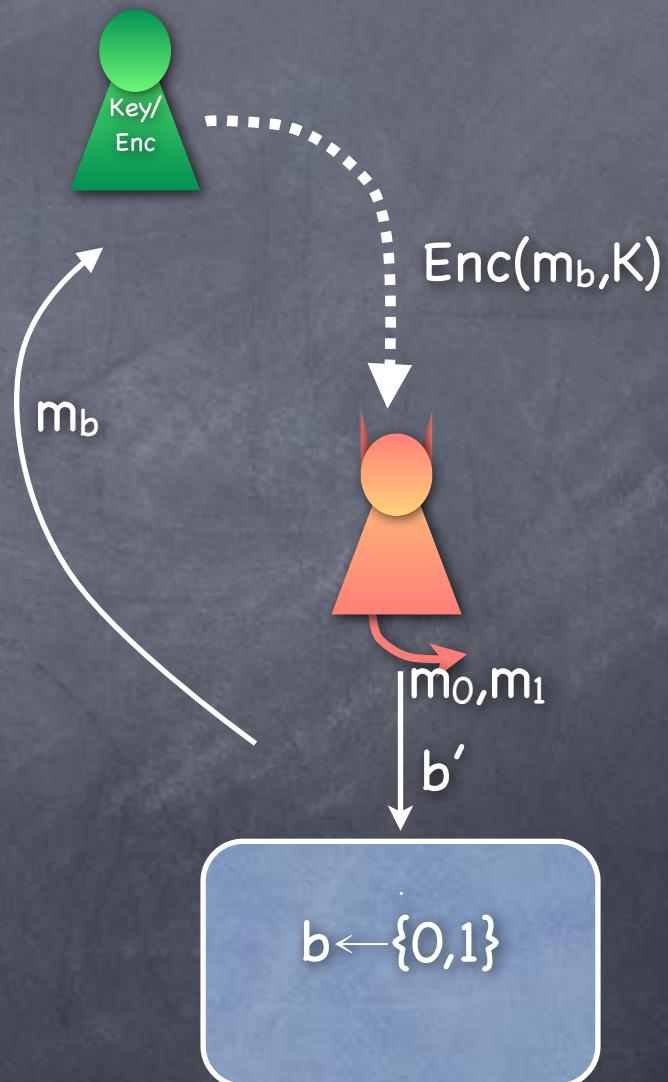


Onetime Encryption

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'

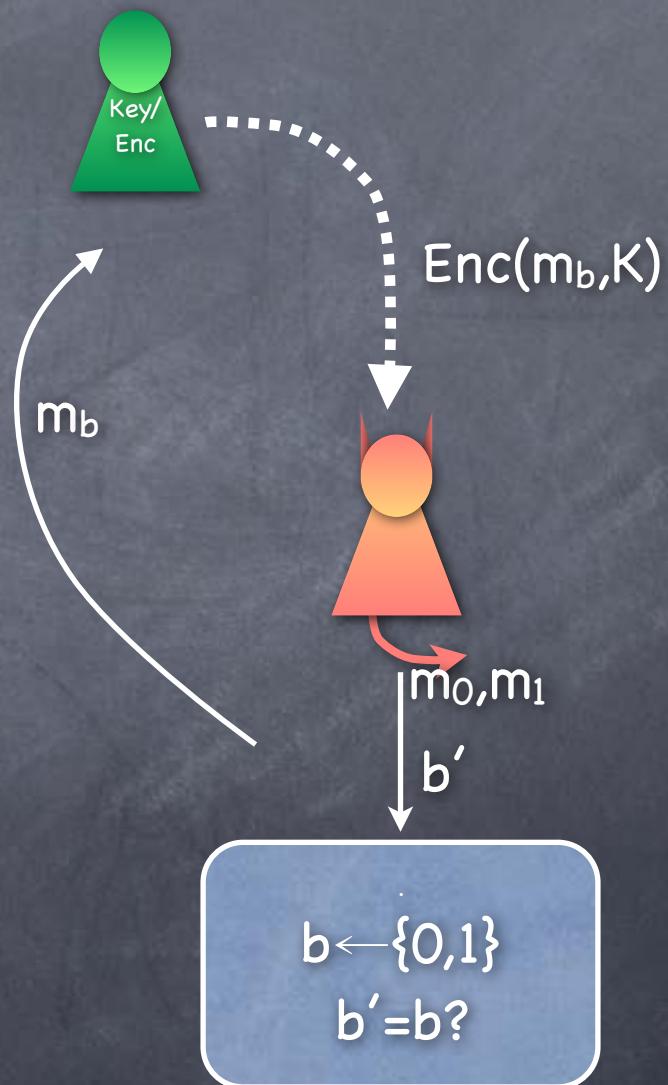


Onetime Encryption

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'

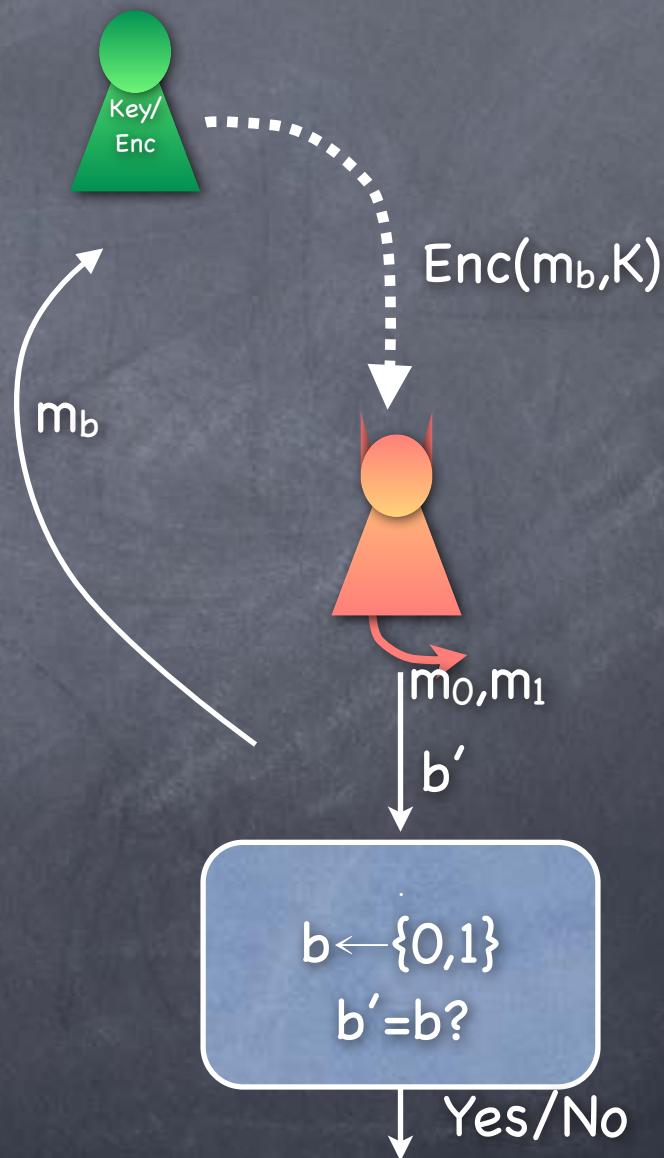


Onetime Encryption

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 $b' = b$

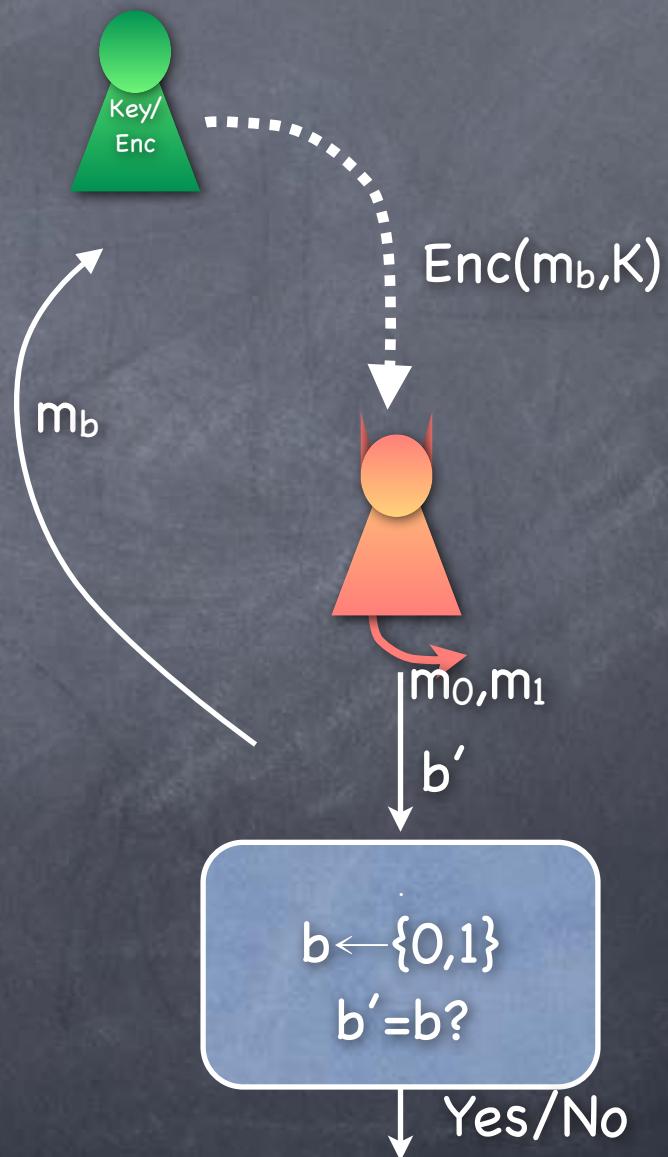


Onetime Encryption

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, $\Pr[b' = b] = 1/2$

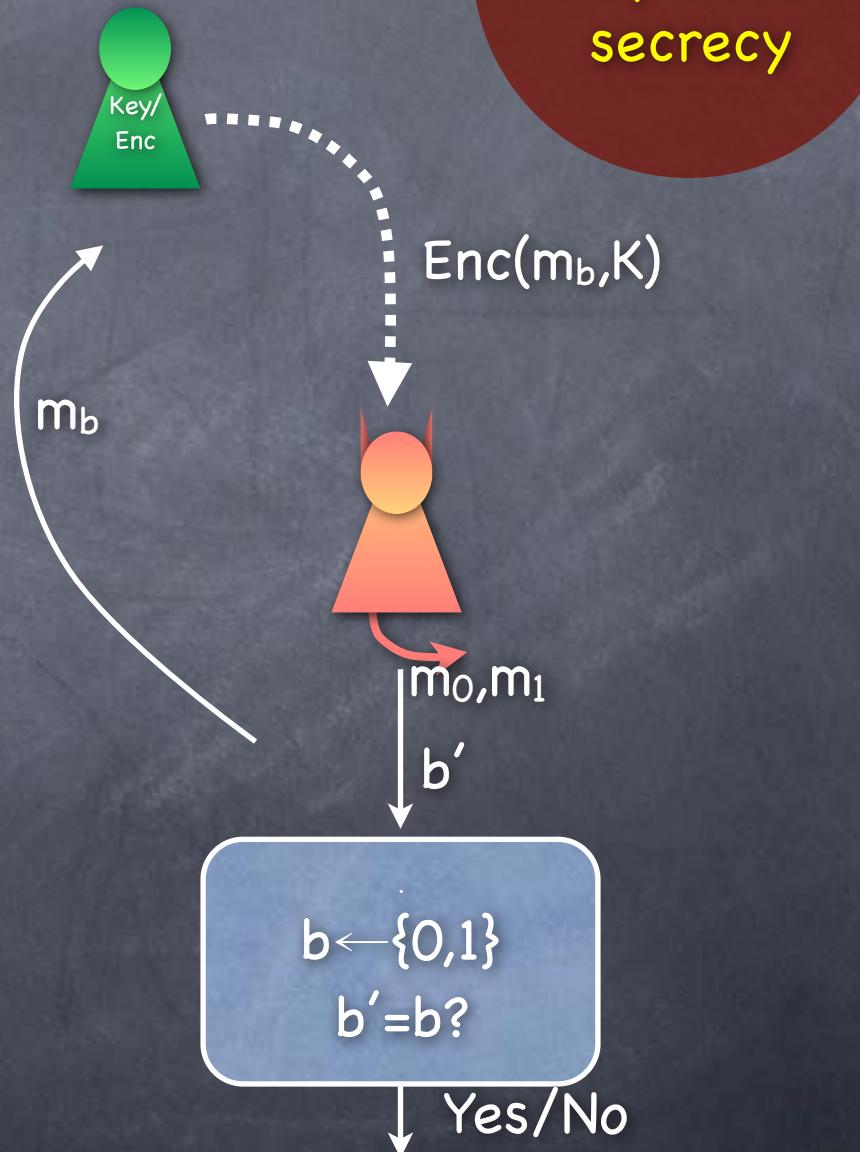


Onetime Encryption

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 $\text{Enc}(m_b, K)$
- Adversary returns a guess b'
- Experiments outputs 1 iff $b' = b$
- IND-Onetime secure if for every adversary, $\Pr[b' = b] = 1/2$



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

Security of Encryption

Security of Encryption

- Perfect secrecy is too strong for multiple messages (though too weak in some other respects...)

Security of Encryption

- Perfect secrecy is too strong for multiple messages (though too weak in some other respects...)
- Requires keys as long as the messages

Security of Encryption

- Perfect secrecy is too strong for multiple messages (though too weak in some other respects...)
 - Requires keys as long as the messages
- Relax the requirement by restricting to **computationally bounded adversaries** (and environments)

Security of Encryption

- Perfect secrecy is too strong for multiple messages (though too weak in some other respects...)
 - Requires keys as long as the messages
- Relax the requirement by restricting to **computationally bounded adversaries** (and environments)
- Coming up: Formalizing notions of “computational” security (as opposed to perfect/statistical security)

Security of Encryption

- Perfect secrecy is too strong for multiple messages (though too weak in some other respects...)
 - Requires keys as long as the messages
- Relax the requirement by restricting to **computationally bounded adversaries** (and environments)
- Coming up: Formalizing notions of “computational” security (as opposed to perfect/statistical security)
 - Then, security definitions used for encryption of multiple messages

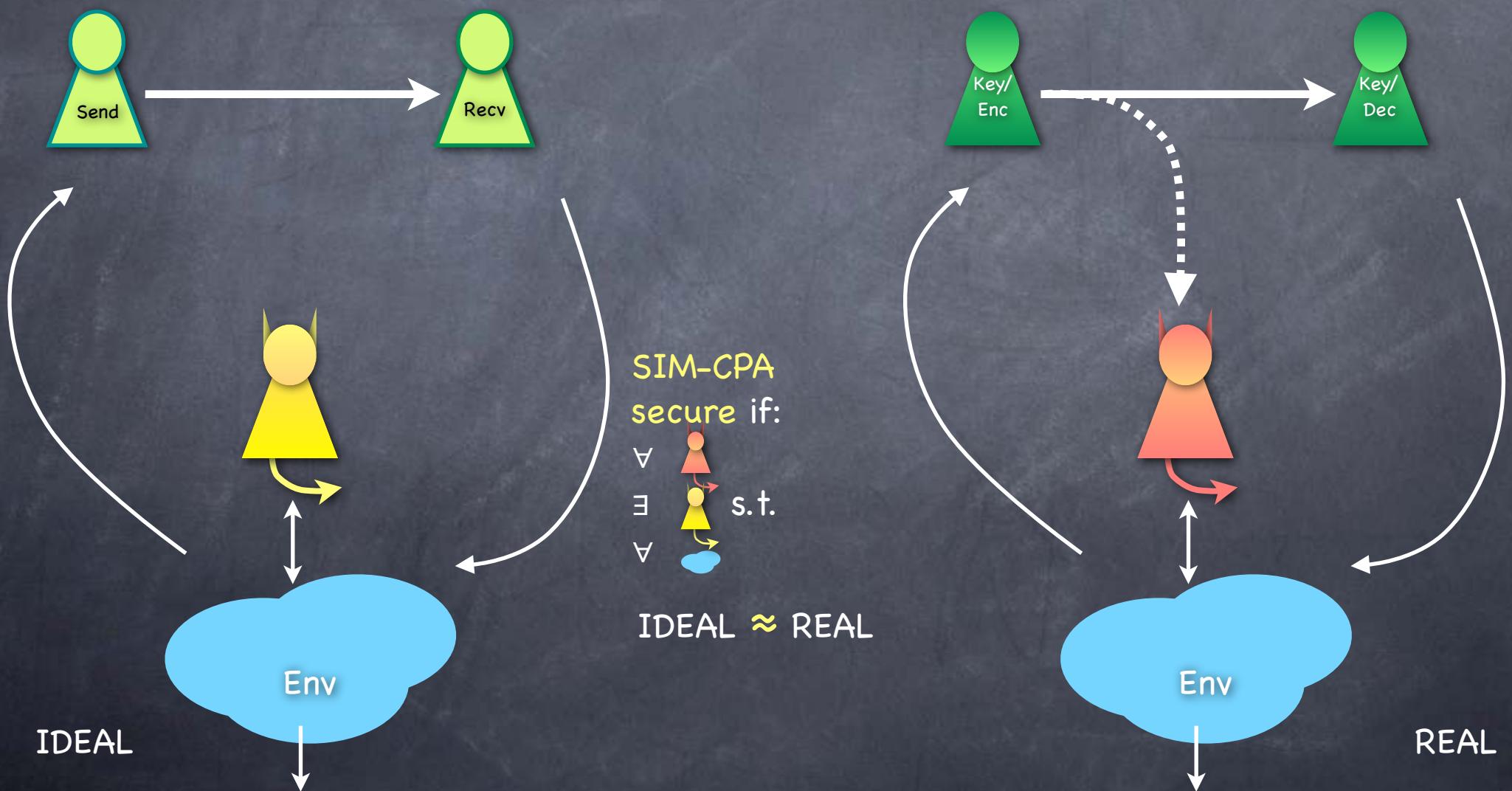
Symmetric-Key Encryption

The Syntax

- Shared-key (Private-key) Encryption
 - Key Generation: Randomized
 - $K \leftarrow \mathcal{K}$, uniformly randomly drawn from the key-space (or according to a key-distribution)
 - Encryption: Randomized
 - $\text{Enc}: \mathcal{M} \times \mathcal{K} \times \mathcal{R} \rightarrow \mathcal{C}$. During encryption a fresh random string will be chosen uniformly at random from \mathcal{R}
 - Decryption: Deterministic
 - $\text{Dec}: \mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$

Symmetric-Key Encryption

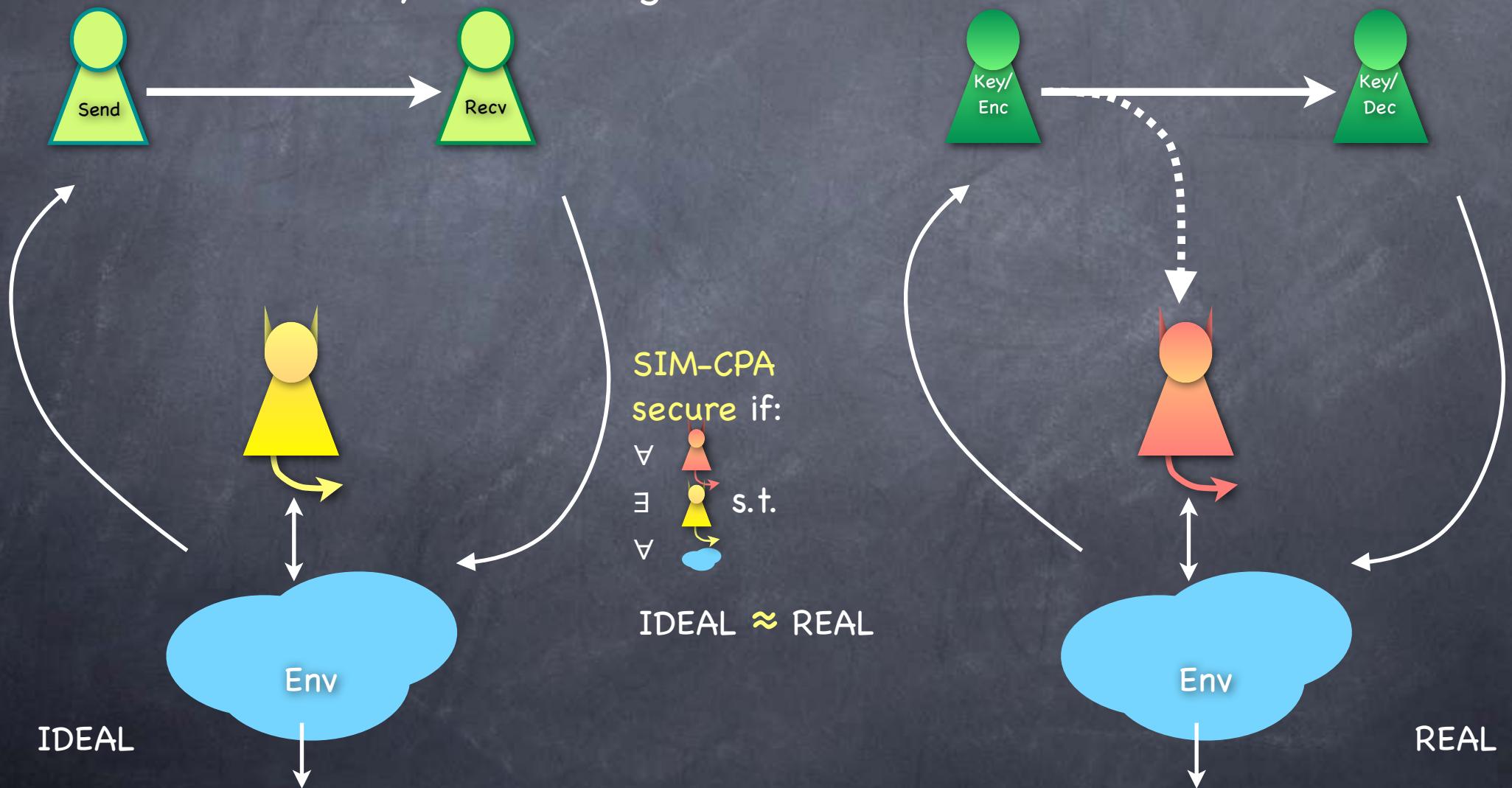
SIM-CPA Security



Symmetric-Key Encryption

SIM-CPA Security

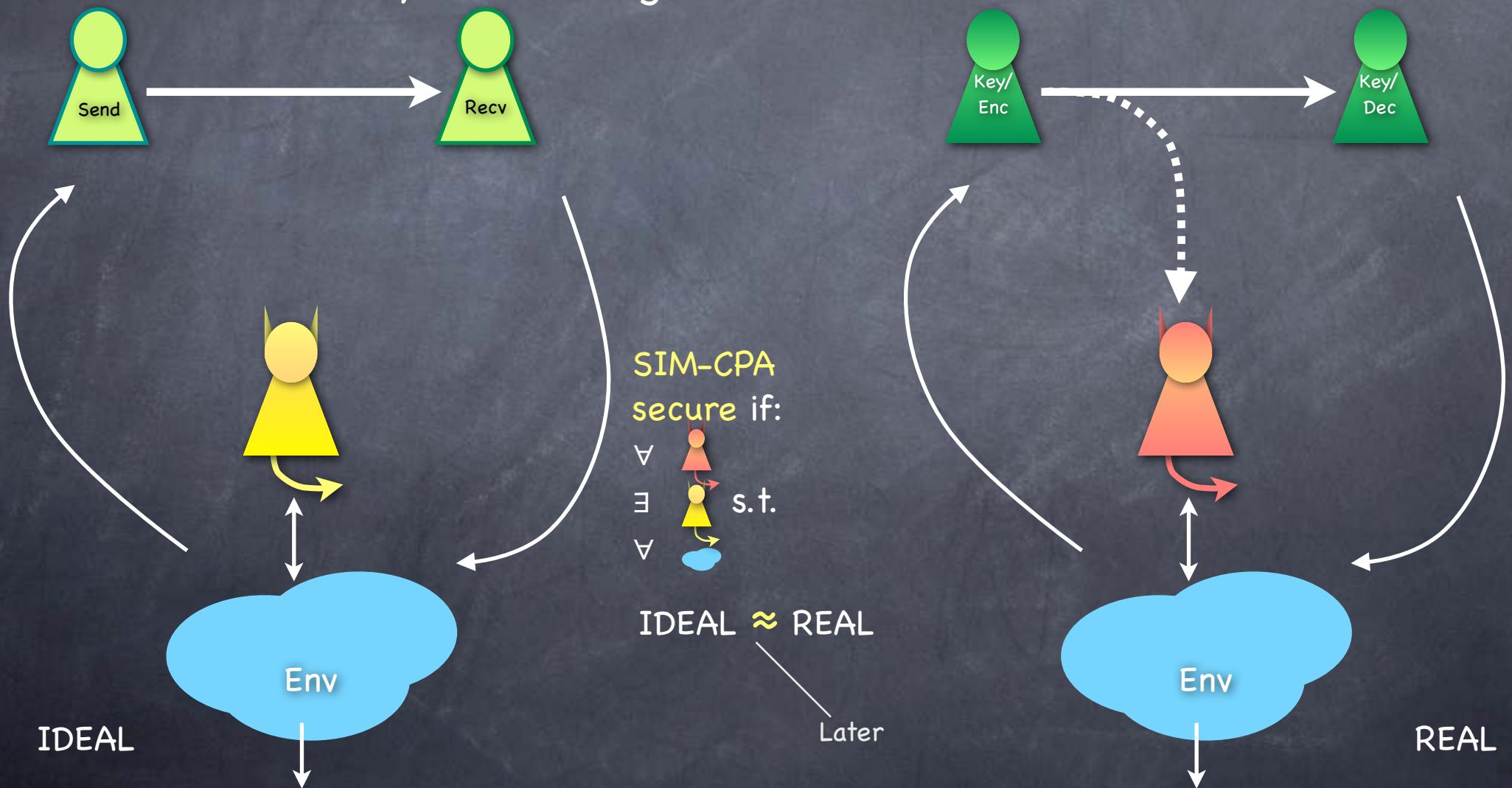
- Same as SIM-onetime security, but not restricted to environments which send only one message. All entities “efficient.”



Symmetric-Key Encryption

SIM-CPA Security

- Same as SIM-onetime security, but not restricted to environments which send only one message. All entities “efficient.”



Symmetric-Key Encryption

IND-CPA Security



Symmetric-Key Encryption

IND-CPA Security

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


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

Symmetric-Key Encryption

IND-CPA Security

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



- For as long as Adversary wants



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

Symmetric-Key Encryption

IND-CPA Security

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



- For as long as Adversary wants

- Adv sends two messages m_0, m_1 to the experiment



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

Symmetric-Key Encryption

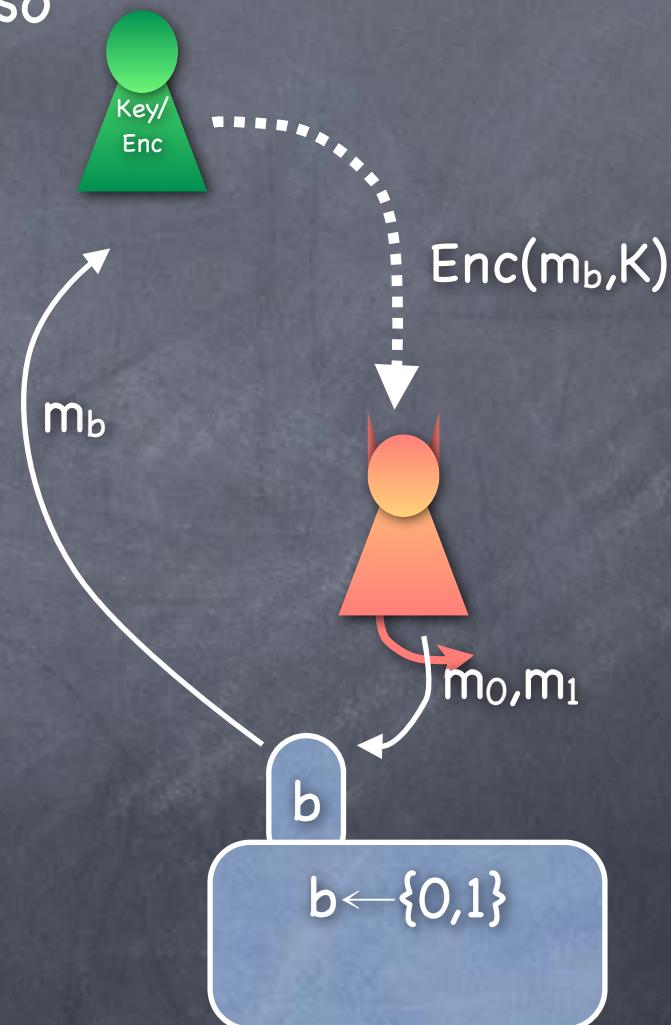
IND-CPA Security

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

- For as long as Adversary wants

- Adv sends two messages m_0, m_1 to the experiment

- Expt returns $\text{Enc}(m_b, K)$ to the adversary

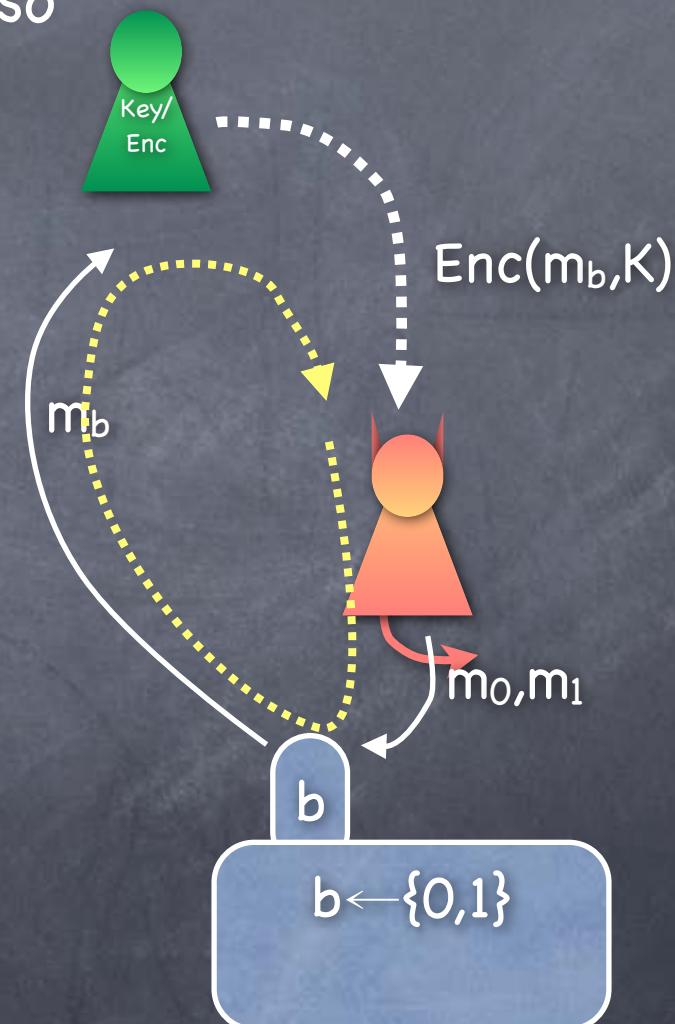


Symmetric-Key Encryption

IND-CPA Security

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

- For as long as Adversary wants
 - Adv sends two messages m_0, m_1 to the experiment
 - Expt returns $\text{Enc}(m_b, K)$ to the adversary

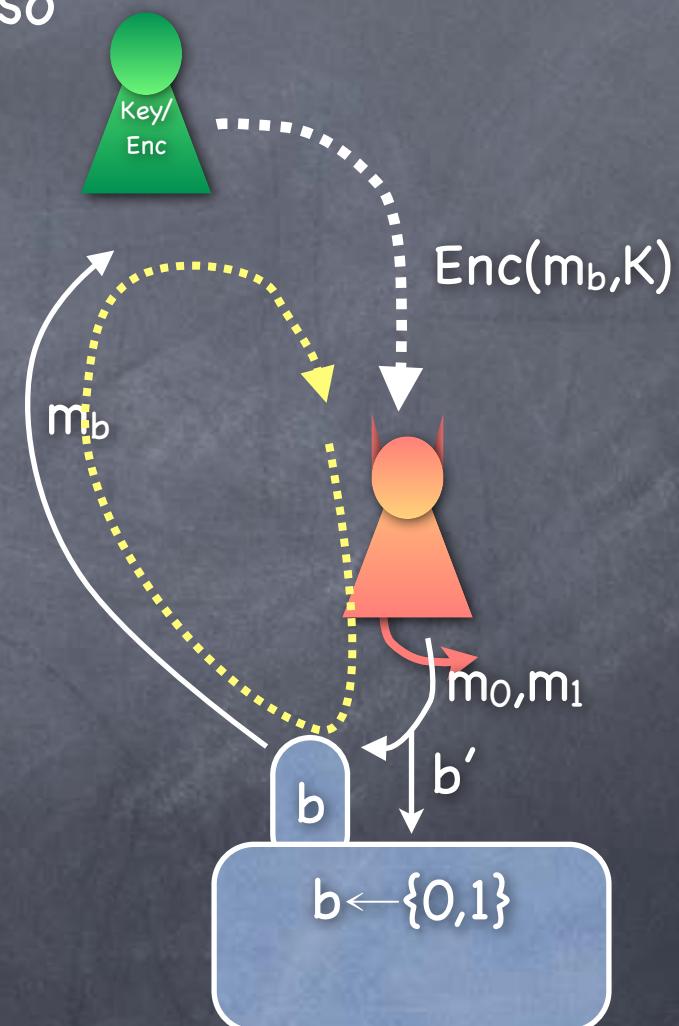


Symmetric-Key Encryption

IND-CPA Security

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

- For as long as Adversary wants
 - Adv sends two messages m_0, m_1 to the experiment
 - Expt returns $\text{Enc}(m_b, K)$ to the adversary
 - Adversary returns a guess b'

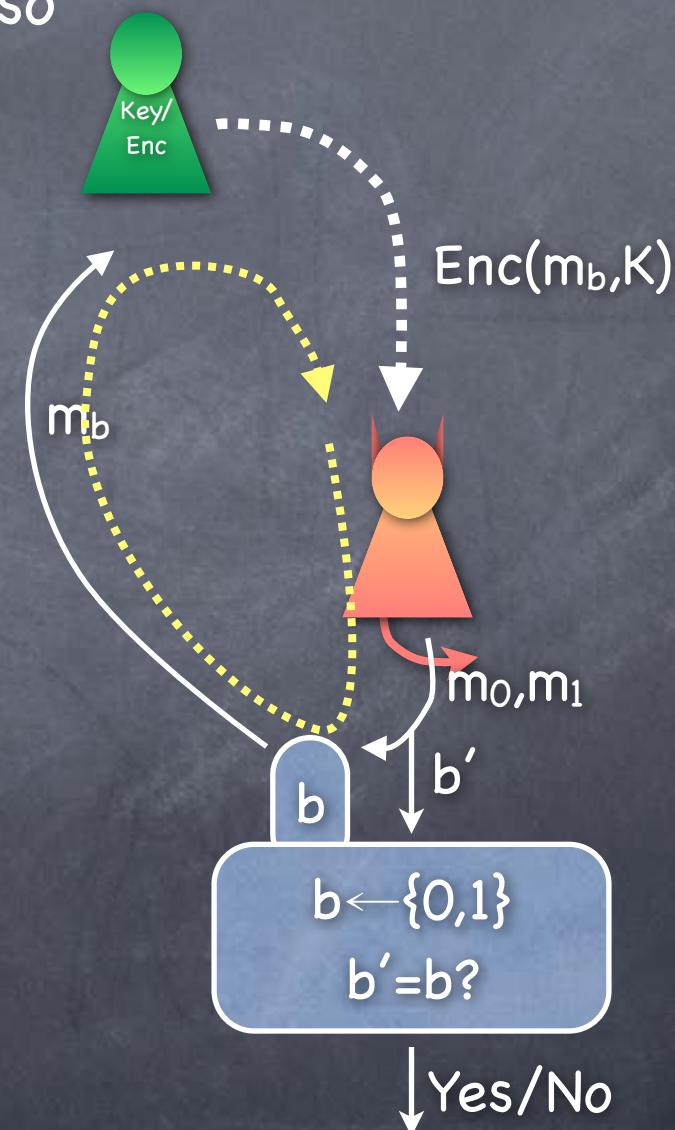


Symmetric-Key Encryption

IND-CPA Security

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

- For as long as Adversary wants
 - Adv sends two messages m_0, m_1 to the experiment
 - Expt returns $\text{Enc}(m_b, K)$ to the adversary
- Adversary returns a guess b'
- Experiment outputs 1 iff $b' = b$

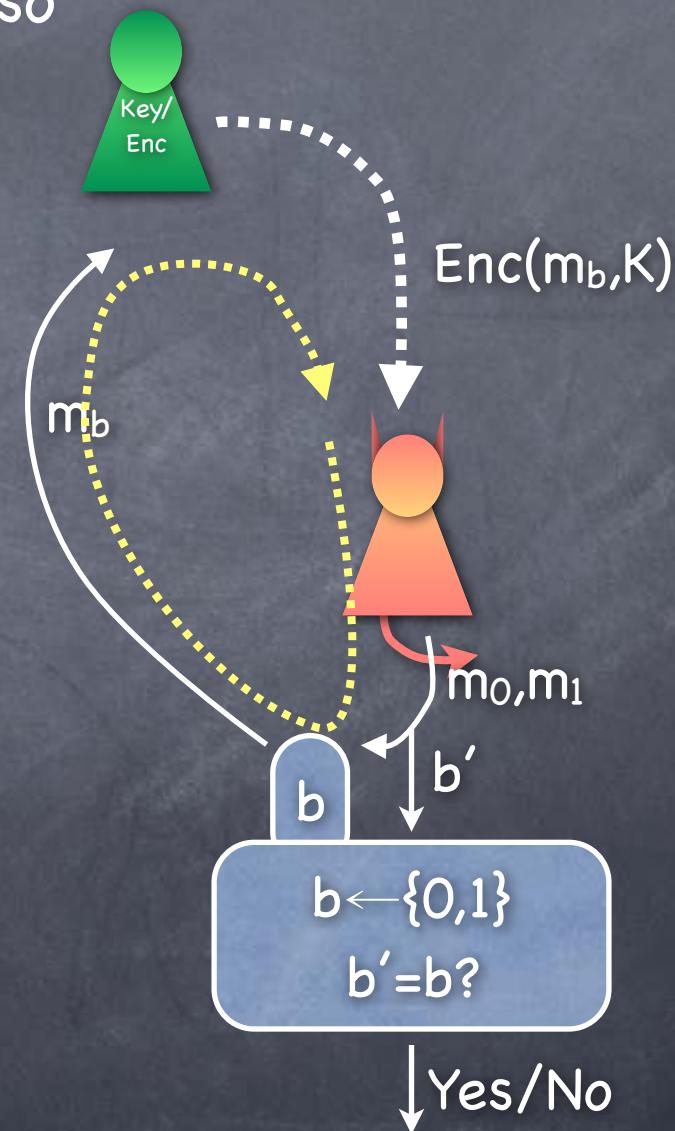


Symmetric-Key Encryption

IND-CPA Security

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

- For as long as Adversary wants
 - Adv sends two messages m_0, m_1 to the experiment
 - Expt returns $\text{Enc}(m_b, K)$ to the adversary
 - Adversary returns a guess b'
 - Experiment outputs 1 iff $b' = b$
- IND-CPA secure if for all “efficient” adversaries $\Pr[b' = b] \approx 1/2$

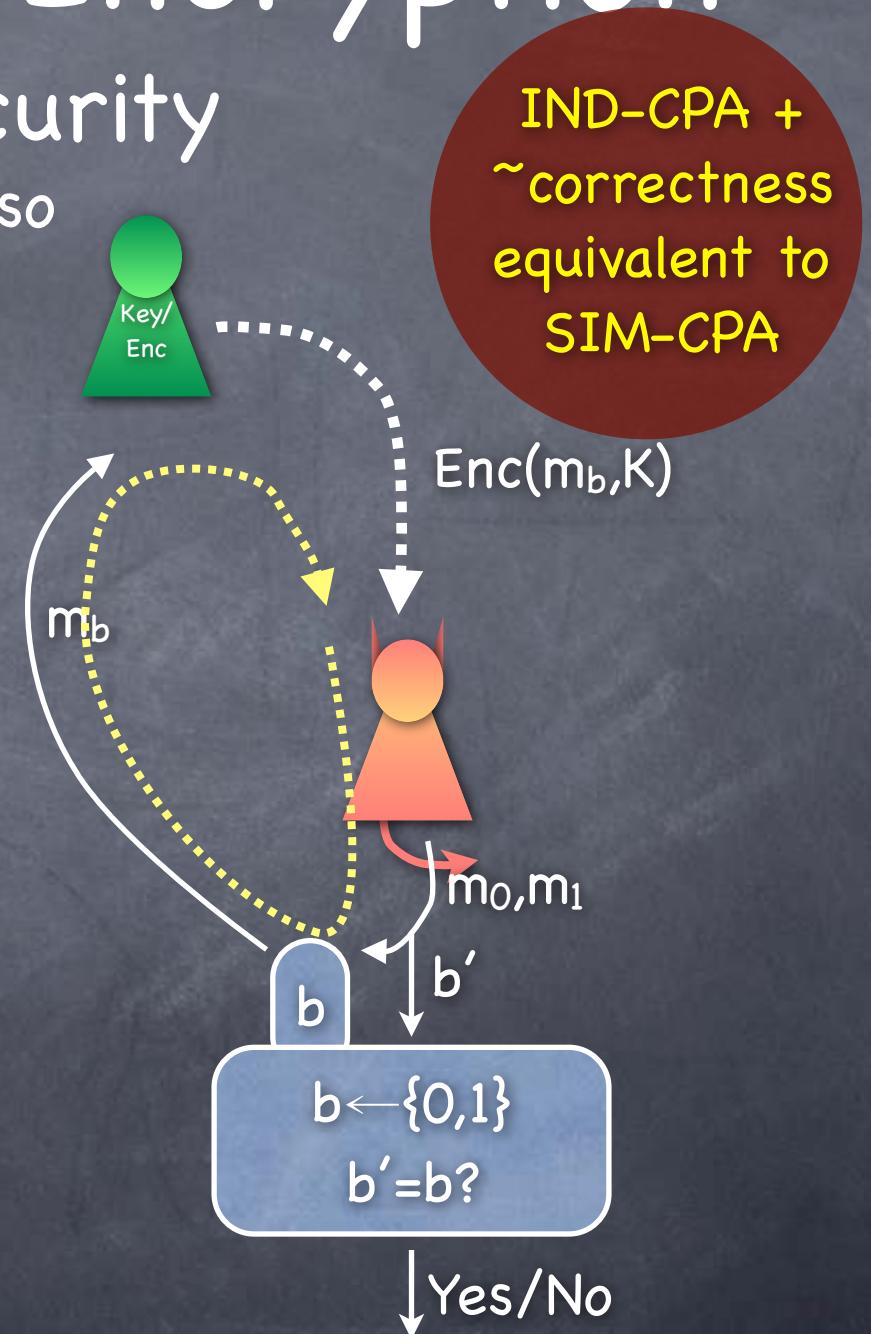


Symmetric-Key Encryption

IND-CPA Security

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

- For as long as Adversary wants
 - Adv sends two messages m_0, m_1 to the experiment
 - Expt returns $\text{Enc}(m_b, K)$ to the adversary
 - Adversary returns a guess b'
 - Experiment outputs 1 iff $b' = b$
- IND-CPA secure if for all “efficient” adversaries $\Pr[b' = b] \approx 1/2$



Definitions Summary

Definitions Summary

- Security definitions:

Definitions Summary

- Security definitions:
 - $\text{SIM-Onetime} = \text{IND-Onetime}/\text{Perfect Secrecy} + \text{correctness}$

Definitions Summary

- Security definitions:
 - $\text{SIM-Onetime} = \text{IND-Onetime}/\text{Perfect Secrecy} + \text{correctness}$
 - $\text{SIM-CPA} = \text{IND-CPA} + \sim\text{correctness}$: allows using the same key for multiple messages

Definitions Summary

- Security definitions:
 - SIM-Onetime = IND-Onetime/Perfect Secrecy + correctness
 - SIM-CPA = IND-CPA + \sim correctness: allows using the same key for multiple messages
 - Later: SIM-CCA = IND-CCA + \sim correctness: allows active attacks

Definitions Summary

- Security definitions:
 - $\text{SIM-Onetime} = \text{IND-Onetime}/\text{Perfect Secrecy} + \text{correctness}$
 - $\text{SIM-CPA} = \text{IND-CPA} + \sim\text{correctness}$: allows using the same key for multiple messages
 - Later: $\text{SIM-CCA} = \text{IND-CCA} + \sim\text{correctness}$: allows active attacks
- Next

Definitions Summary

- Security definitions:
 - $\text{SIM-Onetime} = \text{IND-Onetime}/\text{Perfect Secrecy} + \text{correctness}$
 - $\text{SIM-CPA} = \text{IND-CPA} + \sim\text{correctness}$: allows using the same key for multiple messages
 - Later: $\text{SIM-CCA} = \text{IND-CCA} + \sim\text{correctness}$: allows active attacks
- Next
 - For multi-message schemes we relaxed the “perfect” simulation requirement

Definitions Summary

- Security definitions:
 - SIM-Onetime = IND-Onetime/Perfect Secrecy + correctness
 - SIM-CPA = IND-CPA + \sim correctness: allows using the same key for multiple messages
 - Later: SIM-CCA = IND-CCA + \sim correctness: allows active attacks
- Next
 - For multi-message schemes we relaxed the “perfect” simulation requirement
 - But what is \approx ?

Feasible Computation

Feasible Computation

- In analyzing complexity of algorithms: Rate at which computational complexity grows with input size
 - e.g. Can do sorting in $O(n \log n)$

Feasible Computation

- In analyzing complexity of algorithms: Rate at which computational complexity grows with input size
 - e.g. Can do sorting in $O(n \log n)$
- Only the rough rate considered
 - Exact time depends on the technology

Feasible Computation

- In analyzing complexity of algorithms: Rate at which computational complexity grows with input size
 - e.g. Can do sorting in $O(n \log n)$
- Only the rough rate considered
 - Exact time depends on the technology
 - How much more computation will be needed as the instances of the problem get larger. (Do we scale well?)

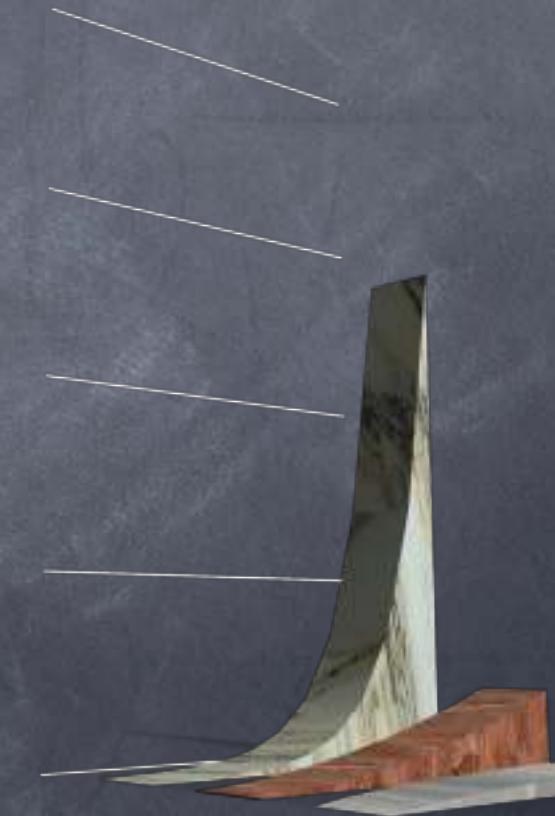
Feasible Computation

- In analyzing complexity of algorithms: Rate at which computational complexity grows with input size
 - e.g. Can do sorting in $O(n \log n)$
- Only the rough rate considered
 - Exact time depends on the technology
 - How much more computation will be needed as the instances of the problem get larger. (Do we scale well?)



Feasible Computation

- In analyzing complexity of algorithms: Rate at which computational complexity grows with input size
 - e.g. Can do sorting in $O(n \log n)$
- Only the rough rate considered
 - Exact time depends on the technology
 - How much more computation will be needed as the instances of the problem get larger. (Do we scale well?)
- “Polynomial time” ($O(n)$, $O(n^2)$, $O(n^3)$, ...) considered feasible



Infeasible Computation

Infeasible Computation

- “Super-Polynomial time” considered infeasible

Infeasible Computation

- “Super-Polynomial time” considered infeasible
 - e.g. 2^n , $2^{\sqrt{n}}$, $n^{\log(n)}$

Infeasible Computation

- “Super-Polynomial time” considered infeasible
 - e.g. 2^n , $2^{\sqrt{n}}$, $n^{\log(n)}$
 - i.e., as n grows, quickly becomes “infeasibly large”

Infeasible Computation

- “Super-Polynomial time” considered infeasible
 - e.g. 2^n , $2^{\sqrt{n}}$, $n^{\log(n)}$
 - i.e., as n grows, quickly becomes “infeasibly large”
- Can we make breaking security infeasible for Eve?

Infeasible Computation

- “Super-Polynomial time” considered infeasible
 - e.g. 2^n , $2^{\sqrt{n}}$, $n^{\log(n)}$
 - i.e., as n grows, quickly becomes “infeasibly large”
- Can we make breaking security infeasible for Eve?
 - What is n (that can grow)?

Infeasible Computation

- “Super-Polynomial time” considered infeasible
 - e.g. 2^n , $2^{\sqrt{n}}$, $n^{\log(n)}$
 - i.e., as n grows, quickly becomes “infeasibly large”
- Can we make breaking security infeasible for Eve?
 - What is n (that can grow)?
 - Message size?

Infeasible Computation

- “Super-Polynomial time” considered infeasible
 - e.g. 2^n , $2^{\sqrt{n}}$, $n^{\log(n)}$
 - i.e., as n grows, quickly becomes “infeasibly large”
- Can we make breaking security infeasible for Eve?
 - What is n (that can grow)?
 - Message size?
 - We need security even if sending only one bit!

Security Parameter

Security Parameter

- A parameter that is part of the encryption scheme

Security Parameter

- A parameter that is part of the encryption scheme
- Not related to message size

Security Parameter

- A parameter that is part of the encryption scheme
 - Not related to message size
 - A knob that can be used to set the security level

Security Parameter

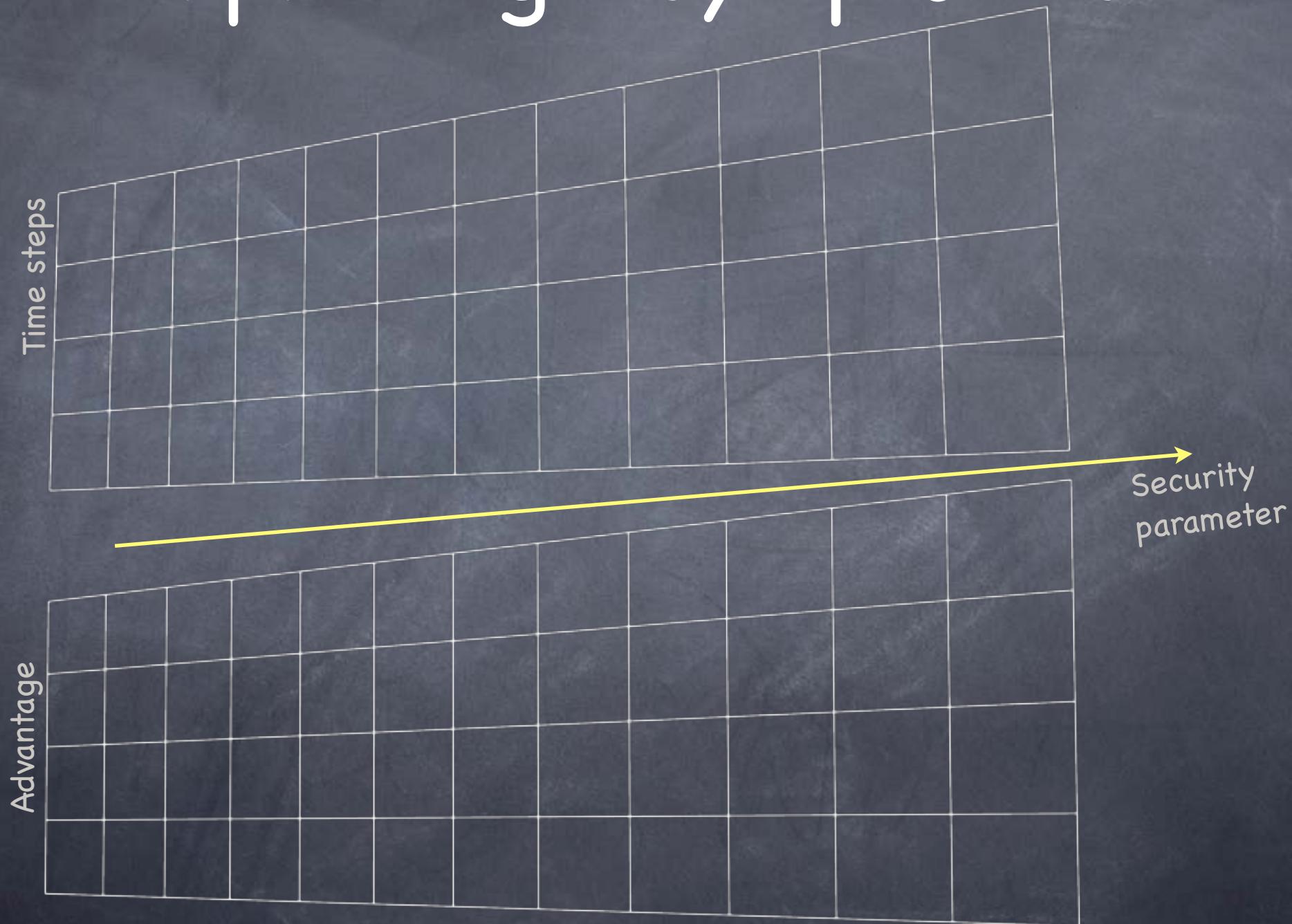
- A parameter that is part of the encryption scheme
 - Not related to message size
 - A knob that can be used to set the security level
 - Will denote by k

Security Parameter

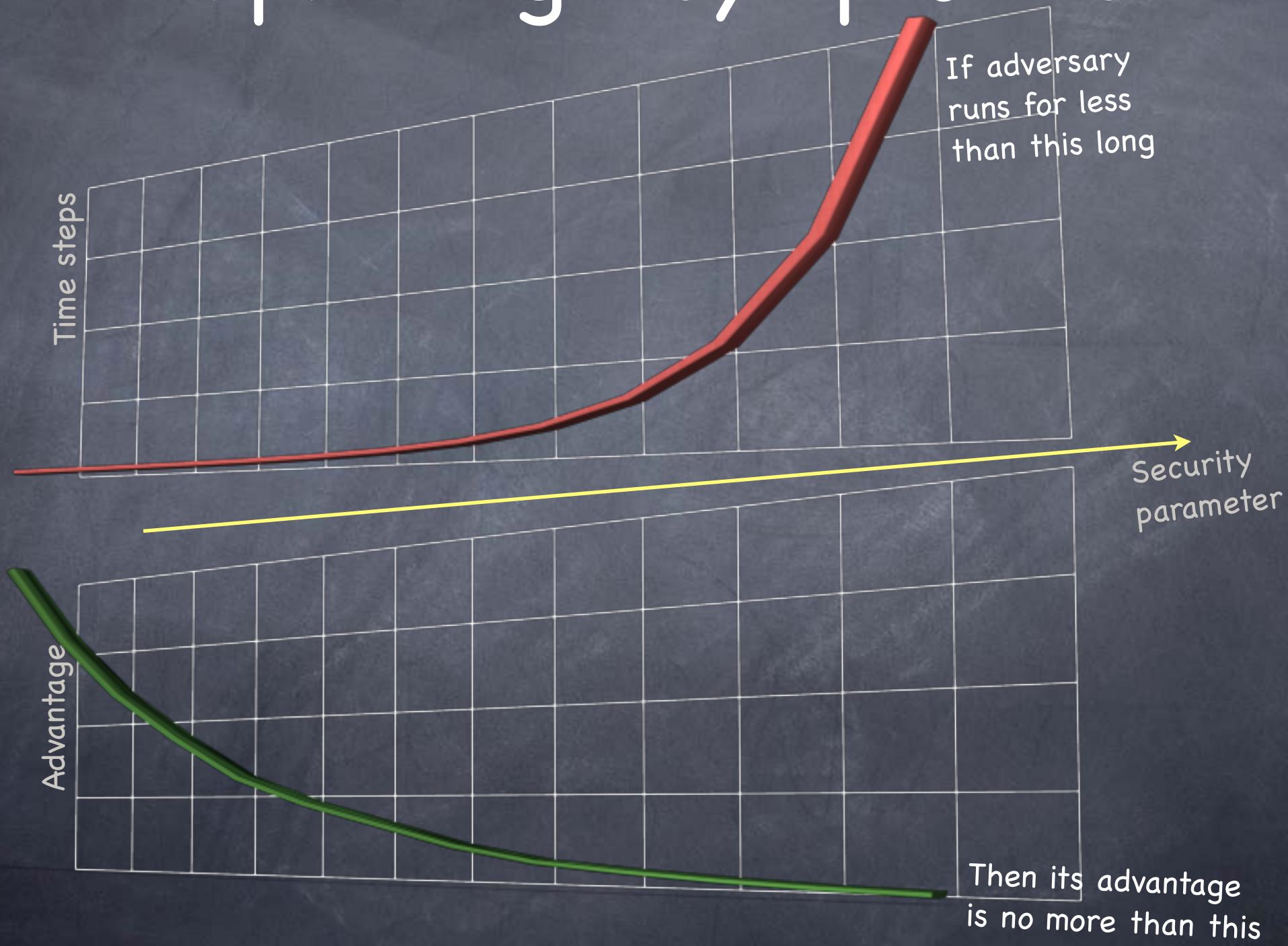
- A parameter that is part of the encryption scheme
 - Not related to message size
 - A knob that can be used to set the security level
 - Will denote by k
- Security guarantees are given asymptotically as a function of the security parameter

Interpreting Asymptotics

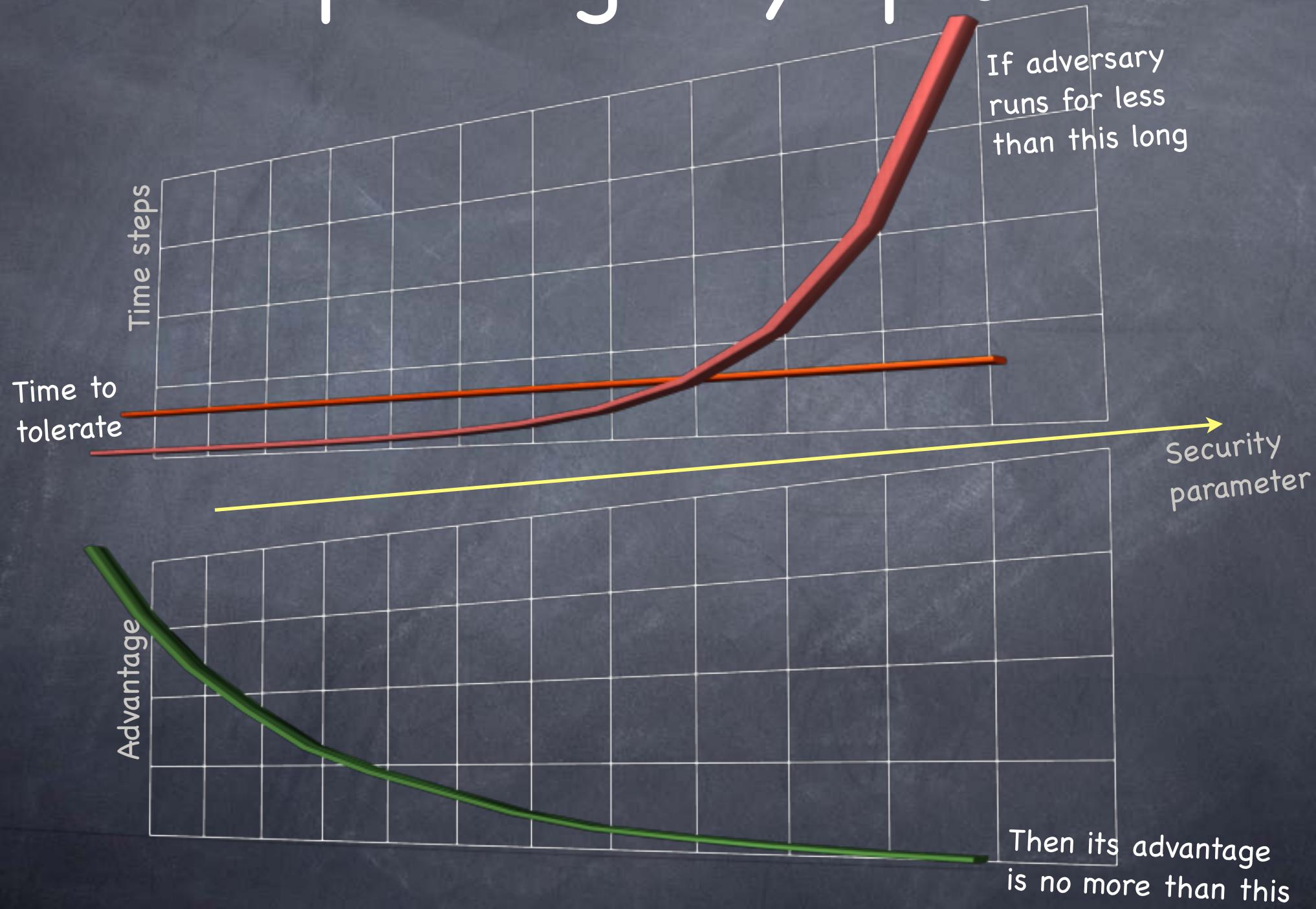
Interpreting Asymptotics



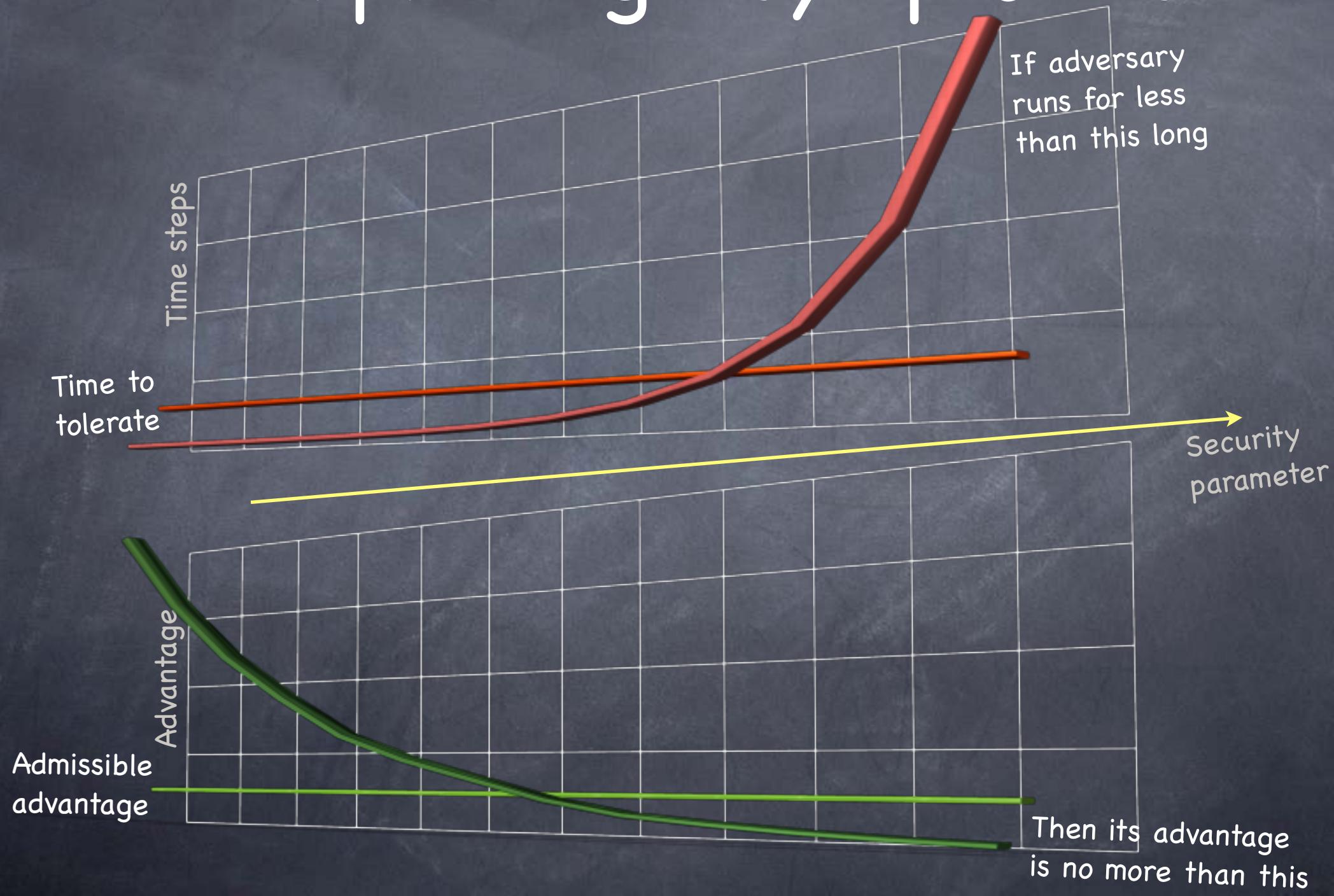
Interpreting Asymptotics



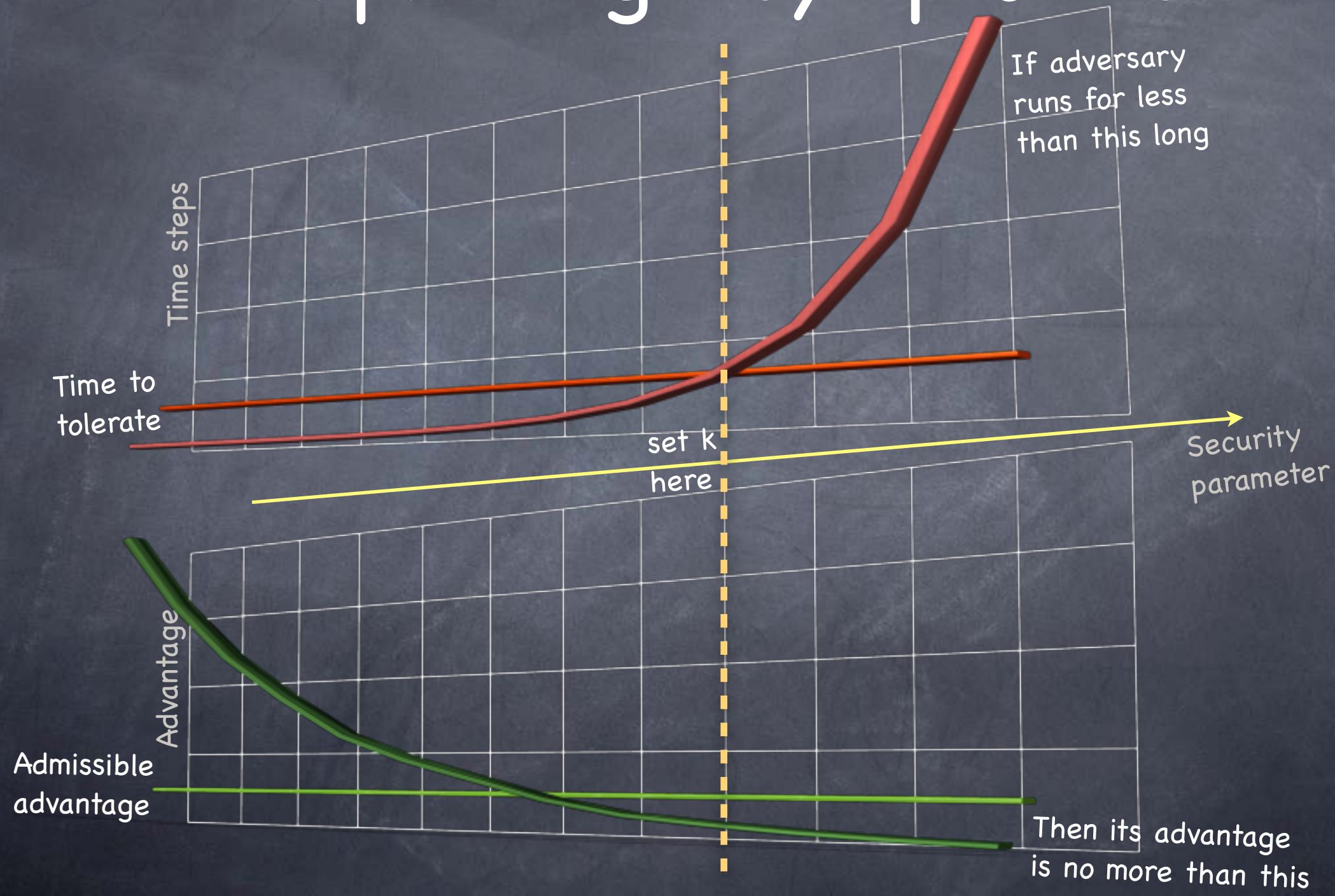
Interpreting Asymptotics



Interpreting Asymptotics



Interpreting Asymptotics



Feasible and Negligible

Feasible and Negligible

- We want to tolerate Eves who have a running time bounded by some polynomial in k

Feasible and Negligible

- We want to tolerate Eves who have a running time bounded by some polynomial in k
- Eve could toss coins: **Probabilistic Polynomial-Time (PPT)**

Feasible and Negligible

- We want to tolerate Eves who have a running time bounded by some polynomial in k
 - Eve could toss coins: **Probabilistic Polynomial-Time (PPT)**
 - It is better that we allow Eve high polynomial times too (we'll typically tolerate some super-polynomial time for Eve)

Feasible and Negligible

- We want to tolerate Eves who have a running time bounded by some polynomial in k
 - Eve could toss coins: **Probabilistic Polynomial-Time (PPT)**
 - It is better that we allow Eve high polynomial times too (we'll typically tolerate some super-polynomial time for Eve)
 - But algorithms for Alice/Bob better be very efficient

Feasible and Negligible

- We want to tolerate Eves who have a running time bounded by some polynomial in k
 - Eve could toss coins: **Probabilistic Polynomial-Time (PPT)**
 - It is better that we allow Eve high polynomial times too (we'll typically tolerate some super-polynomial time for Eve)
 - But algorithms for Alice/Bob better be very efficient
 - Eve could be **non-uniform**: a different strategy for each k

Feasible and Negligible

- We want to tolerate Eves who have a running time bounded by some polynomial in k
 - Eve could toss coins: **Probabilistic Polynomial-Time (PPT)**
 - It is better that we allow Eve high polynomial times too (we'll typically tolerate some super-polynomial time for Eve)
 - But algorithms for Alice/Bob better be very efficient
 - Eve could be **non-uniform**: a different strategy for each k
- Such an Eve should have only a “negligible” advantage (or, should cause at most a “negligible” difference in the behavior of the environment in the SIM definition)

Feasible and Negligible

- We want to tolerate Eves who have a running time bounded by some polynomial in k
 - Eve could toss coins: **Probabilistic Polynomial-Time (PPT)**
 - It is better that we allow Eve high polynomial times too (we'll typically tolerate some super-polynomial time for Eve)
 - But algorithms for Alice/Bob better be very efficient
 - Eve could be **non-uniform**: a different strategy for each k
- Such an Eve should have only a “negligible” advantage (or, should cause at most a “negligible” difference in the behavior of the environment in the SIM definition)
 - What is **negligible**?

Negligibly Small

Negligibly Small

- A negligible quantity: As we turn the knob the quantity should “**decrease extremely fast**”

Negligibly Small

- A negligible quantity: As we turn the knob the quantity should “**decrease extremely fast**”
- Negligible: decreases as $1/\text{superpoly}(k)$

Negligibly Small

- A negligible quantity: As we turn the knob the quantity should “**decrease extremely fast**”
- Negligible: decreases as $1/\text{superpoly}(k)$
 - i.e., faster than $1/\text{poly}(k)$ for every polynomial

Negligibly Small

- A negligible quantity: As we turn the knob the quantity should “**decrease extremely fast**”
- Negligible: decreases as $1/\text{superpoly}(k)$
 - i.e., faster than $1/\text{poly}(k)$ for every polynomial
 - e.g.: 2^{-k} , $2^{-\sqrt{k}}$, $k^{-(\log k)}$.

Negligibly Small

- A negligible quantity: As we turn the knob the quantity should “**decrease extremely fast**”
- Negligible: decreases as $1/\text{superpoly}(k)$
 - i.e., faster than $1/\text{poly}(k)$ for every polynomial
 - e.g.: 2^{-k} , $2^{-\sqrt{k}}$, $k^{-(\log k)}$.
 - Formally: T negligible if $\forall c > 0 \ \exists k_0 \ \forall k > k_0 \ T(k) < 1/k^c$

Negligibly Small

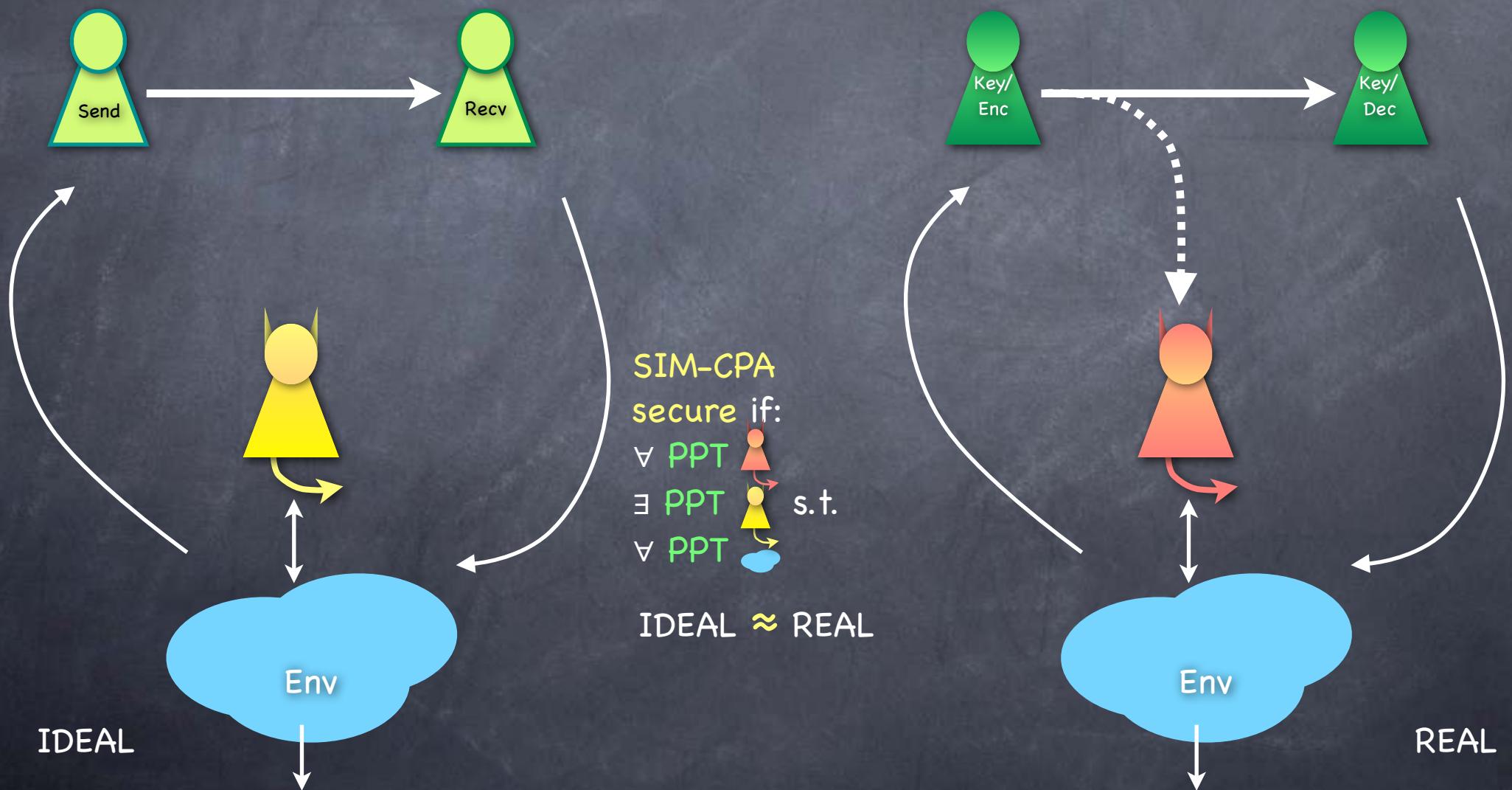
- A negligible quantity: As we turn the knob the quantity should “decrease extremely fast”
 - Negligible: decreases as $1/\text{superpoly}(k)$
 - i.e., faster than $1/\text{poly}(k)$ for every polynomial
 - e.g.: 2^{-k} , $2^{-\sqrt{k}}$, $k^{-(\log k)}$.
 - Formally: T negligible if $\forall c > 0 \ \exists k_0 \ \forall k > k_0 \ T(k) < 1/k^c$
 - So that $\text{negl}(k) \times \text{poly}(k) = \text{negl}'(k)$

Negligibly Small

- A negligible quantity: As we turn the knob the quantity should “decrease extremely fast”
 - Negligible: decreases as $1/\text{superpoly}(k)$
 - i.e., faster than $1/\text{poly}(k)$ for every polynomial
 - e.g.: 2^{-k} , $2^{-\sqrt{k}}$, $k^{-(\log k)}$.
 - Formally: T negligible if $\forall c > 0 \ \exists k_0 \ \forall k > k_0 \ T(k) < 1/k^c$
 - So that $\text{negl}(k) \times \text{poly}(k) = \text{negl}'(k)$
 - Needed, because Eve can often increase advantage polynomially by spending that much more time/by seeing that many more messages

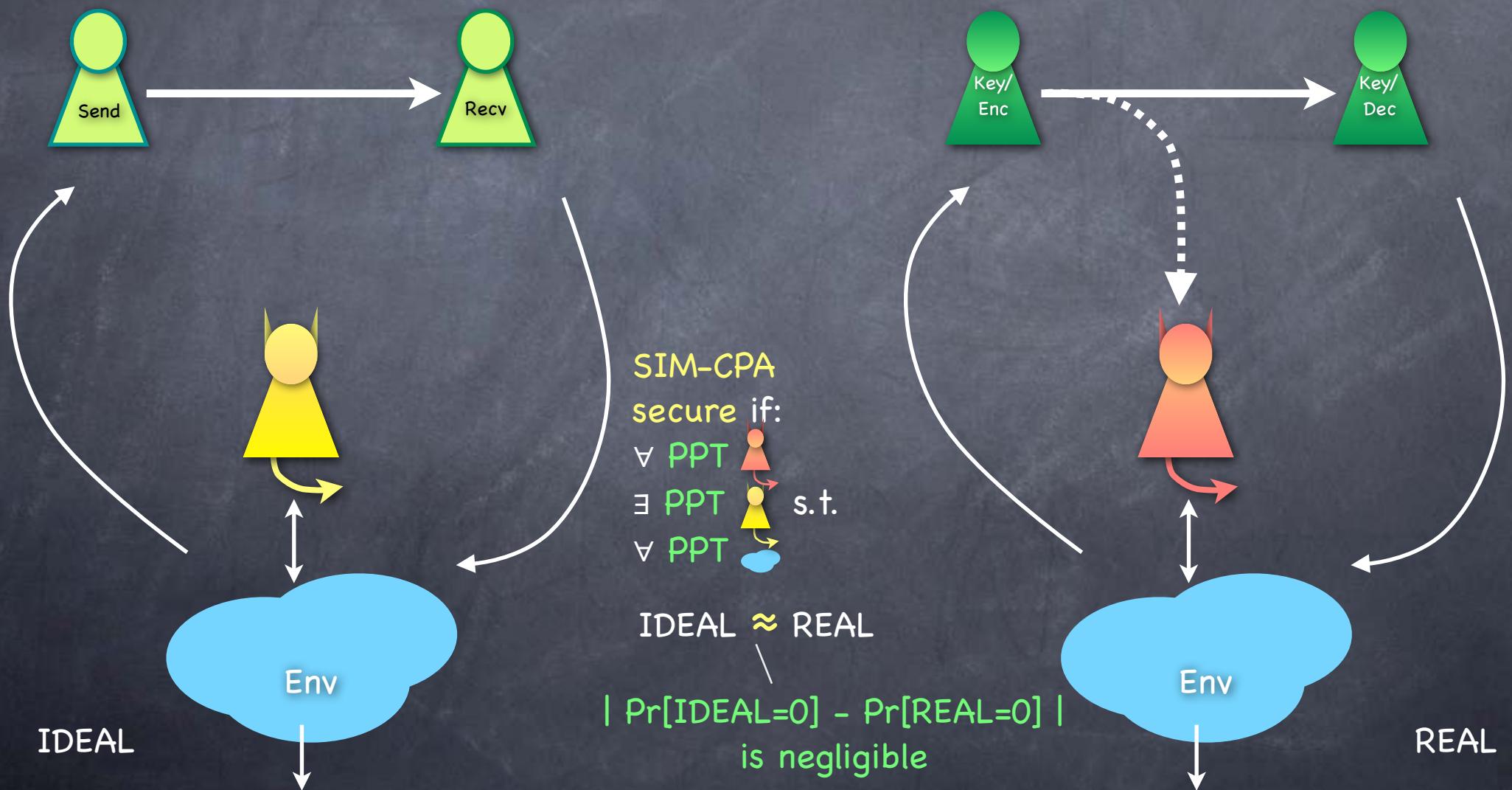
Symmetric-Key Encryption

SIM-CPA Security



Symmetric-Key Encryption

SIM-CPA Security



Next

- Constructing (CPA-secure) SKE schemes
 - Pseudorandomness Generator (PRG)
 - One-Way Functions (& OW Permutations)
 - OWP → PRG → (CPA-secure) SKE