Defining Encryption (ctd.)

Lecture 3

SIM & IND security

Beyond One-Time: **CPA** security

Computational Indistinguishability

Recall

Onetime Encryption

Perfect Secrecy

A (2,2)-secret-sharing scheme: K and Enc(m,K) are shares of m

- **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).

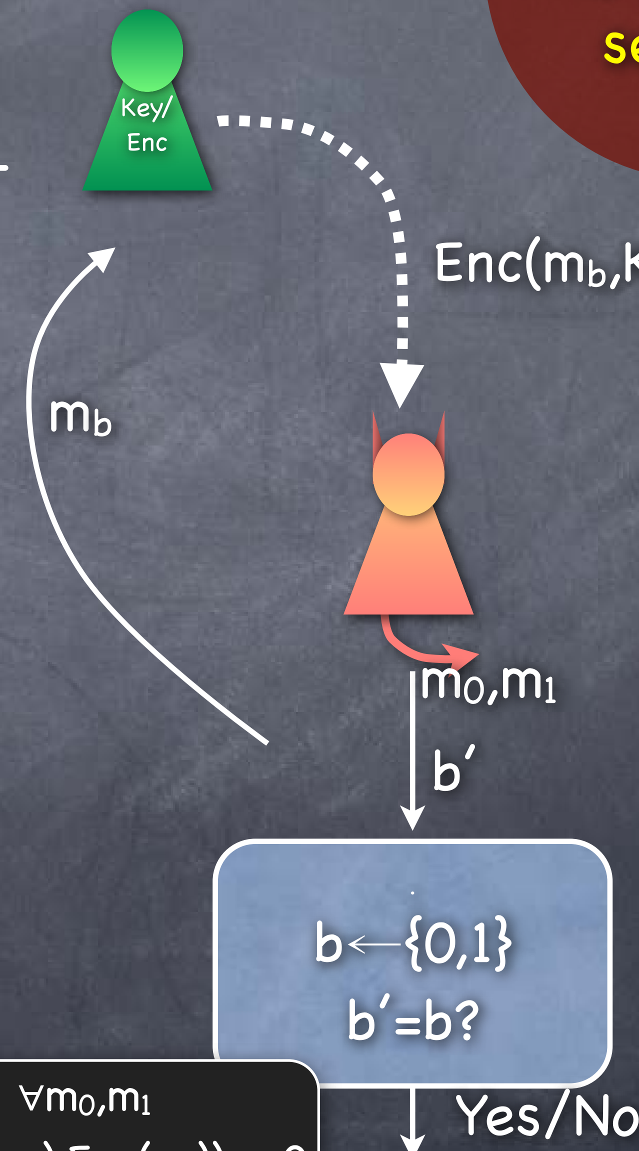
Recall

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



$\forall m_0,m_1$
$\Delta(\mathrm{Enc}(m_0),\mathrm{Enc}(m_1)) = 0$

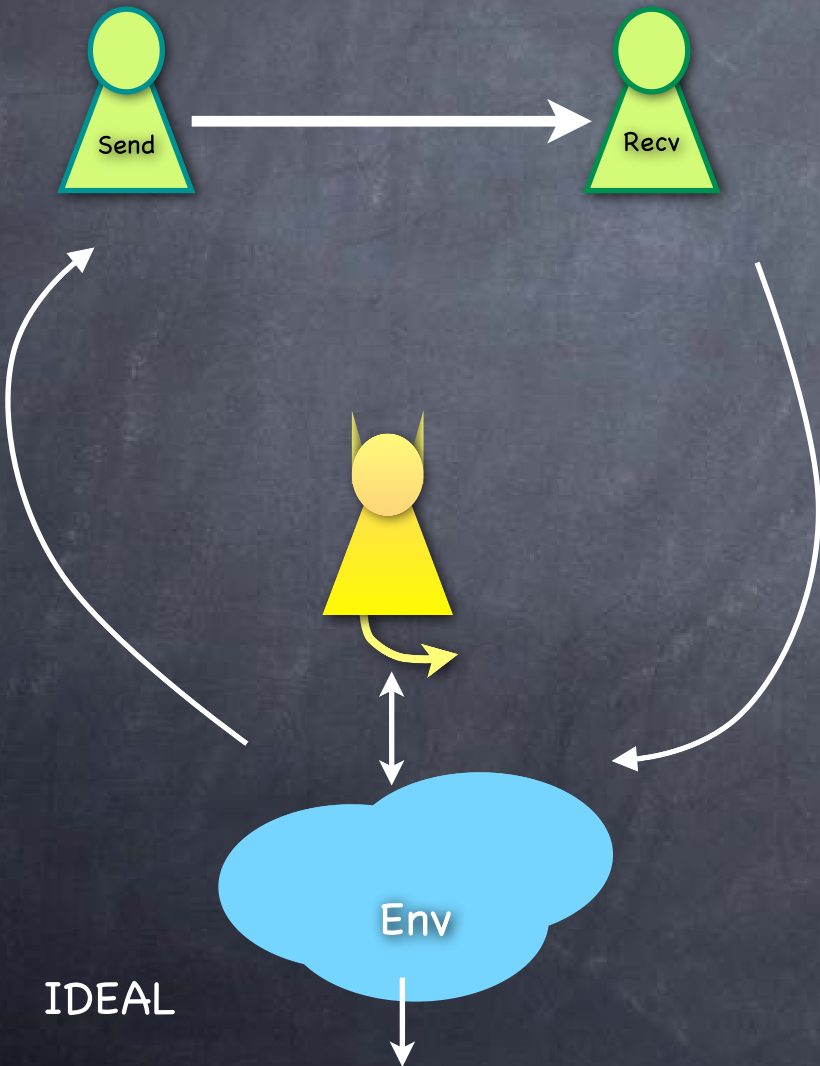
Recall

Onetime Encryption

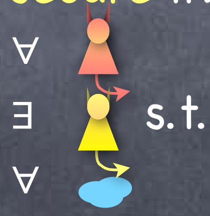
SIM-Onetime Security

Equivalent to perfect secrecy + correctness

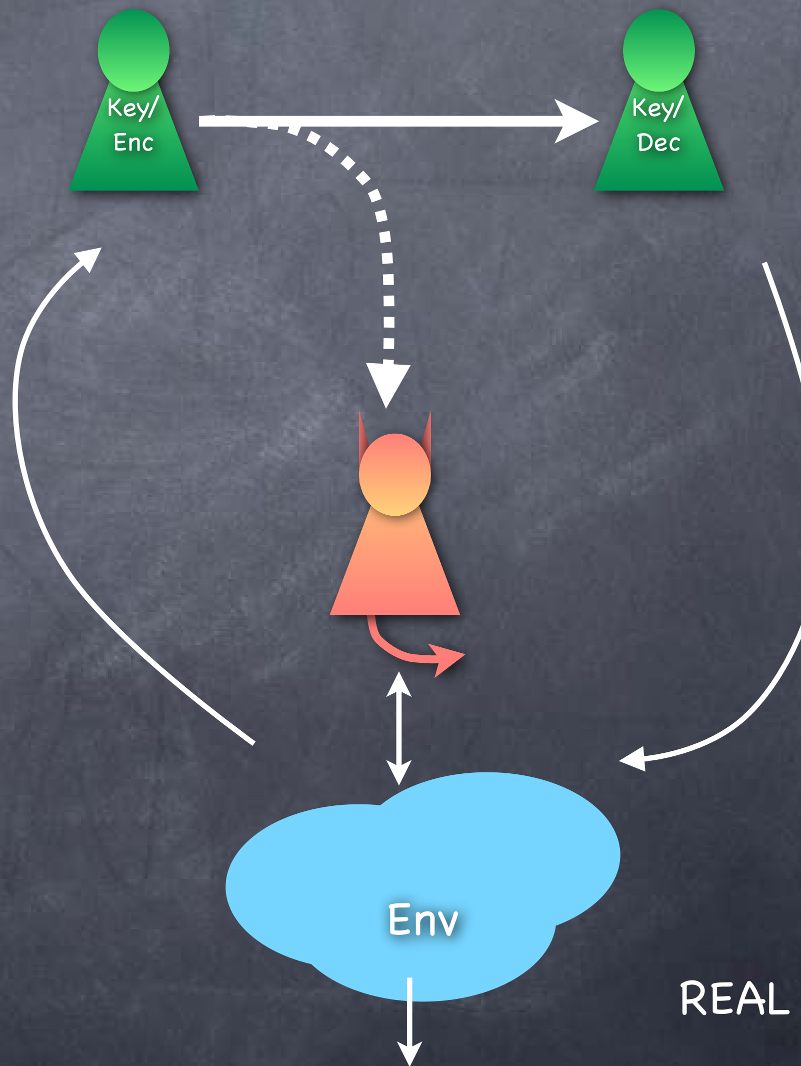
- Class of environments which send only one message



SIM-Onetime secure if:



IDEAL=REAL



Security of Encryption

- Perfect secrecy is too strong for multiple messages (though, as we shall see later, 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
    - 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
    - Dec: $\mathcal{C} \times \mathcal{K} \rightarrow \mathcal{M}$

Symmetric-Key Encryption

Security Definitions

Security of Encryption	Information theoretic		Game-based		Simulation-based
One-time	Perfect secrecy & Perfect correctness	≡	IND-Onetime & Perfect correctness	≡	SIM-Onetime
Multi-msg			IND-CPA & correctness	≡	SIM-CPA
Active/multi-msg			IND-CCA & correctness	≡	SIM-CCA

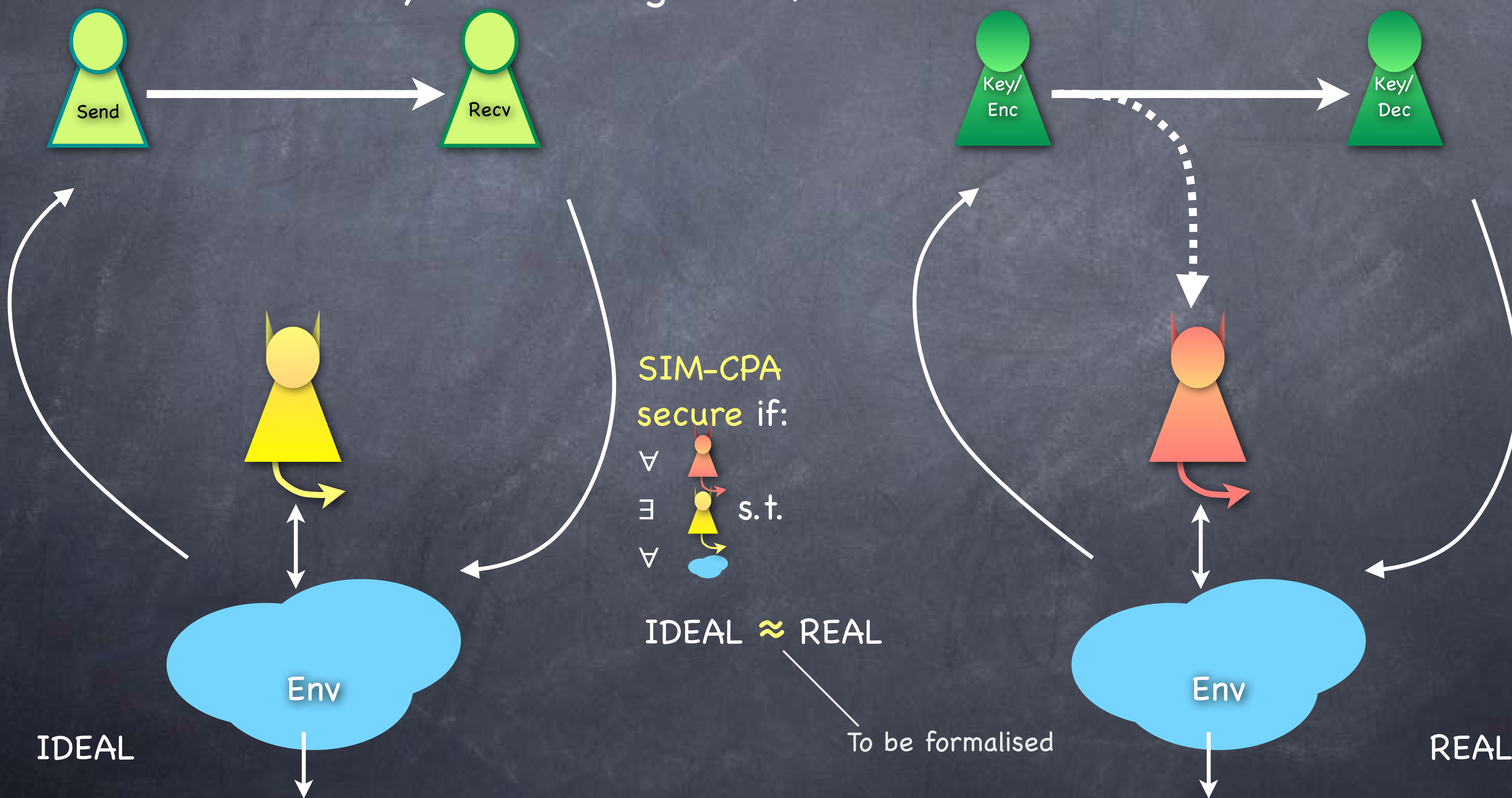
today

- CPA: Chosen Plaintext Attack
  - The adversary can influence/choose the messages being encrypted
  - Note: One-time security also allowed this, but for only one message

Symmetric-Key Encryption

SIM-CPA Security

- Same as SIM-onetime security, but not restricted to environments which send only one message. Also, now all entities "efficient."

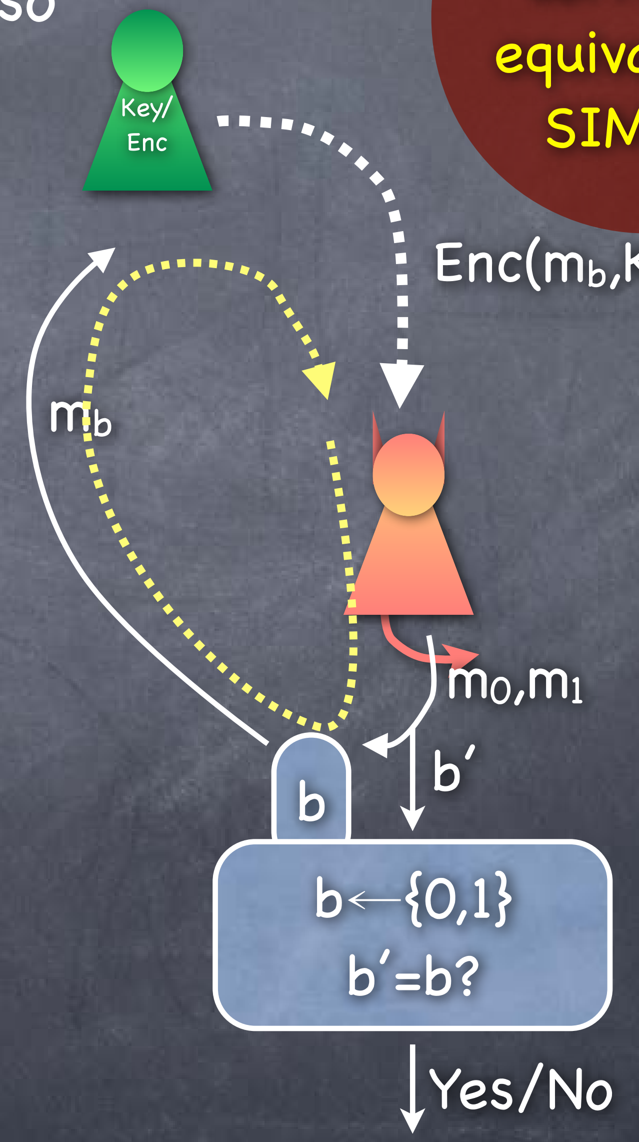


Symmetric-Key Encryption

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 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] ≈ 1/2

IND-CPA + ~correctness equivalent to SIM-CPA

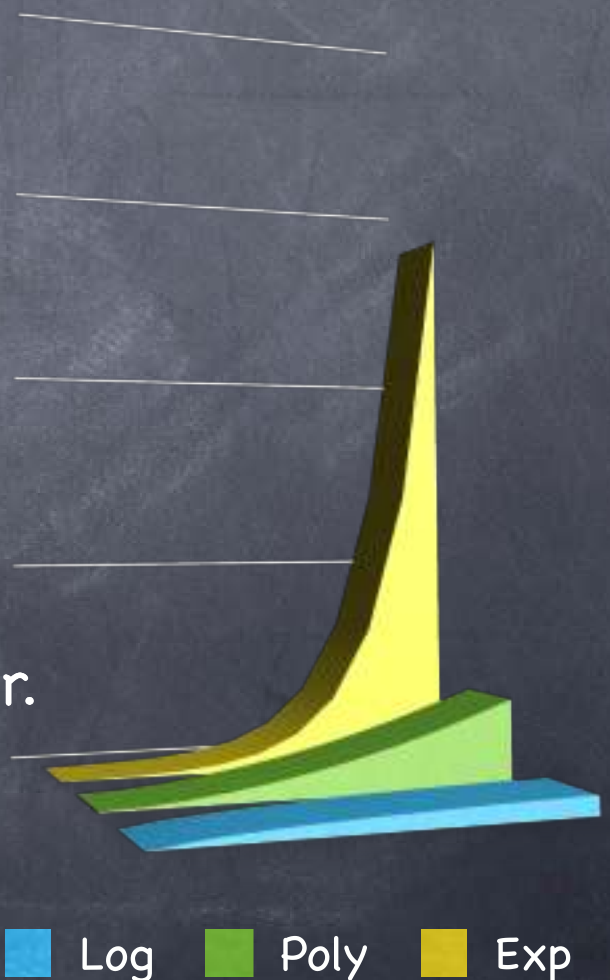


Almost Perfect

- For multi-message schemes we relaxed the "perfect" simulation requirement to IDEAL ≈ REAL
- In particular, we settle for "almost perfect" correctness
  - Recall perfect correctness
    - $\forall$ m, $\Pr_{K \leftarrow \text{KeyGen, Enc}}$ [Dec(Enc(m,K), K) = m] = 1
  - Almost perfect correctness: a.k.a. **Statistical correctness**
    - $\forall$ m, $\Pr_{K \leftarrow \text{KeyGen, Enc}}$ [Dec(Enc(m,K), K) = m] ≈ 1
  - But what is ≈ ?

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
    - Real question: Do we scale well? How much more computation will be needed as the instances of the problem get larger.
    - "Polynomial time" ($O(n)$, $O(n^2)$, $O(n^3)$, ...) considered feasible



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

- Take a 256 bit integer, 11…1 = $2^{256}-1$
- Can a computer just count up to this number?
  - No! Not even if it runs
    - at the frequency of molecular vibrations (10^{14} Hz)
    - for the entire estimated lifetime of the universe (< 10^{18} s)
- What if you recruited every atom in the earth ($\approx 10^{50}$) to do the same?
  - OK, but still will get only to $10^{82} \approx 2^{272}$.
  - And even if you recruited every elementary particle in the known universe ($\approx 10^{80}$), only up to $10^{112} \approx 2^{372}$
- **The whole known universe can't count up to a 400-bit number!**

Infeasible Computation

- The whole known universe can't count up to a 400-bit number!
- But we can quickly add, multiply, divide and exponentiate much larger numbers
  - Roughly, can "compute on" n-bit numbers in n or n^2 steps. Many important problems have such polynomial time algorithms.
- But for some problems we don't know algorithms that do much better than 2^n, $2^{n/2}$ etc.
  - We believe for some such problems no better algorithms exist!
- We will crucially rely on such assumptions

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?
  - But what is n (that can grow)?
  - Message size?
    - We need security even if sending only one bit!

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 <u>asymptotically</u> as a function of the security parameter

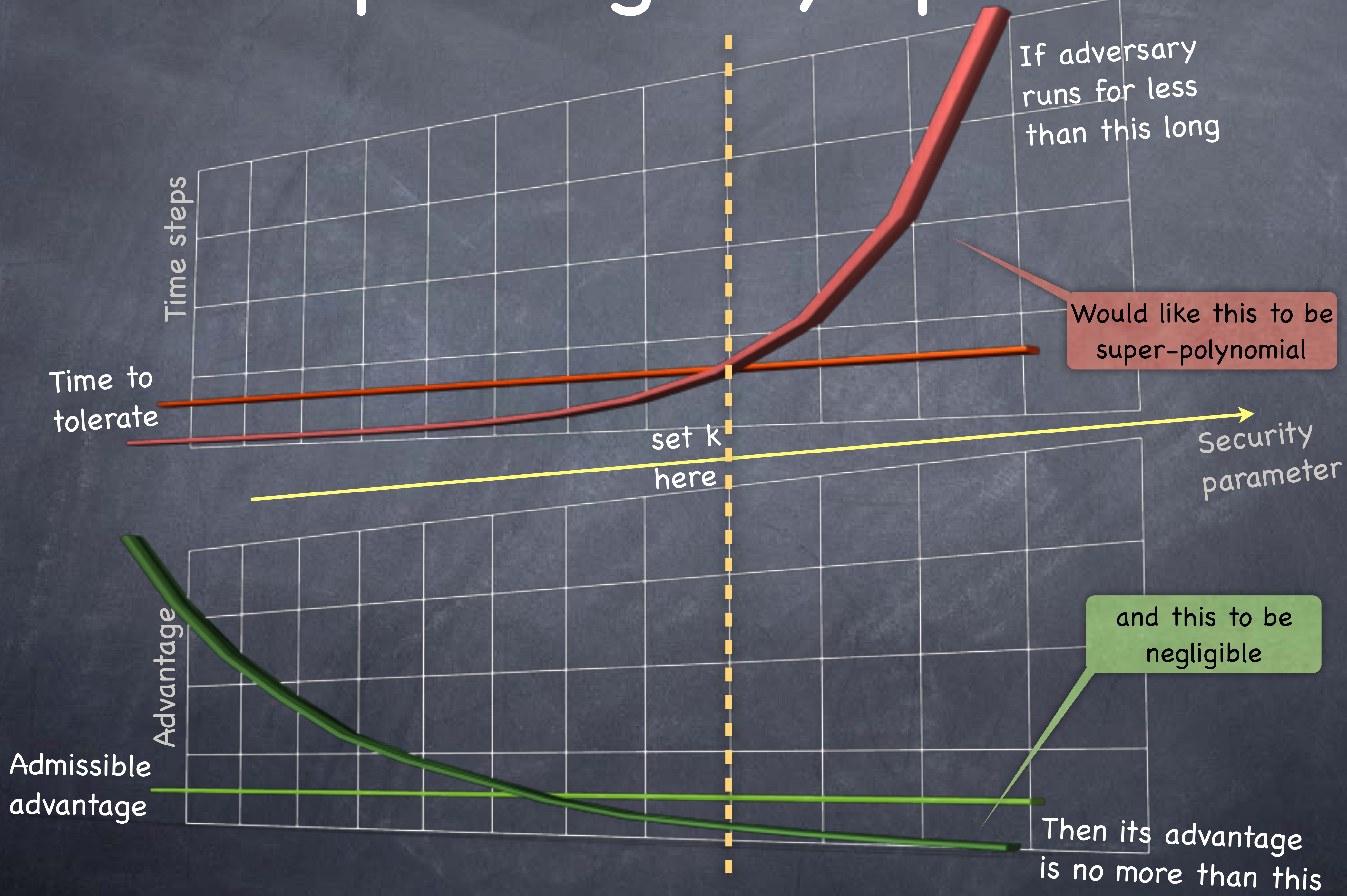
Feasible and Negligible

- 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 behaviour of the environment in the SIM definition)
    - What is negligible?

Negligibly Small

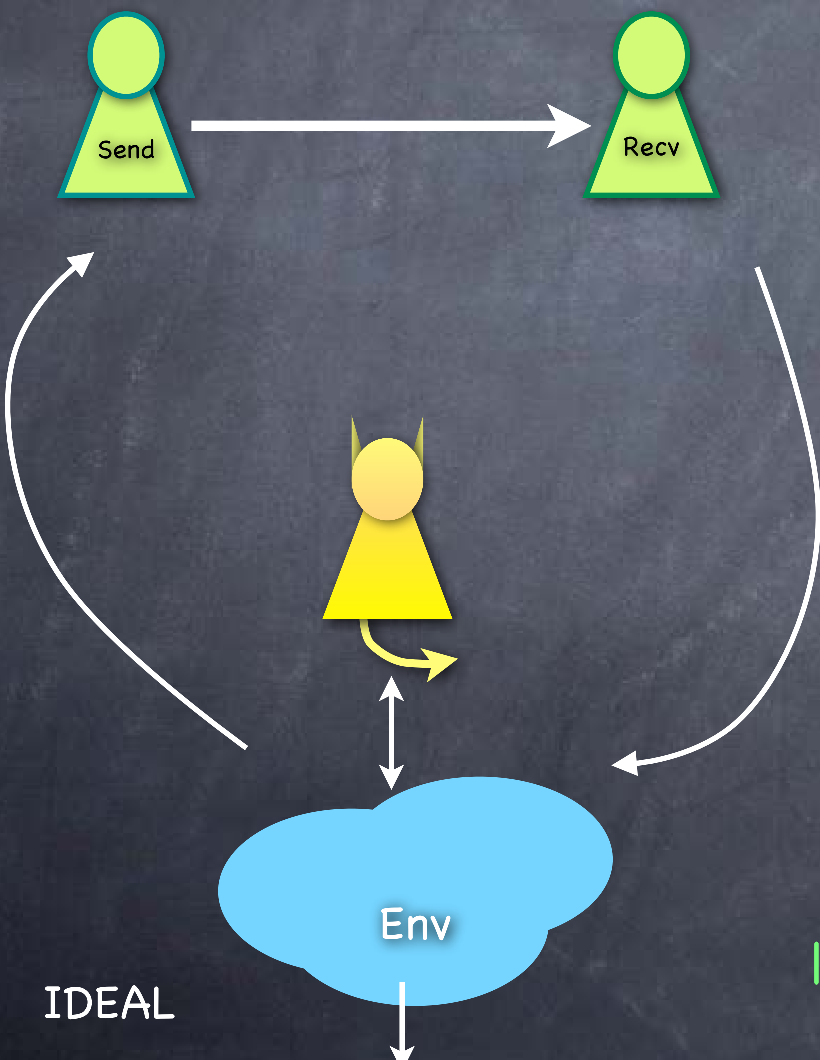
- A negligible quantity: As we turn the knob the quantity should "decrease extremely fast"
  - Negligible: decreases as 1/superpoly(k)
    - i.e., faster than 1/poly(k) for every polynomial
    - e.g.: 2^{-k}, $2^{-\sqrt{k}}$, $k^{-(\log k)}$.
    - Formally: T negligible if $\forall c>0\ \exists k_0\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

Interpreting Asymptotics



Symmetric-Key Encryption

SIM-CPA Security



SIM-CPA secure if:

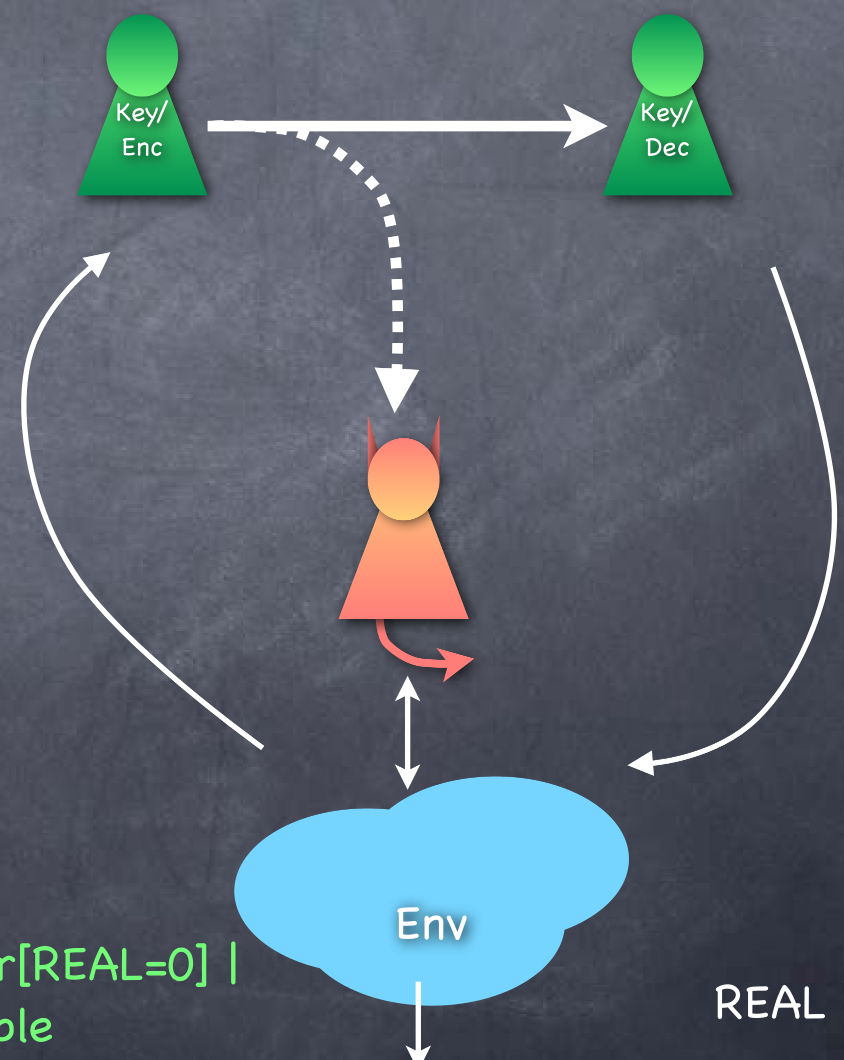
$\forall$ PPT

$\exists$ PPT s.t.

$\forall$ PPT

IDEAL $\approx$ REAL

| Pr[IDEAL=0] - Pr[REAL=0] | is negligible



Symmetric-Key Encryption

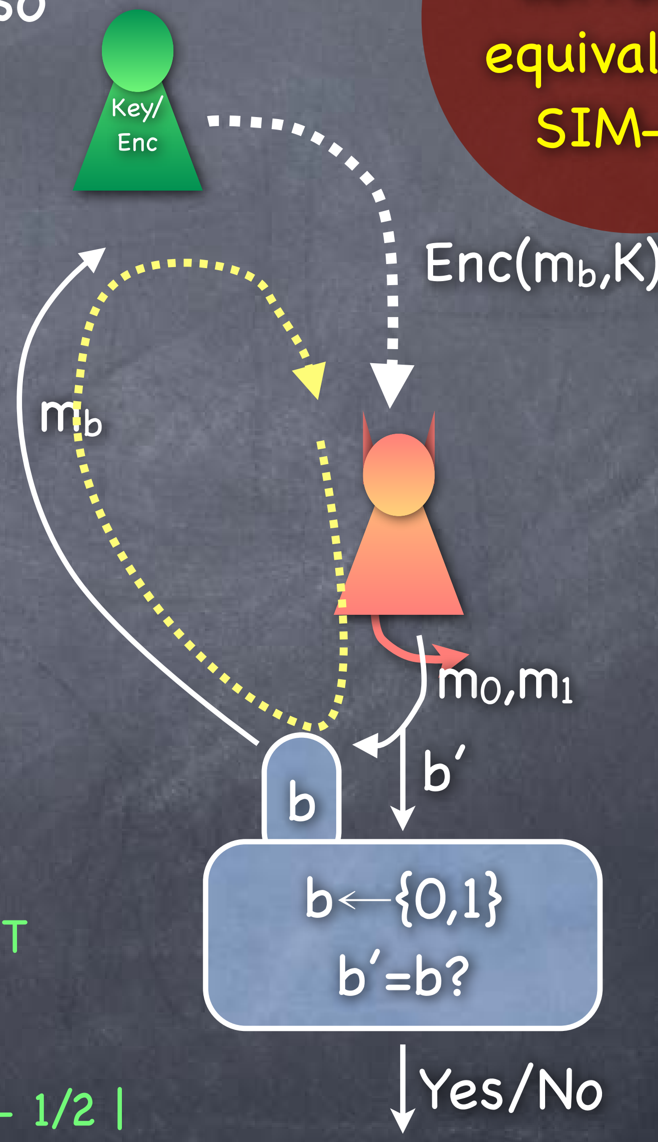
IND-CPA Security

- 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 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] ≈ 1/2

PPT

| Pr[b′=b] - 1/2 | is negligible

IND-CPA + ~correctness equivalent to SIM-CPA



Next

- Constructing (CPA-secure) SKE schemes
  - From a Pseudorandom Function (PRF)
  - And how to construct PRFs?
    - In theory, from a Pseudorandomness Generator (PRG), in turn constructed from any One-Way Function (or more easily from One-Way Permutations)
    - In practice, “block-ciphers”