

# Introduction to Cryptology

## Lecture 7

# Announcements

- HW3 up on course webpage, due at the beginning of class on 2/24
  - Please hand in code for Problem 2
  - Clarification of Problem 1, Part II
    - Encryption and Decryption are performed modulo 9

# Agenda

- Last time:
  - Finished up perfect secrecy (2.3, 2.4)
  - Started computational approach (3.1)
- Today:
  - More on the computational approach (3.1)
  - Defining computationally secure SKE (3.2)
  - Constructing computationally secure SKE (3.3)

# The Computational Approach

Two main relaxations:

1. Security is only guaranteed against efficient adversaries that run for some feasible amount of time.
2. Adversaries can potentially succeed with some very small probability.

# Security Parameter

- Integer valued security parameter denoted by  $n$  that parameterizes both the cryptographic schemes as well as all involved parties.
- When honest parties initialize a scheme, they choose some value  $n$  for the security parameter.
- Can think of security parameter as corresponding to the length of the key.
- Security parameter is assumed to be known to any adversary attacking the scheme.
- View run time of the adversary and its success probability as functions of the security parameter.

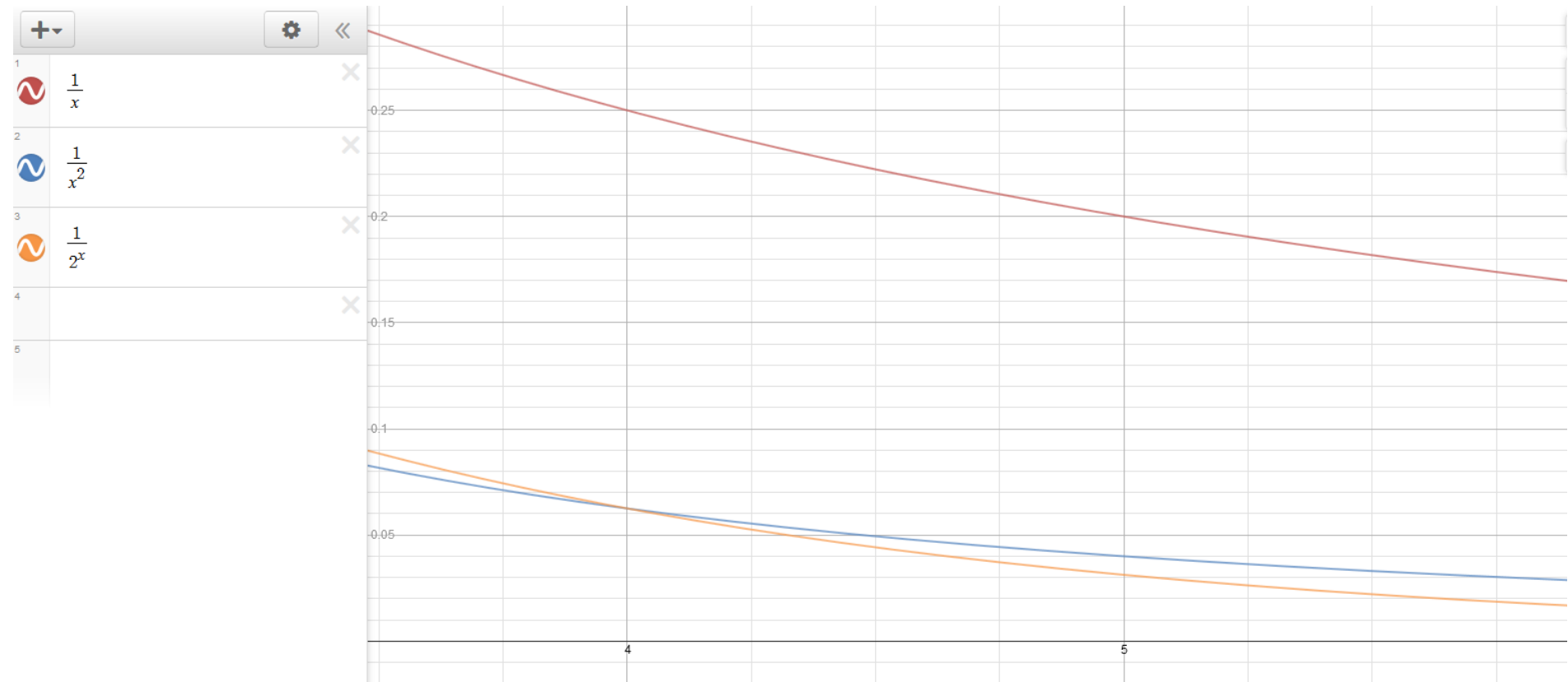
# Polynomial Time

- Efficient adversaries = Polynomial time adversaries
  - There is some polynomial  $p$  such that the adversary runs for time at most  $p(n)$  when the security parameter is  $n$ .
  - Honest parties also run in polynomial time.
  - The adversary may be much more powerful than the honest parties.

# Negligible

- Small probability of success = negligible probability
  - A function  $f$  is negligible if for every polynomial  $p$  and all sufficiently large values of  $n$  it holds that
$$f(n) < \frac{1}{p(n)}.$$
  - Intuition,  $f(n) < n^{-c}$  for every constant  $c$ , as  $n$  goes to infinity.

# Negligible





# Practical Implications of Computational Security

- For key size  $n$ , any adversary running in time  $2^{n/2}$  breaks the scheme with probability  $1/2^{n/2}$ .
- Meanwhile, *Gen*, *Enc*, *Dec* each take time  $n^2$ .
- If  $n = 128$  then:
  - *Gen*, *Enc*, *Dec* take time 16,384
  - Adversarial run time is  $2^{64} \approx 10^{18}$
- If  $n = 256$  then:
  - *Gen*, *Enc*, *Dec* quadruples--takes time 65,536
  - Adversary run time is multiplied by  $2^{64}$ . Becomes  $2^{128} \approx 10^{38}$

# Defining Computationally Secure Encryption

A **private-key encryption scheme** is a tuple of probabilistic polynomial-time algorithms  $(Gen, Enc, Dec)$  such that:

1. The **key-generation algorithm**  $Gen$  takes as input security parameter  $1^n$  and outputs a key  $k$  denoted  $k \leftarrow Gen(1^n)$ . We assume WLOG that  $|k| \geq n$ .
2. The encryption algorithm  $Enc$  takes as input a key  $k$  and a message  $m \in \{0,1\}^*$ , and outputs a ciphertext  $c$  denoted  $c \leftarrow Enc_k(m)$ .
3. The decryption algorithm  $Dec$  takes as input a key  $k$  and ciphertext  $c$  and outputs a message  $m$  denoted by  $m := Dec_k(c)$ .

Correctness: For every  $n$ , every key  $k \leftarrow Gen(1^n)$ , and every  $m \in \{0,1\}^*$ , it holds that  $Dec_k(Enc_k(m)) = m$ .

# Indistinguishability in the presence of an eavesdropper

Consider a private-key encryption scheme  $\Pi = (Gen, Enc, Dec)$ , any adversary  $A$ , and any value  $n$  for the security parameter.

The eavesdropping indistinguishability experiment  $PrivK^{eav}_{A,\Pi}(n)$ :

1. The adversary  $A$  is given input  $1^n$ , and outputs a pair of messages  $m_0, m_1$  of the same length.
2. A key  $k$  is generated by running  $Gen(1^n)$ , and a random bit  $b \leftarrow \{0,1\}$  is chosen. A challenge ciphertext  $c \leftarrow Enc_k(m_b)$  is computed and given to  $A$ .
3. Adversary  $A$  outputs a bit  $b'$ .
4. The output of the experiment is defined to be 1 if  $b' = b$ , and 0 otherwise. If  $PrivK^{eav}_{A,\Pi}(n) = 1$ , we say that  $A$  succeeded.

# Indistinguishability in the presence of an eavesdropper

Definition: A private key encryption scheme  $\Pi = (Gen, Enc, Dec)$  has **indistinguishable encryptions in the presence of an eavesdropper** if for all probabilistic polynomial-time adversaries  $A$  there exists a negligible function  $negl$  such that

$$\Pr \left[ PrivK^{eav}_{A, \Pi}(n) = 1 \right] \leq \frac{1}{2} + negl(n),$$

Where the prob. is taken over the random coins used by  $A$ , as well as the random coins used in the experiment.

# Semantic Security

- The full definition of semantic security is even more general.
- Consider arbitrary distributions over plaintext messages and arbitrary external information about the plaintext.

# Semantic Security

Definition: A private key encryption scheme  $\Pi = (Gen, Enc, Dec)$  is **semantically secure in the presence of an eavesdropper** if for every ppt adversary  $A$  there exists a ppt algorithm  $A'$  such that for all efficiently sampleable distributions  $X = (X_1, \dots)$  and all poly time computable functions  $f, h$ , there exists a negligible function  $negl$  such that

$$|\Pr[A(1^n, Enc_k(m), h(m)) = f(m)] - \Pr[A'(1^n, h(m)) = f(m)]| \leq negl(n),$$

where  $m$  is chosen according to distribution  $X_n$ , and the probabilities are taken over choice of  $m$  and the key  $k$ , and any random coins used by  $A, A'$ , and the encryption process.

# Equivalence of Definitions

Theorem: A private-key encryption scheme has indistinguishable encryptions in the presence of an eavesdropper if and only if it is semantically secure in the presence of an eavesdropper.

# Constructions



# Pseudorandom Generator

- Functionality
  - Deterministic algorithm  $G$
  - Takes as input a short random seed  $s$
  - Outputs a long string  $G(s)$
- Security
  - No efficient algorithm can “distinguish”  $G(s)$  from a truly random string  $r$ .
  - i.e. passes all “statistical tests.”
- Intuition:
  - Stretches a small amount of true randomness to a larger amount of pseudorandomness.
- Why is this useful?
  - We will see that pseudorandom generators will allow us to beat the Shannon bound of  $|K| \geq |M|$ .
  - I.e. we will build a computationally secure encryption scheme with  $|K| < |M|$

# Pseudorandom Generators

Definition: Let  $\ell(\cdot)$  be a polynomial and let  $G$  be a deterministic poly-time algorithm such that for any input  $s \in \{0,1\}^n$ , algorithm  $G$  outputs a string of length  $\ell(n)$ . We say that  $G$  is a **pseudorandom generator** if the following two conditions hold:

1. (Expansion:) For every  $n$  it holds that  $\ell(n) > n$ .
2. (Pseudorandomness:) For all ppt distinguishers  $D$ , there exists a negligible function  $negl$  such that:

$$|\Pr[D(r) = 1] - \Pr[D(G(s)) = 1]| \leq negl(n),$$

where  $r$  is chosen uniformly at random from  $\{0,1\}^{\ell(n)}$ , the **seed**  $s$  is chosen uniformly at random from  $\{0,1\}^n$ , and the probabilities are taken over the random coins used by  $D$  and the choice of  $r$  and  $s$ .

The function  $\ell(\cdot)$  is called the **expansion factor** of  $G$ .