Today

- Review Last time
- Computational Adversary
- Complexity theory review
- One-Way functions and complexity assumptions
- Weak vs. Strong One-Way functions
- Universal One-Way functions

Review from last time

We say that $(G,E,D)$ satisfies
- **perfect secrecy over** $M$ if $\forall m_1, m_2 \in M, \forall c$: $\Pr(E(K,m_1)=c) = \Pr(E(K,m_2)=c)$ and
- **perfect secrecy** if it satisfies perfect secrecy over all $M$.

- probabilities taken over choices of key $K$ by generating algorithm $G$, and
- Coin tosses of $E$ if its probabilistic

Review from Last time

We say that $(G,E,D)$ satisfies **Shannon secrecy** if $\forall$ probability distributions over $M$, $\forall m$, and $\forall c$

$\Pr(M=m) = \Pr(M=m \mid E(K,M)=c)$

- (the first probability is over the choice of the message and the second over the choice of the message, the key and the coin tosses of $E$ in case it is randomized)
Review from Last Time

• Theorem 1: Let \( (G,E,D) \) be an encryption scheme. Then, it satisfies Shannon's secrecy iff it satisfies perfect secrecy.

• Idea: Simple repeated use of Bayes theorem.

Review from Last Time

Claim 2: One Time Pad encryption scheme satisfies perfect secrecy (for sending 1 message).

Main observation of proof:
Let \(|m|=l\). Then, \( \forall c \)
\[
\Pr(E(K,M)=c) = \frac{1}{2^l} \quad \text{(taken over } K \text{ and } M) \\
\Pr(E(K,m)=c \mid M=m) = \frac{1}{2^l} \quad \text{(taken over } K) .
\]

In other words:
random variable \( C \) (that takes on the value of the ciphertext) is independent of random variable \( M \) (that takes on the value of the message).

New Material

• Theorem 3: If \( (G,E,D) \) satisfies perfect secrecy over \( M \), then \(|K| \geq |M|\)
  - Since the number of key is greater than the number of possible cleartexts, this also implies that the size of the key must be at least as large as the size of the cleartext that will be communicated using this key.
  - In particular, it means that using the one-time-pad method with the same key for more than one message results in a scheme which is NOT perfectly secret.

Optimality of the One Time Pad

• Using the one-time-pad for \( t, l\)-bit messages as follows:
  - choose \( k_i \in \{0,1\}^l \) at random for \( i=1,\ldots,t \)
  - For each new cleartext \( m_i \), let \( c_i = m_i \oplus k_i \)

• Disadvantages
  - The size of the key is huge: as much key bits as message bits
  - Receiver needs to know which key goes with which ciphertext (some synchronization or state)

• Advantage
  - By Theorem 3, this is BEST POSSIBLE, key can get no shorter.
Modern Cryptography

• 1976, New Directions in Cryptography

Diffie and Hellman

Computational Complexity based Theory of Cryptography

• New Computational Notions
  Perfect Secrecy       Computational Secrecy
  Gives no Information  Zero Knowledge
  Randomness           Pseudo Randomness
  Proof                Probabilistically-
                      Verifiable Proof

• Enable New Possibilities

Computationally Bounded Adversary

Adversary: Computationally bounded
  Possible = feasible (fast)
  Impossible = infeasible (too slow)

Feasible: runs in polynomial time and uses polynomial resources such as space, processors, randomness.

New Possibilities

- Encryption of many messages with fixed size keys (i.e. polynomial number of messages)
- Deterministic pseudo random number generation
- Public Key Cryptography (i.e. A and B never have to meet to agree on a secret key)
- Digital Signatures
- Zero Knowledge Interactive Proofs
- Secure Function Evaluation (or Games Over the Telephone without referees)
  • Privacy Preserving Data Mining
The Setting

All algorithms $G$, $E$, $D$, $EVE$:

- **Randomized polynomial time algorithms**
  - Algorithm $A$ is Polynomial time: $\exists$ a polynomial $Q$, $\forall$ input $x$, $A(x)$ terminates in polynomial time
  - Algorithm $A$ is randomized: can flip fair coins
    - *Las Vegas*: for decision problems, $\forall$ input always correct, runs in expected polynomial time
    - *Monte Carlo*: for decision problems, $\forall$ inputs correct with high probability over coin tosses of the algorithm, always runs in polynomial time

- **Polynomial in what?**
  - In our setting, everything is polynomial in the Security parameter $n$.
  - Security parameter = size of the secret key

- **Notation**: PPT-algorithm

Accordingly: Polynomially Secure Encryption

Encryption scheme $(G,E,D)$ where $G,E,D$ are PPT algorithms is **Polynomially Secure** if

- $\forall$ PPT algorithms $EVE$,
  $\left| \Pr(EVE (C) = m_1 \mid C= E(K,m_1))) - \Pr (EVE (C) = m_1 \mid C= E(K,m_2)) \right| < \text{neg} (n)$
- Where $m_1$ and $m_2$ are chosen by $EVE$ and
- the probabilities are taken over the choice of $K$ by generating algorithm $G(1^n)$, and the coin tosses of $EVE$ and $E$.
- Security parameter $n = |K|$

To be precise: Polynomially Secure Encryption

Encryption scheme $(G,E,D)$ where $G,E,D$ are PPT algorithms is **Polynomially Secure** if

- $\forall$ PPT algorithms $EVE$,
  $\left| \Pr(EVE (1^n,C) = m_1 \mid C= E(K,m_1))) - \Pr (EVE (1^n,C) = m_1 \mid C= E(K,m_2)) \right| < \text{neg} (n)$
- Where $m_1$ and $m_2$ are chosen by $EVE$ and
- the probabilities are taken over the choice of $K$ by generating algorithm $G(1^n)$, and the coin tosses of $EVE$ and $E$. 
Negligible Functions

- We say that a function $f$ is **negligible** if for all $n \geq n_0$, $f(n) < 1/P(n)$
- E.g. $f(n) = 1/n^{\log n}$, $1/2^n$

Differences with Perfect Secrecy

- Adversary EVE can't tell apart Cipher texts of $m_1$ and $m_2$
- More than a negligible fraction of the time

Polynomially Secure Encryption for many messages

$(G,E,D)$ is an encryption scheme which is **Polynomially Secure** if
- $G,E$ and $D$ are PPT algorithms
- $\forall$ PPT algorithms $EVE, \forall$ polynomial $P$
  - $|Pr(EVE(1^n, E(K,m_1), E(K,m_2),..., E(K,m_{P(n)})) = 1) - Pr(EVE(1^n, E(K,m'_1), E(K,m'_2),..., E(K,m'_{P(n)})) = 1)| < \neg(n)$
  - where $m_1,...,m_{P(n)}$ and $m'_1,...,m'_{P(n)}$ are chosen by $EVE$
  - the probabilities are taken over the choice of $K$ by generating algorithm $G(1^n)$, and the coin tosses of $EVE$ and $E$.

Do Polynomially Secure Encryption Schemes Exist

- Is a PPT-algorithm any different than Shannon's adversary?
One Way Function (OWF)

Say that $f: \{0,1\}^* \rightarrow \{0,1\}^*$ is a one-way function

- **Easy to Evaluate**: There exists a PPT algorithm $A$ s.t. $A(x) = f(x)$ for all $x$.
- **Hard to Invert**: For all PPT algorithms $B$, for all $n$ sufficiently large,
  $$\Pr (B(f(x)) = x' \mid f(x) = f(x')) < \negl(n)$$
  $x \in \{0,1\}^n$
coins of $B$

Remarks on Definition

1. Will refer to $n$ as the security parameter.
2. Typical asymptotic definition: asked to hold when $n \rightarrow \infty$ but may not be true for small sizes, e.g. $n = 512$.
3. $B$ succeeds if finds any inverse of $f(x)$.
4. Probabilistic guarantee: negligible probability over the inputs and coin tosses of algorithm.
5. Exact Security: Any $B$ that takes $T$ steps has probability $\alpha$ to invert.

OWF implies Secure Encryption

- We will show later in this class that
- OWF $\Rightarrow$ existence of cryptographically strong pseudo random number generators (CPSRG)
- CPSRG $\Rightarrow$ the existence of secure encryption

- Necessary and sufficient condition

Do One Way Functions exist

- We do **not** know, but we have candidates
- In this course, we shall **assume** OWF exist and **prove** secure encryption schemes exist based on this assumption
- Proof takes the form of an efficient reduction

Given any adversary Strategy to break the encryption scheme in time $T(k)$ with prob. $\alpha$
Construct an algorithm to Invert the one-way function in time $T' = \text{poly} (T(k))$ with prob. $\alpha/\text{poly} (k)$
How Strong is this assumption

• Let’s Relate it to our knowledge of complexity theory
• Let’s review some basic complexity theory

Basic Complexity Theory

• Complexity Classes
  - **P**: all problems solved in polynomial time
  - **NP**: all problems for which it is possible to verify the correctness of a solution in polynomial time
  - **BPP**: all problems solved by PPT time algorithms
  - **P/poly**: all problems solved by polynomial time non-uniform algorithms (algorithms whose size can grow as a function of the size of the input)

Complexity state of the world as we know it

• \( P \subseteq NP \)
• \( BPP \subseteq NP \)
• \( P \subseteq BPP \)

• Currently do not know how to prove a non-linear lower bound for one NP-hard problem (considered to be the hardest problems in NP)

Goal: Provable One-Way Function

- Proving \( f \) is a OWF i.e. \( f(z) \) is
  - efficiently computable
  - hard to invert by ppt algorithm (for random \( f(z) \))
- Will not be easy: implies \( BPP \neq NP \)!
  - Inverting \( f \) is \( NP \) (\( z \) is a witness for \( y=f(z) \))
  - \( f ~ OWF \implies \)
    Any ppt algorithm fails to invert \( f \) on some input
- Is it equivalent?
  - Does \( BPP \neq NP \) imply that one-way functions exist?
  - Stated differently: can you base the existence of one-way functions on the hardness of an NP-hard problem?
Tomorrow’s talk by Adi Akavia in the CIS-seminar addresses this open problem: Can we base OWF on NP-hardness?

Talk: 10:30-12:00 at 32-G449 (Kiva)

**NP ≠ BPP vs. OWFs**

### NP ≠ BPP:
- Worst Case hardness, i.e., there exists instances which are hard for BPP algorithms but they may be rare

### One-Way Function f:
- Average case hard to invert f
- ∃ efficient algorithm that generates random instance-witness pairs: (f(z), z)

**Candidate OWF: multiplication**

- Consider f(x, y) = xy where |xy| = n
- For most of the inputs of length n it is easy to invert
- Still: there are some inputs for which it is hard to find their divisors

**Number Theory**

**Hard Computational Problem: Factoring Integers**

products of two primes

*Given integer N = 1007350487*
*Find its prime divisors 13367 x 75361*

*Best Algorithm known: Sub-exponential time*

*What fraction is this of Total inputs of length n? O(1/n^2)*

*Easy to multiply p.q N=pq
hard for average input?*
Weak One Way Functions

• Weak One Way Function: \exists polynomial Q s.t. such that \forall PPT algorithms A, \forall n sufficiently large, 
  \Pr( A(f(x)) \neq x’ such that f(x)=f(x’) ) \geq 1/Q(n) 
  probability is taken over x s.t. |x|=n and A’s coins

• Intuitively: there is some hard-core polynomial fraction set of instances on which it is always hard to invert f

Weak OWF = Strong OWF

• Theorem: one-way functions exist iff weak one-way functions exist
• Proof: in class

Problems used in Cryptography tend to be

• Not known to be NP-hard P nor in BPP, for example:
  • Factoring Integers
  • Graph Isomorphism
  • Approximating size of the shortest vector in a lattice to within sqrt(n) of the optimal for dimension n
• Can an NP-hard problem be hard on the average?

What to do till can prove a particular function is OWF

• Make theory depend on general OWF and not a particular one.

• Theorem: Can construct a universal one-way function
• Proof: read in Goldreich's book

• g is a Universal one-way function: if there exists an one-way function, then g is a one-way function.
Collections of One Way Functions

Definition: \( F = \{ f_i : D_i \rightarrow R_i \}_{i \in I} \) where \( I \) is a set of indices, and \( D_i \), \( R_i \) are finite sets.

- **Easy to Evaluate:** \( \exists \) PPT algorithm \( A \) s.t.
  \[ A(i, x) = f_i(x) \] for all \( i, x \)

- **Hard to Invert:** \( \forall \) PPT \( B \), \( \forall \) sufficiently large \( k \),
  \[ \text{prob} \left( B(i, f_i(x)) = x' \text{ s.t } f_i(x) = f_i(x') \right) < \text{neg}(k) \]
  \( |i|=n \), \( i \in I \), \( x \in D_i \), coins of \( B \)

- **Generation:** \( \exists \) PPT (in \( n \)) algorithm \( G \) that selects a random \( f_i \) in \( F \) such that \( |i| = n \).
  \( G \) takes as input \( n \), the security parameter.