Cryptography

Boran Erol

March 2024

1 One-Way Functions

The existence of OWFs is equivalent to the existence of all (non-trivial) private-key cryptography.

Notice that the probability of inversion is taken over an experiment in which y is generated by choosing a uniform element x in the domain rather than choosing y uniformly in the range of f. It is also taken over the random coins of the adversary.

The definition by Lindell and Katz in Intro to Modern Cryptography is great.

Definition 1. A function $f : \{0,1\} \to \{0,1\}$ is **one-way** if the following two conditions hold

It is unknown whether we can construct CRHFs from OWFs.

Definition 2. A function f that is length preserving and a one-way function is said to be a **one-way permutation**.

Notice that the existence of OWFs is much stronger than P $\neq$ NP. We're not assuming worst-case hardness, we're assuming average-case hardness.

1.1 Candidate One-Way Functions

1. Integer Factorization
2. Subset Sum
3. The Hardness of Discrete Log (OWP)

1.2 Hard-Core Predicates

Let f be a one-way function. Notice that the function $g(x_1, x_2) := (f(x_1), x_2)$ is also a OWF but leaks half of its input.

Definition 3. A function $hc : \{0,1\}^* \to \{0,1\}$ is a **hard-core predicate of a function** f if hc can be computed in polynomial time and for every probabilistic polynomial-time algorithm A

$$\Pr_{x \xleftarrow{\$} \{0,1\}^n} [A(1^n, f(x)) = hc(x)] \leq \frac{1}{2} + negl(n)$$

Notice that the above definition doesn't require f to be one-way. Moreover, it's easy to construct functions that are trivially hard-core predicates (see Katz Lindell 7.1), but these are useless.

After seeing this definition, one natural question to ask is the following: Is there a function hc that is a hard-core predicate for all one-way functions f. It turns out that the answer is negative.

Let's now examine a plausible looking hc and construct a OWF f to break it. Let hc be defined by XORing the bits of x.

Lemma 1.1. For any fixed hc, there's a one-way function f such that hc is not a hard-core predicate of f

Proof. □

Goldreich and Levin Theorem shows how to construct a hard-core predicate for every OWF.

1.3 PRGs from OWPs and OWFs

Definition 4. A deterministic polynomial time algorithm G is a PRG if there's a stretch function $l : \mathbb{N} \to \mathbb{N}$ (satisfying $\forall k : l(k) > k$) such that for all PPT's D, for all positive polynomials p, and for all sufficiently large k we have that

$$|\Pr[D(G(U_k)) = 1] - \Pr[D(U_{l(k)}) = 1]| \leq \frac{1}{p(k)}$$

where the probability is taken over U_k and the coin tosses of D.

PRGs can be used to reduce the randomness complexity of randomized algorithms without making a noticeable difference in their output.

Statistically close distributions are trivially computationally indistinguishable.

You can upgrade single-sample indistinguishability to multi(polynomial)-sample indistinguishability by using a hybrid argument where the neighboring distributions differ by a single sample.

We'll now show how to get a PRG from an OWP. It is well-known that an OWF suffices.

Theorem 1.2. Let f be an OWP and hc be a hard-core predicate of f. Then, $G(s) = f(s)||hc(s)$ is a PRG with expansion factor $l(n) = n + 1$.

Proof. □

Let's now try to build PRGs with arbitrary expansion factors.

Theorem 1.3. If there exists a PRG with expansion factor $l(n) = n + 1$, then for any polynomial f there exists a PRG with expansion factor $f(n)$.

1.4 PRFs from PRGs

Theorem 1.4. If there exists a PRG with expansion factor $l(n) = 2n$, then there exists a PRF.

Theorem 1.5. If there exists a PRF, then there exists a strong PRF.

1.5 OWF Exercises, Lindell and Katz

1.5.1 Exercise 1

Lemma 1.6. If there exists a OWF, there exists a OWF f such that $\forall n \in \mathbb{N} : f(0^n) = 0^n$.

Proof. Intuitively, this is because in the inversion experiment, we sample the all zeros string with negligible probability. □

1.5.2 Exercise 3

Lemma 1.7. The existence of OWFs imply the existence of OWPs.

Proof. □

1.5.3 Exercise 4

Lemma 1.8. Let (Gen, H) be a collision-resistant hash function, where H maps string of length $2n$ to strings of length n. Then, the function family $(Gen, Samp, H)$ is one-way, where $Samp$ is the trivial algorithm that samples a uniform string of length $2n$.

In particular, the existence of CRHFs imply the existence of OWFs.

Proof. □

1.5.4 Exercise 5

Let F be a length-preserving pseudorandom permutation.

1.5.5 Exercise 15

Just put append n keys together and put the functions one after each other.

1.5.6 Exercise 19

Just feed the challenge string to the adversary A and act accordingly. There'll be a noticeable difference.

2 Trapdoor OWPs

The following is based on Lecture 1, Katz UMD Cryptography course. Check the lecture notes there for an in-depth discussion.

Let's first not worry about security at all and give a syntactic definition of trapdoor permutation families.

A **trapdoor permutation family** is a tuple of algortihms ($\mathbf{Gen}, \mathbf{Sample}, \mathbf{Eval}, \mathbf{Invert}$) such that

1. $\mathbf{Gen}(1^k) = (i, td)$
2. $\mathbf{Sample}(1^k, i) = x$, where x is uniformly random in D_i.

   Intuitively, **Sample** let's us sample uniformly from D_i.
3. $\mathbf{Eval}(1^k, i, x) = y$

   For all i output by **Gen**, $Eval(1^k, i, \cdot)$ is a permutation.
4. $\mathbf{Invert}(1^k, td, y) = x$

We require **Eval** and **Invert** to be deterministic. Notice that we're feeding i to **Invert** through y implicitly.

Definition 5 (Correctness)**.** For all i output by **Gen** and x output by **Sample**, we have that

$$\mathbf{Invert}(1^k, td, \mathbf{Eval}(1^k, i, x)) = x$$

Notice that f^{-1} exists even without the trapdoor.

Invert just makes f^{-1} efficient using td. Intuitively, we'll require that computing f^{-1} is difficult without td.

Definition 6. A trapdoor permutation family ($\mathbf{Gen}, \mathbf{Sample}, \mathbf{Eval}, \mathbf{Invert}$) is **one-way** if for any PPT A:

2.1 RSA as a Trapdoor OWP

$\mathbf{Gen}(1^k)$

Sample two k bit primes p and q. Set $N = pq$.

Sample e from Z_N^* and set d to be the inverse of d, i.e. $ed = 1$.

Let $i = (N, e)$ and $td = (N, d)$.

Set $D_i = Z_N^*$.

$\mathbf{Sample}(1^k, i)$

Sample from Z_N^*.

$\mathbf{Eval}(1^k, (N, e), x) = x^e \pmod N$

$\mathbf{Invert}(1^k, (N, d), y) = y^d \pmod N$

3 Claw-Free Permutations

Definition 7. A group of three numbers (x, y, z) is a **claw** for permutations f_0 and f_1 if

$$f_0(x) = f_1(y) = z$$

A pair of permutations is said to be **claw-free** if there's no efficient algorithm for computing a claw.

Constructing perfectly hiding commitment schemes from claw-free permutations is easy. Here's how to do it.

The sender samples a uniformly random bit and sends $f_b(x)$ to the receiver. The receiver has no clue which permutation the sender user and the sender can't cheat since the permutations are claw-free.

Claw-free permutations are similar to CRHFs in the following sense.

Lemma 3.1. Claw-free permutations imply CRHFs.

Proof. □

4 Commitment Schemes

There are also instance-dependent commitment schemes. For example, a commitment scheme might be hiding for $x \in L$ and binding for $x \notin L$. Notice that this would be very useful in ZK proofs.

4.1 Pedersen Commitments

4.2 Commitment Schemes from CRHFs

4.3 Commitment Schemes in ROM

4.4 Commitment Schemes from OWFs

4.5 Coin Flipping from Bit Commitment Protocols

Here's how to get a secure coin flipping protocol from a bit commitment protocol.

Alice samples a random bit b_0 and sends a commitment of b_0 to Bob. Bob randomly samples b_1 and sends it to Alice. Alice then decommits to b_0. Both parties calculate $b_0 \oplus b_1$.

The fact that Alice can't cheat comes from the binding property of the commitment scheme. The fact that Bob can't cheat comes from the hiding property of the commitment scheme.

5 Collision-Resistant Hash Functions

See Chapter 5.6 in Lindell and Katz for some applications of CRHFs.

Lemma 5.1. Let (Gen, H) be a CRHF. Define $(Gen, \hat{H})$ by $\hat{H}^s(x) = H^s(H^s(x))$. $(Gen, \hat{H})$ is also a CRHF.

Proof. By contradiction, assume there's some PPT adversary A that finds x, x' such that $\hat{H}^s(x) = \hat{H}^s(x')$ with noticeable probability. Then, either

$$H^s(x) = H^s(x')$$

$$\hat{H}^s(x) = \hat{H}^s(x')$$

In each case, we break the collision resistance of H. We thus conclude the proof.

$\square$

6 PKE

6.1 CPA Security

Definition 8. A public key encryption scheme is a triple of algorithms (Gen, Enc, Dec) such that

1. $Gen(1^k, r) = (pk, sk)$
2. $Enc(m, pk) = c$
3. $Dec(c, sk) = m$

Definition 9 (Correctness)**.**

Definition 10 (Semantic Security)**.**

Definition 11 (IND-CPA Security)**.**

The challenger runs $\mathbf{Gen}(1^k, r)$ and sends pk to the adversary.

Then, the adversary picks two messages m_0, m_1 and sends them to the challenger.

The challenger flips a coin $b \overset{\$}{\leftarrow} \{0, 1\}$ and sends $Enc(m_b, sk)$.

The adversary wins if it can predict b.

Notice that the adversary can predict with probability $\frac{1}{2}$ trivially.

Also notice that our encryption algorithm needs to be randomized in order to satisfy IND-CPA security. Otherwise, tha adversary can encrypt the challenge strings on its own and beat the challenge trivially.

Also, by a hybrid argument, PKE schemes that are single-message secure are secure for polynomially many messages.

6.2 DDH and El-Gamal PKE

DDH was introduced in 1976, for which Diffie and Hellman got the Turing Award. El-Gamal PKE is almost identical to the Diffie-Hellman Key Exchange, but it took 9 years to discover. This is another example of a simple brilliant idea hiding under our nose.

Here's the decisional Diffie-Hellman assumption:

Definition 12. Let q be a prime and consider a group G with $|G| = q$. The following two distributions are computationally indistinguishable:

$$(x, y \overset{\$}{\leftarrow} G : g^x, g^y, g^{xy}) \approx_c (x, y, r \overset{\$}{\leftarrow} G : g^x, g^y, g^r)$$

Here's how to construct a key exchange from the DDH assumption:

Here's how to construct El-Gamal PKE from the DDH assumption:

6.3 PKE from Trapdoor OWPs

6.4 CCA-1 Security (Lunchtime Attacks)

Suppose an employee decrypts some important information about a company during lunch. Afterwards, the employee is fired. Is the encryption scheme still secure now that the adversary has access to plaintexts and ciphertexts of its choice?

Simply put, we'd like our PKE scheme to be secure **even if** the adversary gets access to the decryptions of polynomially many ciphertexts before the challenge.

Proving the security of CCA-1 secure schemes are often as hard as constructing the schemes. The reductions are pretty hard.

6.4.1 Naor-Yung Construction of CCA1 Secure PKE Schemes using Adaptively-Secure NIZKs

This was the first ever CCA-1 secure PKE scheme.

Resources

1. Jonathan Katz UMD Graduate Course Lecture 6,7
2. The original paper from 1990

For **Gen**, run two key generation algorithms from a CPA-secure PKE scheme. For encryption, encrypt the message using both schemes using different randomness. Then, give a NIZK proof that you encrypted the same message.

6.5 CCA-2 Security

In CCA-2 security, the adversary can get decryptions of ciphertext even after the challenge sends the challenge ciphertext c. Naturally, we don't allow the adversary to ask for decryptions of the two ciphertexts it received as a challenge.

Notice that CCA-2 security doesn't hold for homomorphic encryption schemes. The adversary can just decrypt twice the encrypted message and ask for the challenger to decrypt.

CCA-2 security is equivalent to non-malleability.

In 1998, NIST published a CCA-1 secure encryption scheme as the encryption standard and Daniel Bleichenbacher embarassed them. He has Asperger's and got extremely mad at Rafi because Rafi supported the Iraq war.

Cramer-Shoup encryption based on El-Gamal encryption is the classic CCA-2 secure encryption scheme.

6.5.1 Dwork, Dolev and Naor 1991 CCA-2 Secure Construction

Let the length of public key of the signature scheme be l.

Run $2l$ semantically secure PKE KeyGen algorithms. Let r be a random string.

Let h be a CRH.

7 Digital Signature Schemes

7.1 Resources

1. Chapter 12 in Katz and Lindell

7.2 Introduction and Motivation

Digital signature schemes are **publicly verifiable** and **transferable**.

They also have the property of **non-repudiation**.

Digital signature schemes are NOT inverse PKE schemes.

8 Common Reference String Model

Captures the assumption of a trusted setup where all parties have access to the same string crs (common reference string) taken from some distribution D.

9 Private Information Retrieval

Private information retrieval is a relaxation of OT where a user queries data from a server which has a database, and the server doesn't gain information about which data the user received. However, we remove the requirement that the user doesn't learn anything about the data it doesn't retrieve. PIR was introduced by Chor, Goldreich, Kushilevitz and Sudan in 1995.

Let's consider a game with a user U and a database DB. WLOG, assume the database is some string $x \in \{0,1\}^n$.

Notice that we can trivially achieve PIR by downloading the entire database. Therefore, the challenge in PIR is to reduce the communication complexity as much as possible. It's not a problem of feasibility.

Notice that even without privacy, the communication complexity is $O(\log n)$ where the user sends i in the open and the server sends x_i.

Theorem 9.1 (Cnor, Goldreich, Kushilevitz, Sudan 1995)**.** Information-theoretic PIR with a single server with communication complexity less than n bits is impossible.

Theorem 9.2 (Kushilevitz, Ostrovksy 1997)**.** It is possible for computational security.

9.1 Multi-server PIR

Here's a basic scheme for two servers. We assume that the two servers don't collude. View the DB $x \in \{0,1\}^n$ as an $\sqrt(n) \times \sqrt(n)$ matrix X. The client wants to read $(i,j) \in [\sqrt(n) \times \sqrt(n)]$.

Rest in Notability.

9.2 Single Server PIR

The idea is to encrypt the query instead of secret sharing it.

It uses linearly homomorphic encryption.

The security follows from the security of LHE.

The communication is now $2\sqrt(n)$ ciphertexts.

It is possible to get $polylog(n)$ communication.

9.3 PIR in Practice

To support longer DB records, we can just use PIR for 1-bit records l times. Therefore, it's as if we have l individual databases.