CSE 5852 – Modern Cryptography: Foundations - Fall 2016

Paramik Dasgupta
Department of Computer Science and Engineering
University of Connecticut,Storrs
Lecture 16

October 26, 2016

---

1 Last Class

Last class we defined a pseudorandom function and showed how to construct it using a pseudorandom generator.

Consider two experiments: $\mathtt{exp}-\mathtt{prf}^{f}$ and $\mathtt{exp}-\mathtt{r}$. Let $\mathcal{A}$ be some PPT algorithm that outputs either 1 or 0.

Experiment $\mathtt{exp}-\mathtt{prf}^{f,\mathcal{A}}$:
Select random s of length κ.
Repeat an arbitrary number of times:
Receive x_i from $\mathcal{A}$.
Give $y_i = f_s(x_i) = f(s, x_i)$ to $\mathcal{A}$.
When $\mathcal{A}$ outputs "finished" and a bit b, output b.

Experiment $\mathtt{exp}-\mathtt{r}^{\mathcal{A}}$:
Initialize an empty table of values.
Repeat an arbitrary number of times:
Receive x_i from $\mathcal{A}$.
Lookup x_i in the table of values
if it exists return y_i the stored value.
else randomly select y_i and
store (x_i, y_i) in the table.
When $\mathcal{A}$ outputs "finished" and a bit b, output b.

Construction 1. *[2] Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a PRG. Use $G_0(s)$ to denote the left half of G's output and $G_1(s)$ to denote the right half* *of G's output. Then the following function $f(s,x)$ is a PRF:* $f(s,x) = G_{x_n}(G_{x_{n-1}}(....G_{x_1}(s)))$.

Pseudorandom functions are sufficient to create a "secure channel" between two participants that share a key. There are some important things that need to be considered: key management, side-channel attacks, padding, modes of operations. These things are all important. We omit them from this class not because of importance but to explore other paradigms for cryptography.

2 Data Encryption Standard (DES)

1. Built by IBM
   2. Shown to NSA
      a). changed some constants (Differential Cryptanalysis)
      b). reduced key length
   3). Became standard in 80s and 90s
   4). Key space became exhaustible mid 90s

3 Advanced Encryption Standard(AES) [3]

1). Open competition by National Institute of Standards and Technology
   2). Started in 1998
   3). Key sizes 128, 192, 256 bits

The winners were Rijndael, i.e. Vincent Rijmen and Joun Daemen

The current best attacks run in 2^{126} for 128 bit key.

AES $\{0,1\}^{128}$ x $\{0,1\}^{128}$ $->$ $\{0,1\}^{128}$
k m

a = k $\oplus$ m (value a, which is sum of key and message)

a_1 a_2 a_3 a_4
a_5 a_6 a_7 a_8
a_9 a_{10} a_{11} a_{12}
a_{13} a_{14} a_{15} a_{16} $- - - - - - - - - - - ->$ (S-box substitution)

b_1 b_2 b_3 b_4
....
....
....
$- - - - - - - - - - - - - - - ->$ shift rows

b_1 b_2 b_3 b_4
b_6 b_7 b_8 b_5
b_{11} b_{12} b_9 b_{10}
b_{16} b_{13} b_{14} b_{15} $- - - - - - - - - - - - ->$ mix columns

linear transformation of each row
....

It works in hierarchical organizations, but does not work online and in large networks.

4 Public Key Cryptography

In the previous two months we've shown how to create a secure channel between two participants that share a key. We now want to ask what happens if they don't have that key. The first task we'll consider is something called key agreement. We want a sender and receiver to agree on a K.

There exists a sender, receiver, and in between them, there could be an attacker.

SENDER $- - - - - - - - - - - - - - ->$ RECEIVER
K K
$< - - - - - - - - - - - - - -$

K is pseudorandom having seen entire conversation.

5 Need to create asymmetry between sender/receiver and attacker

Our one problem : Discrete log

Assumption 1: For all PPTA, negligible ϵ(n), ...

Pr[A(p,g,g^x mod p) = x] $\leq$ ϵ(n) Here p and g are known to all.

Sender	Receiver	Attacker
p	p	p
g	g	g
g^x	g^x	g^x
x		g^y

What can the attacker compute ?

$g^x . g^y = g^{x+y}$
The sender can compute $(g^y)^x = g^{xy} = (g^x)^y$
This is the Diffie-Hellman protocol [1]

Let's ask if this can be easily attacked. What are actions we know how to do mod p

1. Exponentiate to arbitrary power
2. Multiply values (add in exponent)
3. Square roots
4. compute inverse
5. Take mod

None of these strategies make it immediately obvious that $\mathcal{A}$ can compute g^{xy}. Ideally, we would like to show that an adversary that can compute g^{xy} can be used to compute x or y. However, this is not known either. There is no know reduction from learning g^{xy} to the discrete logarithm assumption. This leaves us in the somewhat troubling place of having to introduce another assumption:

Claim 1. *If you can compute discrete log, Then DH is insecure.*

We actually need to create a new assumption.

Assumption 1. *[1][Computational Diffie-Hellman Assumption] For any PPT $\mathcal{A}$, there exists a negligible ϵ such that for a random n-bit p and its generator and select a random $x, y \in \mathbb{Z}_p^*$,*

$$\Pr[\mathcal{A}(1^n, p, g, g^x \mod p, g^y \mod p) = g^{xy}] \leq \epsilon(n).$$

Claim 2. *If the CDH problem is hard then so is Discrete log.*

This assumption says it will be unlikely for an attacker to be able to predict the value g^{xy} which we'd like to use as the key. As before this doesn't tell us anything about whether the adversary has some information about g^{xy}. They might know the first/last bit (as in the case of the pseudorandom generator. This leads us to yet another assumption.

Assumption 2. *[1][Decisional Diffie-Hellman Assumption] For any PPT $\mathcal{A}$, there exists a negligible ϵ such that for a random n-bit p and its generator and select a random $x, y, z \in \mathbb{Z}_p^*$,*

$$\Pr[\mathcal{A}(1^n, p, g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{A}(1^n, p, g, g^x, g^y, g^z) = 1] \leq \epsilon(n).$$

We noted above that Assumption 2 implies Assumption 1 (that is if we have an efficient algorithm to solve discrete log we also have an efficient algorithm to solve computational Diffie-Hellman). We'll now show that Assumption 3 implies Assumption 2.

Theorem 1. *If there exists PPT $\mathcal{A}$ that breaks the computational DH assumption with an inverse polynomial probability then there exists PPT $\mathcal{A}'$ that breaks the decisional DH assumption with an inverse polynomial probability. (That is, decisional DH implies computational DH.)*

5.1 Drawbacks of Diffie-Hellman

1. Interactive (Both sending messages)
2. g^x, g^y cannot be reused (at least this isn't clear).
3. Not secure against active attacker A (attacker -in-middle)

References

[1] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[2] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[3] Frederic P Miller, Agnes F Vandome, and John McBrewster. Advanced encryption standard. 2009.