A Theorist's Toolkit **(CMU 18-859T, Fall 2013)**

Lecture 23: Cryptography

November 25, 2013

Lecturer: Ryan O'Donnell *Scribe: Linus Hamilton*

1 Introduction

Alice wants to send a secret message m to Bob, but doesn't want the eavesdropper Eve to get any infomation about m. To do this, she encrypts the plaintext m to get the ciphertext $c = Enc(m)$. Then she sends c to Bob, who decrypts it to obtain the message $m = Dec(c)$.

Simple, right?

2 Symmetric-Key Cryptography

Of course, if Eve and Bob have equal information, then Eve can decode any message Bob can. To get around this, Alice meets with Bob beforehand to share a secret key SK. They agree to use SK in the encryption and decryption algorithms. Now the procedure looks like this:

- Alice and Bob secretly generate $SK = Gen()$, using the function Gen to generate a random secret key.
- Later, Alice wants to send the message m to Bob. She computes $c = Enc(m, SK)$ and sends it to Bob. Eve might eavesdrop on the ciphertext c.
- Bob decrypts $m = Dec(c, SK)$ and gets the original message back.

This kind of procedure is called an SKE scheme, or symmetric-key encryption scheme.

We'd like some measure of security, so that Alice and Bob can be sure Eve doesn't find out their secret. One reasonable security condition is that Eve should gain absolutely no information about m from seeing c.

Definition. An SKE scheme is *perfectly secure* (or "Shannon secure") if Eve gains no information about the plaintext m from seeing the ciphertext c; in other words, if the distribution $\{Enc(m, SK) : SK \leftarrow Gen()\}$ is the same for any message m.

Let's find an SKE scheme which is perfectly secure. For notation's sake, from now on, write U_n to be the uniform distribution on $\{0,1\}^n$.

Definition. The One-Time Pad encryption scheme lets Alice send an n-bit message to Bob, as follows. Gen generates a random SK from U_n. Then $Enc(m, SK) = m \oplus SK$, and $Dec(c, SK) = c \oplus SK$, where $\oplus$ denotes bitwise XOR.

Theorem. *The One-Time Pad scheme is perfectly secure.*

Proof. For any message m, the distribution $\{Enc(m, SK) : SK \leftarrow Gen()\}$ is the uniform distribution U_n. □

Great! So why don't we just use the One-Time Pad everywhere and call it a day? Well, unfortunately, the OTP has some problems. For instance, like the name indicates, you can only use an OTP once. If Eve gets her hands on both $m_0 \oplus SK$ and $m_1 \oplus SK$, she now knows $m_0 \oplus m_1$ with certainty, which can help her break the code. (In practice, if m_0 and m_1 are English sentences, and you know $m_0 \oplus m_1$, it's almost always trivial to work out both m_0 and m_1.)

Another problem is that the key in an OTP has to be as long as the message. It's pretty easy to see that *any* perfectly secure cipher must have this property. So much for perfect security. In light of these problems, cryptologists usually use a relaxed notion of security.

Computational Indistinguishability

For perfect security, we required that for any two possible messages m_0 and m_1, the distributions of their possible ciphertexts are statistically indistinguishable. For computational security, we require only that the distributions be *computationally indistinguishable*. Intuitively, we assume that Alice, Bob, and Eve are PPT (i.e. they can only run probabilistic polynomial time algorithms). For an encryption scheme to be secure, Eve should only figure out Alice's message with negligible probability (e.g. if she gets extremely lucky and happens to guess the secret key).

Definition. A function is *negligible* if it is less than $\frac{1}{n^c}$ for every constant c. In the vein of big-O notation, we write $negl(n) = \frac{1}{n^{\omega(1)}}$.

Definition. Let $X_n = \{X_{n_1}, X_{n_2}, \ldots, X_{n_m}\}$ and $Y_n = \{Y_{n_1}, Y_{n_2}, \ldots, Y_{n_k}\}$ be *ensembles* on $\{0,1\}^{\ell(n)}$, i.e. sequences of random variables taking values in $\{0,1\}^{\ell(n)}$. X_n and Y_n are *computationally indistinguishable* if no algorithm can reliably tell them apart. That is, for every PPT algorithm A,

$$|\,\mathbf{Pr}[A(X_n) \text{ returns true}] - \mathbf{Pr}[A(Y_n) \text{ returns true}]| \leq negl(n).$$

When X_n and Y_n are computationally indistinguishable, write $X_n \overset{c}{\approx} Y_n$.

Fact 1. *$\overset{c}{\approx}$ is transitive, even if you chain polynomially many of them together. That is, if $X_1, \ldots, X_{poly(n)}$ are ensembles on $\{0,1\}^n$, and $X_i \overset{c}{\approx} X_{i+1}$ for all i, then $X_1 \overset{c}{\approx} X_{poly(n)}$.*

Proof. Use the triangle inequality. For every PPT algorithm A,

$$\begin{aligned}
&|\,\mathbf{Pr}[A(X_1) \text{ returns true}] - \mathbf{Pr}[A(X_{poly(n)}) \text{ returns true}]| \\
&\leq \sum_i |\,\mathbf{Pr}[A(X_i) \text{ returns true}] - \mathbf{Pr}[A(X_{i+1} \text{ returns true})] \\
&\leq \sum_i negl(n) = poly(n) \cdot negl(n) \\
&= negl(n).
\end{aligned}$$

□

Fact 2. *If $X_n \overset{c}{\approx} Y_n$, then for any PPT function B, $B(X_n) \overset{c}{\approx} B(Y_n)$.*

Proof. If $B(X_n) \overset{c}{\not\approx} B(Y_n)$, then Eve can distinguish X_n from Y_n by applying B to both. □

Using these definitions, we can formulate a definition of cryptographically secure.

Definition. An SKE scheme (Gen, Enc, Dec) is *single-message secure* if for any two messages m_0 and m_1, with $SK \leftarrow Gen()$, we have $Enc(m_0, SK) \overset{c}{\approx} Enc(m_1, SK)$. (Here $Enc(m_0, SK)$ is understood to be a single-element ensemble.)

In the next section, we'll see how to use cryptographic PRGs to make secure SKE schemes.

Cryptographic PRGs

Definition. A *cryptographic PRG* is a deterministic polytime-computable function $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ such that $G(U_n) \overset{c}{\approx} U_{\ell(n)}$.

A cryptographic PRGs is very similar to a normal PRGs (from the lecture on derandomization). You could say that a cryptographic PRG is a normal PRG that $negl(n)$-fools the class of PPT-computable functions.

For the purposes of cryptography, it is useful to make the following assumption:

Conjecture. *Cryptographic PRGs G with $\ell(n) = n + 1$ exist.*

Remark. This conjecture implies $\mathsf{P} \neq \mathsf{NP}$. Proof sketch: If $\mathsf{P} = \mathsf{NP}$, then we can check whether any string $s \in \{0,1\}^{n+1}$ is one of G's possible outputs in polynomial time. In that case, the algorithm $A(s) =$ [true if $s \in image(G)$, else false] distinguishes $G(U_n)$ from U_{n+1}, so G isn't a cryptographic PRG after all.

Since the above conjecture implies $\mathsf{P} \neq \mathsf{NP}$, we aren't going to prove it anytime soon.

In the conjecture, $\ell(n) = n + 1$ may seem like a weak assumption. But it is actually enough, due to the following theorem.

Theorem. *If there are crypto PRGs with $\ell(n) = n + 1$, then there are crypto PRGs with $\ell(n) = poly(n)$ for any desired polynomial.*

Proof. For all n, let $G_n : \{0,1\}^n \to \{0,1\}^{n+1}$ be a crypto PRG with input length n.

We will construct a procedure for a crypto PRG G' with $\ell(n) = poly(n)$. The idea is just to use G over and over.

Given an input $x \in \{0,1\}^n$, let $G'(x) = G_{poly(n)-1} \circ G_{poly(n)-2} \circ \cdots \circ G_{n+1} \circ G_n(x)$.

To prove this works, we repeatedly apply Facts 1 and 2, above.

We have $G_n(x) \overset{c}{\approx} U_{n+1}$

So $G_{n+1}(G_n(x)) \overset{c}{\approx} G_{n+1}(U_{n+1}) \overset{c}{\approx} U_{n+2}$

So $G_{n+2}(G_{n+1}(G_n(x))) \overset{c}{\approx} G_{n+2}(U_{n+2}) \overset{c}{\approx} U_{n+3}$

Et cetera. □

Before we continue with the abstract theory, here's an example of a function believed to be a crypto PRG.

Blum-Blum-Shub: Frst pick N, the product of two 3 (mod 4) primes. Now, given input X, repeatedly set $X \leftarrow X^2 (\text{mod } N)$ and output the least significant bit of X. The stream of outputs is believed to be a crypto PRG.[5]

Now, we'll show how to generate a single-message computationally secure SKE scheme using any crypto PRG.

Theorem. *Let G be a crypto PRG. Set $Gen() = U_n$, $Enc(m, SK) = m \oplus G(SK)$, $Dec(c, SK) = c \oplus G(SK)$. Then (Gen, Enc, Dec) is single-message secure.*

Proof. Just do it: for any message m, $Enc(m, SK) = m \oplus G(U_n) \overset{c}{\approx} m \oplus U_{\ell(n)} = U_{\ell(n)}$. $\square$

This works fine for single-message security, but what if Alice wants to send more than one message? Then the above scheme has the same problem that using a One-Time Pad twice does. Worse still, let's say Eve is allowed to execute *known-plaintext attacks:* she has a set of previous coded messages sent by Alice, along with their decoded plaintexts. This may sound unreasonable, but it's not. If Alice and Bob attack Eve one day at dawn, she can deduce that their previous message decoded to "ATTACK AT DAWN."

Nevertheless, under some plausible assumptions, Alice and Bob can still communicate securely.

Theorem. *[Hill '99] Even if Eve can execute known-plaintext attacks, Alice and Bob can still communicate securely if pseudorandom function families (PRFs) exist.*

Intuitively, think of a PRF as follows. A PRF is a family of functions $\{0,1\}^n \to \{0,1\}$. It should be pseudorandom, in the following sense. If we give Eve a function f sampled randomly from the PRF, along with a random oracle $O : \{0,1\}^n \to \{0,1\}$, then she should be unable to figure out which one is which, except with negligible probability.

Goldreich, Goldwasser, and Micali showed how to create a PRF from a PRG.[3] We know the following chain of implications:

$$\begin{aligned} \text{One–way functions exist} &\iff \text{Crypto PRGs exist} \\ &\iff \text{Crypto PRFs exist} \\ &\implies \text{Symmetric–key encryption is possible} \end{aligned}$$

The first in the chain of implications, one-way functions, are basically functions that are hard to invert.

Definition. A *one-way function* is a function f which is polytime-computable, but for any PPT algorithm A, A almost never successfully inverts f. That is, given a random input x to f, $\mathbf{Pr}[f(A(f(x))) = f(x)]$ is negligible in the input size.

By the way, theorists are pretty sure that one-way functions exist. (Although it's still stronger than $\mathsf{P} \neq \mathsf{NP}$, so don't start F5ing the Wikipedia page.) We know this fact:

Fact. *If even a "weak one-way function" (i.e. a one-way function but with negligible probability replaced by $\frac{1}{poly(n)}$ probability) exists, then a full one-way function exists.*

The following is one candidate for a one-way function:

In the following function, $a_1, \ldots, a_n$ are integers mod 2^n, and S is a subset of $[n]$.
$f(a_1, a_2, \ldots, a_n, S) = (a_1, a_2, \ldots, a_n, \sum_{s \in S} a_s)$.
The statement "f is a one-way function" basically expresses the belief that subset-sum is hard on average.

3 Public Key Cryptography

Eve is getting frustrated. Alice and Bob are sending each other messages back and forth with wild abandon, using their darn secret key SK.

She resolves not to make this mistake again. She won't let her new enemies, Alice' and Bob', share a secret key. Since Alice' and Bob' live on opposite sides of the Earth, they don't stand a chance. Eve intercepts everything they say, and as a result, they can never share a single secret.

Three weeks later, Eve opens her communication log to find that Alice' has sent Bob' a paper on lattice methods in public-key cryptography. She throws her keyboard against the wall — all is lost.

In public-key cryptography, Alice wants to send Bob a secret message, but they can't meet up beforehand to share a key. Amazingly, even if Eve has access to their entire record of conversation, it is still possible for them to send messages securely. A typical protocol proceeds like so:

• Bob runs $Gen()$ to generate PK and SK, a public key and a secret key respectively. He publishes PK and keeps SK to himself.

• When Alice wants to send Bob a message m, she encrypts it to $c = Enc(m, PK)$ and sends it to Bob.

• Bob decrypts $m = Dec(c, SK)$ with his secret key.

Theorem. *If Alice can send Bob one-bit messages securely with public-key cryptography (i.e. $(PK, Enc(\text{"0"}, PK)) \stackrel{c}{\approx} (PK, Enc(\text{"1"}, PK))$, then she can send arbitrary messages securely.*

Just like one-way functions and SKE cryptography, most theorists believe that public-key cryptography is possible. In other words, out of Impagliazzo's five possible worlds[4], most people believe that we live in Cryptomania.

As witness, RSA and Diffie–Hellman. In recent years, lattice methods have also become popular. Here's an example of a lattice-based cryptosystem, based on the Learning With Errors (LWE) problem.

Fix n, q (usually $\sim n^2$ or n^3), and α (usually $\sim \frac{1}{\sqrt{n}\log n}$).

Bob picks a secret $(s_1, \ldots, s_n) \in \mathbb{Z}_q^n$. He publishes a batch of about $O(n \log n)$ equations of the form
"$a_1 s_1 + \cdots + a_n s_n \approx b$".
Here the a_i are random numbers mod q, the s_i are variables, and $b = (\sum a_i s_i) + error$. The error term is chosen from the Gaussian $\mathcal{N}(0, \alpha^2 q^2)$.

To send Bob a random bit, Alice first chooses a random subset of the equations and adds them together to get $A_1 s_1 + \cdots + A_n s_n \approx B$. If the bit is zero, she sends this equation to Bob. If the bit is 1, she instead sends$A_1 s_1 + \cdots + A_n s_n \approx B + \lfloor \frac{q}{2} \rfloor$.
To find out Alice's message, Bob checks whether the equation is true or far from true. We can see with a Chernoff bound that one of these will almost always be the case.

The key to this cryptosystem's security is that Eve, given the batch of equations, cannot work out the values of $s_1, \ldots, s_n$. Working out these values is called the Learning With Errors problem.

Theorem. *(Ajtai-Dwork, Regev '05) The Learning With Errors (LWE) problem is average-case hard, if the Shortest Vector Problem is hard to $n^{1.5}$-approximate in the worst case on a quantum computer.[1]*

Now that we know that quantum computers can break RSA and Diffie–Hellman easily, lattice-based cryptography and other non-number-theoretical methods are growing more popular. If you think the NSA has a quantum computer, then these are the cryptosystems for you.

References

[1] Miklos Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. *ACM STOC*, 1997.

[2] Yevgeniy Dodis. Introduction to Cryptography, 2012.

[3] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions, 1985.

[4] Russell Impagliazzo. A Personal View of Average-Case Complexity, 1995.

[5] Pascal Junod. Cryptographic Secure Pseudo-Random Bits Generation : The Blum-Blum-Shub Generator, 1999.