CS276 (Spring 2006) Cryptography **Lecture: 27**

Cryptographic “Complexity Classes”

Lecturer: David Wagner *Scribe: Dominic Antonelli*

1 Introduction

In previous lectures, we have implicitly and explicitly made assumptions about the existence of various cryptographic primitives. In this lecture, we examine these assumptions and how they are related. We note that it is possible that none of the cryptographic primitives we've discussed even exist and that the existence of any of these cryptographic primitives in the asymptotic sense implies that $\mathcal{P} \neq \mathcal{NP}$, thus making any such proof very difficult (or impossible).

In this lecture, we are primarily interested in the asymptotic versions of the definitions of primitives. As such, we give an example of this type of definition and leave it to the reader to deduce the analogous definition for the others.

2 Abreviations

We use the following abreviations throughout this document:

- PRP: Pseudorandom Permutation
- PRF: Pseudorandom Function
- PRG: Pseudorandom Generator
- OWF: One-way Function
- OWP: One-way Permutation
- IND-CPA: Indistinguishability under Chosen Plaintext Attack
- IND-CCA2: Indistinguishability under Chosen Ciphertext Attack

3 Asymptotic Definition of a PRG

Let $\mathcal{G}$ be a family of deterministic functions:

$$\mathcal{G}_n : \{0,1\}^n \to \{0,1\}^{l(n)}$$

$\mathcal{G}$ is a polynomially-secure PRG iff

- $\forall n : l(n) > n$
- $\exists alg$: alg is computable in $p(n)$ time for some polynomial p and $\forall x \in \{0,1\}^*, alg(x) = \mathcal{G}(x)$.
- $\forall A, q : time(A) \leq poly(n) \Rightarrow Adv(A) = o\left(\frac{1}{q(n)}\right)$

Basically, $\mathcal{G}$ has to stretch its input, must be computable in polynomial time, and no adversary that runs in polynomial time can break the PRG with advantage greater than one over any polynomial (recall breaking a PRG is distinguishing its output from uniform).

4 Existence of OWPs Implies $\mathcal{P} \neq \mathcal{NP}$

Let $\mathcal{F}$ be a polynomially-secure family of OWPs. By definition we have:

- $\mathcal{F}_n : \{0,1\}^n \to \{0,1\}^n$
- $\forall n : \mathcal{F}_n$ is bijective
- $\mathcal{F}_n$ is computable in $poly(n)$ time
- $\mathcal{F}_n^{-1}$ is not computable in $poly(n)$ time

We will prove the contrapositive: if $\mathcal{P} = \mathcal{NP}$, then such a family cannot exist. The intuition here is that we can, given a value y of the form $y = \mathcal{F}_n(x)$, and given a guess x', we can verify whether $x' = \mathcal{F}_n^{-1}(y)$ in polynomial time. We simply apply $\mathcal{F}_n$ to x' and check if the result equals y.

Formally, let L be the following language

$$L = \{(y,t) : \exists x \ s.t. \ f(x) = y \wedge x \leq t\}$$

Given a pair (y', t') and a witness w, we can verify that (y', t') is in L by checking that $f(w) = y$ and that $w \leq t'$. Therefore, L is a language in $\mathcal{NP}$. Since we assumed $\mathcal{P} = \mathcal{NP}$, L must be in $\mathcal{P}$. Because L is in $\mathcal{P}$, we can recognize whether any given pair (y,t) is in L in polynomial time (without a witness). But now we can compute $\mathcal{F}_n^{-1}$ in polynomial time. The algorithm is basically, given y, do a binary search on t to find the smallest t such that (y,t) is in L. $\mathcal{F}_n^{-1}(y) = t$. This algorithm breaks any potential OWP, so they cannot exist if $\mathcal{P} = \mathcal{NP}$.

Note that we did not use the bijective property of $\mathcal{F}$. Thus, this exact same argument holds for OWFs too.

5 Existence of PRGs Implies Existence of OWFs

First, we note that the existence of a PRG of stretch $l(n) = 1$ implies the existence of a PRG of stretch $l(n) = n$. This is true for polynomially-secure families of PRGs also. Next, let $\mathcal{G}_n : \{0,1\}^n \to \{0,1\}^{2n}$ be a family of PRGs. Then, the family of functions $\mathcal{F}_n : x \mapsto \mathcal{G}_n(x)$ is a polynomially-secure family of OWFs. From the definition of a PRG, we know that $G(x) \sim u$. Also, due to the stretch of $\mathcal{G}$, we have

$$\Pr\left[u \xleftarrow{\$} \{0,1\}^{2n}; \exists s : \mathcal{G}_n(s) = u\right] \leq \frac{1}{2^n} \tag{1}$$

because $\mathcal{G}_n$ can only output at most 2^n different values out of the 2^{2n} possible strings of length $2n$. Now, the definition of advantage for a family of OWFs, $\mathcal{F}$ is

$$Adv(A) = \Pr[x \xleftarrow{\$} \{0,1\}^n; \mathcal{F}_n(A(\mathcal{F}_n(x))) = \mathcal{F}_n(x)]$$

Replacing $\mathcal{F}_n(x)$ with what we defined it to map to, we have

$$Adv(A) = \Pr[x \xleftarrow{\$} \{0,1\}^n; \mathcal{G}_n(A(\mathcal{G}_n(x))) = \mathcal{G}_n(x)]$$

But the output of $\mathcal{G}_n$ is indistinguishable from uniform (of length $2n$), so

$$Adv(A) \approx \Pr[u \xleftarrow{\$} \{0,1\}^{2n}; \mathcal{G}_n(A(u)) = u]$$

From Equation 1, we know that the chance that a value, s, even exists such that $\mathcal{G}_n(s) = u$ is $\leq \frac{1}{2^n}$, so there is no way that A can find such a value with probability greater than that. Therefore,

$$Adv(A) \leq \frac{1}{2^n}$$

and this is a polynomially-secure family of OWFs.

6 Secure Encryption Implies OWFs Exist

We prove this for IND-CPA secure symmetric-key and asymmetric-key encryption.

6.1 Symmetric-Key

We did not really discuss this before, but a secure symmetric-key encryption scheme actually needs three functions: E, D, and a key-generator, KG. The key-generator can be thought of as a randomized algorithm that takes nothing and return a key, k. However, it can also be thought of as a function that takes some random coin flips and deterministically generates k. Using this, we can construct of OWF, f, as $f(coins) = E_{KG(coins)}(0)$. Note that if an adversary could invert this function, then they could trivially recover the key the symmetric-key encryption scheme using one known plaintext (PTXT = 0) query. All it has to do is invert $y = f(coins)$ by computing $KG(f^{-1}(y)$. Thus, no adversary can invert f unless (E, D, KG) isn't IND-CPA secure.

6.2 Asymmetric-Key

Only a minor tweak to the above argument is needed. KG now outputs both the public and private keys, and $f(coins) = E_{first(KG(coins))}(0)$.

7 Creating Public-key Signatures from OWFs

First, we prove that we can create a one-time signature—a signature that can only be used once. Extending this into multi-use signatures is simple, we just sign (mes, new public key) instead of just the message. Also, in practice, a hash (such as SHA1) is taken of the message before signing in order to shrink the amount of signature needed down to a fixed, smaller size. These practical considerations are not needed for a proof of existence and are thus ignored in this description. We give only a simple construction which is intended to make it plausible that it is possible to build a secure public-key signature scheme out of any OWF. Actually proving this result require a more complicated construction and more work to come up with a proof, but the following conveys the essential insight.

Assume we have a OWF, f. The private key is a sequence of blocks of random bits,

$$tk = (x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}, \ldots, x_{n,0}, x_{n,1})$$

The size of each x block determines the security parameters. Let $y_{i,j} = f(x_{i,j})$, then the public key is

$$pk = (y_{1,0}, y_{1,1}, y_{2,0}, y_{2,1}, y_{3,0}, y_{3,1}, \ldots, y_{n,0}, y_{n,1})$$

Now, if you want to sign message b, containing bits, $b_1, b_2, b_3, \ldots, b_n$, you send the signature,

$$sig = (x_{1,b_1}, x_{2,b_2}, \ldots, x_{n,b_n})$$

along with the message. Note that since this reveals parts of the private key, it can only be used once.

As presented, this scheme has an extremely large public key, and it cannot be extended to multiple uses in the manner described above. However, the following variant can, if f is also assumed to be collision-resistant. Instead of sending all the y values, we instead send a single value α that summarizes all of them. α is created via a tree structure of f invocations. Here is a simple example with $n = 4$.

$$z_1 = f(y_{1,0}||y_{1,1})$$

$$z_2 = f(y_{2,0}||y_{2,1})$$

$$\alpha = f(z_1||z_2)$$

Now, if you want to send message b, you need to sign with

$$sig = (x_{1,b_1}, y_{1,\overline{b_1}}, x_{2,b_2}, y_{2,\overline{b_2}}, \ldots, x_{n,b_n}, y_{n,\overline{b_n}})$$

The receiver can compute f for each x in the signature, and, using the bits of the message, put the resulting y values in the correct order. Then the receiver can apply f in the tree structure described to see if α results; If so, the signature is valid. Also, since the actual public key transmitted is smaller than the message, we can sign (mes, new public key). In the previous scheme you needed at least 1 bit of key per bit of sent message, so this was impossible.

8 Other Proven Implications

The following is a collection of implications, given without proof. These were only briefly mentioned in class.

- If $\mathcal{F}$ is a polynomially-secure family of PRFs, then $\mathcal{G}(k) = \mathcal{F}(k, 0)$ is a polynomially-secure family of PRGs.
- The existence of PRGs implies the existence of PRFs (GGM construction, lecture 4)
- The existence of PRPs implies the existence of PRFs (switching lemma, lecture 5)
- The existence of PRFs implies the existence of PRPs (not proven in this course)
- The existence of OWFs implies the existence of PRGs [1]
- The existence of OWPs implies the existence of PRGs (also given in [1] and much easier to prove; see lecture 17)
- Secure public-key encryption implies that secure key agreement schemes exist (Alice generates key and encrypts with Bob's public key)
- The existence of trapdoor OWPs implies the existence of secure public-key encryption (Goldwasser-Micali, lecture 15)

Using these, and a few others not mentioned here, we get a set of primitives such that if any one of them exists, they all exist. This set includes:

- OWFs
- PRGs
- PRFs
- PRPs
- IND-CPA symmetric-key encryption
- MACs
- IND-CCA2 symmetric-key encryption
- secure bit-commitment
- coin flipping
- public-key signatures

Note that this set does *not* include OWPs, public-key encryption, trapdoor OWPs, or key agreement schemes. The following is a list of interesting implications whose inverses are not believed to be true:

- OWFs imply $\mathcal{P} \neq \mathcal{NP}$
- OWPs imply OWFs
- Trapdoor OWPs imply secure public-key encryption
- Public-key encryption implies key-agreement
- Public-key encryption implies OWFs
- key-agreement implies OWFs

Another interesting fact to note is that existence of any of the cryptographic primitives imply OWFs exist, and thus that $\mathcal{P} \neq \mathcal{NP}$.

References

[1] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.