ICS 280: Introduction to Cryptography 1/27/2004

Handout 1: Exploring the cryptographic definitions of security

This handout is intended to guide you through some thought experiments about definitions of computational security for a cryptographic scheme. In modern cryptography, we usually define security of a scheme by requiring that every (probabilistic) polynomial time adversarial algorithm has only a negligible *advantage* in breaking this scheme. The "adversarial advantage" is defined depending on what security property our security definition tries to model. For example, in the two types of cryptographic schemes we have seen so far in class, the adversarial advantage was defined as follows:

- One-way functions are defined in terms of (non-existence of) an adversary whose advantage is the probability of inverting this function on a random value.
- An encryption scheme is defined in terms of (non-existence of) an adversary whose adversarial advantage is the difference between the probability that the adversary guesses correctly which of the two given messages are encrypted in the ciphertext, and the probability that the adversary errs.

In this handout we'll ask two questions about such definitions:

1. If there exists a probabilistic polynomial time (PPT) adversarial algorithm whose advantage against some scheme is higher than negligible, but still very small, does such an algorithm really constitute a practical threat against this scheme? For example, can this algorithm be transformed into another PPT algorithm that breaks the scheme with an *overwhelming* probability?
2. If there exists a PPT adversarial algorithm whose advantage against a scheme is indeed negligible, can a repeated execution of such adversary boost this advantage, resulting in an adversarial algorithm which still runs in probabilistic poly-time but whose advantage is higher than negligible?

Recall first these two definitions:

Definition 1 *A probabilistic algorithm A is em probabilistic polynomial time [PPT] if the running time function $T_A(n)$ is polynomial in n, i.e. there is a constant d s.t. $T_A(n) = O(n^d)$. In other words, for all n and $x \in \{0,1\}^n$, the running time of A on input x is upper bounded by $T_A(n)$, where function T_A is (asymptotically) bounded from above as $T_A(n) = O(n^d)$.*

Definition 2 *Function $\epsilon : \boldsymbol{Z} \to R$ is* negligible *if it is asymptotically bounded from above by an inverse of any polynomial, i.e. if $\epsilon(n) = O(1/n^d)$ for* every *constant d.*

1 Can we boost a small but not negligible adversarial advantage into an *overwhelming* advantage?

Assume that there exists a PPT algorithm A whose adversarial advantage is higher than negligible, but still very small. For example, let A's advantage be $1/p(n)$ for some polynomial $p(n)$.

Assume that the adversarial advantage for the cryptographic scheme is defined in such a way that:

1. An event of A's success on instance y of the scheme can be *easily tested*. This is the case for one-way functions: In the definition of OWF (see the appendix), A's success on instance y is defined as an event that A outputs z s.t. $f(z) = y$. This can be easily tested by computing f on A's output z and comparing it to y. By contrast, there is no such test of adversarial success in the definition of a (security of) indistinguishable encryption: If an adversary guesses correctly on input (m_0, m_1, c) that c is a ciphertext of m_0 rather than m_1, there is no way to efficiently test if this judgment is correct.

2. For an adversary A whose adversarial advantage is ϵ, we additionally require that *for every* instance y of the attacked scheme, the probability that A succeeds in attacking *this instance* of the scheme is ϵ. This does not need to be always the case. For example, a one-way-function inverting algorithm A might have an advantage ϵ in inverting a OWF candidate function f, but it might always fail to invert $y = f(x)$ for even x's, and have 2ϵ probability of inverting $y = f(x)$ instances for every odd x. However, there are many cryptographic schemes, called *random self-reducible*, which have this property. See the subsection below.

If the scheme meets these two properties, then we can construct an algorithm A' which, on instance y of the scheme with security parameter k, will execute as follows: For $i = 1$ to $n * p(n)$, A' does the following: (1) A' runs A on input y and gets A's output z; (2) A' tests if z indeed "breaks" the instance y [here we use the property that A's success in breaking an instance of the scheme can be efficiently tested], and if it does, A outputs z and stops. Otherwise, A' repeats the loop for the next i. If A never succeeds in these $n * p(n)$ repeated trials, A' fails (i.e. outputs a "fail" output).

By the second property, we have that every time A executes on input y, the probability that its output z is a success against the scheme is $1/p(n)$. In this case the probability that A' succeeds on instance y is almost 1, namely $Adv_{A'}(n) = 1 - 2^{-n}$, because the probability that A' *fails* is at most the probability that each of the $n * p(n)$ runs of A fails. Since, by the first property, each run of A fails with probability at most $1 - 1/p(n)$ and moreoever all these events (the event that A fails in the first run, the event that it fails in the second run, and so on) are independent, we have that A' fails with probability at most

$$(1 - 1/p(n))^{n*p(n)} = ((1 - 1/p(n))^{p(n)})^n \approx (1/e)^n < (1/2)^n = 2^{-n}$$

(We use the fact that $(1 - 1/k)^k$ approaches $1/e < 1/2$ for large k.)

1.1 *Random Self-Reducible* Schemes

The second assumption about the considered scheme, i.e. that if an adversary has an overall advantage ϵ against the scheme then it has this advantage for every instance of the scheme, is satisfied by schemes which are random self-reducible. We call a scheme *random self-reducible* if any instance y of the scheme can be efficiently transformed into a *random* instance y', s.t. the solution to the y' instance can be then efficiently translated back into the solution for the instance y.

We'll explain this concept on the example of one-way functions: A OWF function f defined on domain D_k for security parameter k (see Appendix A) is *random self-reducible* if there are two probabilistic poly-time procedures R_1, R_2, s.t. R_1 on input $(y, 1^k)$ outputs pair (y', d), and R_2 on input (z', d) outputs z, with the following properties:

1. For every k, every $x \in D_k$, and $y = f(x)$, output y' of $R_1(y, 1^k)$ is distributed as the random variable $f(z)$ for z chosen uniformly in D_k.

2. For every k, every $x \in D_k$, $y = f(x)$, and every output (y', d) of $R_1(y, 1^k)$, if z' is a solution to instance y', i.e. if $f(z') = y'$, then $R_2(z', d)$ returns solution z to instance y, i.e. z s.t. $f(z) = y$.

We will show that every one-way function which is homomorphic over the domain of this function, is also random self-reducible.

First we need to explain the new term: We call a function f *homomorphic* over some group, for example the additive group $\mathbb{Z}_{p-1}$, if there exists some operation *op* s.t. $f(x+d) = op(f(x), f(d))$ for all $x, d \in \mathbb{Z}_{p-1}$, where '$x + d$' stands for an operation in $\mathbb{Z}_{p-1}$, i.e. for $x + d \bmod p - 1$. For example, the discrete exponentiation function $f_{(g,p)}(x) = g^x \bmod p$, which is believed to be one-way for example if g is a generator of $\mathbb{Z}_p^*$ and $p - 1$ has a large enough prime factor, is homomorphic over $\mathbb{Z}_{p-1}$, because $g^{(x+d \bmod p-1)} = g^x * g^d \pmod{p}$.

It's easy to see that such homomorphic one-way function is random self-reducible because *any* instance $y = f_{(g,p)}(x) = g^x \bmod p$ for some $x \in \mathbb{Z}_{p-1}$ can be translated into a *random* instance y' by picking $d \leftarrow \mathbb{Z}_{p-1}$ at random and assigning $y' = y * g^d \pmod p$. In other words, the PPT algorithm R_1 which *reduces* any instance y to a random instance picks d at random in $\mathbb{Z}_{p-1}$ and outputs $y' = y * g^d \pmod p$. Since d is picked uniformly in $\mathbb{Z}_{p-1}$, $x+d \bmod p-1$ is uniformly distributed in $\mathbb{Z}_{p-1}$ as well. Then, if some algorithm inverts this random instance y' by outputing z' s.t. $y' = f_{(g,p)}(z')$ then $R_2(z', d)$ returns $z = z' - d \bmod p - 1$, and then if $g^{z'} = y' \bmod p$ then $g^z = y \bmod p$.

2 Can we boost a negligible advantage into a non-negligible one?

If some adversarial PPT algorithm A has indeed only negligible but non-zero probability of success (say, in inverting a one-way function), is it possible to transform A into another algorithm A' which would still run in a polynomial time, but whose success would be non-negligible?

Well, we don't know: Maybe such strategy does exist, in which case there would be no one-way functions![1] However, here is an argument why at least one generic attempt at creating such A' does not succeed:

Assume that PPT algorithm A has advantage ϵ_n in inverting a one-way function s.t. for all polynomials $p(n)$, for all large enough n we have $\epsilon_n < 1/p(n)$. Assume furthermore just like in the previous section that A's success can be tested and than OWF f is random self-reducible. Let's try to use the same idea as before, and construct A' which executes n^d trials of A on the same input, for some constant d.[2] We'll show that the advantage of A' in inverting the input instance is still negligible.

Assume for contradiction that A' does invert f with probability higher than negligible, i.e. that there is a constant c s.t. $Adv_{A'}(n) > 1/n^c$ for infinitely many n's. By the same argument as in the previous section, we have that the advantage of A' is equal to $1-(1-\epsilon_n)^{n^d}$, and hence it must be that for infinitely many n's, $1-(1-\epsilon_n)^{n^d} > 1/n^c$, and hence $\epsilon_n > 1-(1-1/n^c)^{1/n^d}$. We'll show that there exists constant c' s.t. $1-(1-1/n^c)^{1/n^d} > 1/n^{c'}$ for all sufficiently large n's, which will then mean that ϵ_n is non-negligible, and thus the assumption that $Adv_{A'}$ is non-negligible leads to a contradiction.

We want to find c' s.t.

$$(1-1/n^{c'}) > (1-1/n^c)^{1/n^d}$$

But

$$(1-1/n^c)^{1/n^d} = ((1-1/n^c)^{n^c})^{1/n^{d+c}} \approx (1/e)^{1/n^{d+c}}$$

so we need c' s.t.

$$(1-1/n^{c'}) > (1/e)^{1/n^{d+c}}$$

i.e.

$$(1-1/n^{c'})^{n^{d+c}} > 1/e$$

Take $c' = d+c+1$, and we have

$$(1-1/n^{c'})^{n^{d+c}} = (1-1/n^{d+c+1})^{n^{d+c+1}/n} \approx (1/e)^{1/n} > 1/e$$

A Definition of One Way Functions

We give a definition of a one-way function which accounts for different *domains* of the function dopending on the security parameter:

Definition 3 *Function $f : \{0,1\}^* \to \{0,1\}^*$ is one-way (for domains $D_1, D_2, ...$) if:*

- *It is easy to compute, i.e. if there is a PPT algorithm A s.t. $A(x) = f(x)$ for all x.*
- *It is hard to invert, i.e. if for all PPT algorithms A, there exists a negligible function $\epsilon(n)$ s.t. $Adv_A(n) \leq \epsilon(n)$, where the adversarial advantage $Adv_A(n)$ is defined as:*

$$Adv_A(n) = Pr[f(z) = y \mid x \leftarrow D_n; y \leftarrow f(x); z \leftarrow A(y, 1^n)]$$

[1] Because there's always a PPT algorithm A which has a *negligible but non-zero* advantage $Adv_A(n)$. On input $(y, 1^n)$, A chooses $x \in D_n$ at random and outputs x if $f(x) = y$ (otherwise A fails). Note then A is polynomial-time (because D_n must be efficiently samplable), and that $Adv_A(n)$ is at least $1/2^{|D_n|} > 0$.

[2] The argument of the previous section shows that if A's advantage was $1/n^c$, it was enough to run $n * n^c = n^{c+1}$ trials. Here we try n^d trials for any constant d.