CS.202: Intro to Cryptography 10/7/2016

Feasible Attack = Polynomial-time + non-negligible advantage

Will not grade solutions, but do it by Thursday, 10/13: Will release solutions then.

This homework will not count towards the grade, but should do it for the sake of your own understanding, and you should do it *this week* because it guides you to gaining some mental comfort with the notion of polynomial-time bound on adversary's computing time and the characterization of "insignificant" probability using the notion of a negligible function. We will use these notions throughout the class, and I will assume you will know how to work with these concepts, e.g. by being able to answer questions like those below.

1 Polynomial times negligible is still negligible

Recall that function $\epsilon : N \rightarrow [0,1]$ is negligible if for every polynomial $p(\cdot)$ there exists point n_0 s.t. for all $n \geq n_0$ we have that $\epsilon(n) < 1/p(n)$. (Btw, this is equivalent to asking that for every d there exists n_0 s.t. for all $n \geq n_0$ we have that $\epsilon(n) < 1/n^d$.) Show that function $p(n) \cdot \epsilon(n)$ is negligible if ϵ is negligible and p is a polynomial.

2 How safe are negligible attacks if attack resources grow?

Adversarial advantage in breaking some cryptographic scheme is a function of the security parameter λ used to instantiate the scheme (λ is typically the length of the key) and of the adversary's computational resources. Assume that the adversarial advantage grows linearly with adversary's resources, i.e. if adversary's computing power grows by a factor of f then his chance of breaking a cryptosystem will grow by the same factor of f. Assume that an adversarial advantage against a cryptographic scheme, for an adversary whose computational resources are limited by some fixed bound X (for concreteness think of X as 2^{60} CPU ops), is expressed by a (decreasing) function $\epsilon(\lambda)$ of the security parameter λ. Assume that $\epsilon_1 = 2^{-30}$ is considered safe in the context of some application. Whatever function $\epsilon(\lambda)$ is, this maps to some threshold value λ_1 which is the smallest value s.t. $\epsilon(\lambda_1) \leq \epsilon_1$, but since we can assume that ϵ is *strictly* decreasing, you can just define λ_1 as the value s.t. $\epsilon(\lambda_1) = \epsilon_1$.

Consider the possibility that adversary's resources grow by a factor of f, e.g. because of continuing decrease in computational costs. To achieve the same security bound of 2^{-30} against such stronger attacks we need to increase the security parameter to a new threshold value λ_2 s.t. $\epsilon(\lambda_2) = \epsilon_2$ where $\epsilon_2 = (1/f) * \epsilon_1$. To see why, note that if the adversary has ϵ advantage using X computation, then if he can perform $f * X$ computation instead, then one thing he can do is to repeat the X-computation attack f times, which will bump up his probability of attack to $1 - (1-\epsilon)^f$, which for small ϵ comes very close to $f * \epsilon$. This is why if adversary's resources grow by a factor of f and we need to maintain the 2^{-30} security bound then we need to set λ s.t. $\epsilon(\lambda)$ satisfies $f * \epsilon(\lambda) \leq 2^{-30}$.

1. Consider the following functions $\epsilon(\lambda)$, and in each case do the following: (a) compute λ_1 s.t. $\epsilon(\lambda_1) = \epsilon_1 = 2^{-30}$, (b) express λ_2 as a function of f (note that λ_2 should satisfy $\epsilon(\lambda_2) = \epsilon_2 = (1/f) * \epsilon_1 = (1/f) * 2^{-30}$), (c) compute λ_2 for $f = 2^{40}$.

1. $\epsilon(\lambda) = 2^{-\lambda}$
2. $\epsilon(\lambda) = 2^{-\sqrt{\lambda}}$
3. $\epsilon(\lambda) = 1/\lambda^2$

2. Assume that the computational cost of the cryptographic scheme used in this application is $T(\lambda) = \lambda^2$. (The cost of a realistic scheme can always be at most some polynomial in λ.) For each of the above three cases of function ϵ, do the following: (a) compute $T_1 = T(\lambda_1)$, (b) state what $T_2 = T(\lambda_2)$ is as a function of f, (c) express the fractional increase in the running time of the scheme, i.e. $e = T_2/T_1$, as a function of f.

3. In each of the three cases of function ϵ, describe the relation between e and f which you found in part (2c) in words, e.g. is it polynomial? is it linear? is it (poly)logarithmic? Note that the relation between e and f is the relation between the growth in the computational cost born by the "good guy", i.e. the user of the cryptographic scheme, and the growth in the computational cost born by the "bad guy", i.e. the adversary trying to break this scheme. For each of these three cases state whether this relation between e and f seems good or bad to you and explain why.

4. For each of the three functions ϵ, state whether this function is negligible or non-negligible. Explain any correlation you see between your answer here and in part (3).

3 No perfect security against even very simple attacks

Consider a stream cipher $E(k, m) = G(k) \oplus m$ where G is a PRG s.t. $|G(k)| = p(n)$ if $|k| = n$. For security parameter n the key space of E is $K_n = \{0, 1\}^n$ and the message space is $M_n = \{0, 1\}^{p(n)}$. Pick polynomial p and messages m_0, m_1 in M_n in any way you like, and show a (simple!) algorithm A which attacks the indistinguishability of E by performing a single execution of G (i.e. your A should run G on just one n-bit input string) and which achieves a non-zero adversarial advantage against E. Specifically, construct A s.t. $Pr[A(m_0, m_1, E(k, m_1)) = 1] = 1/2 + \epsilon$ and $Pr[A(m_0, m_1, E(k, m_0)) = 1] = 1/2 - \epsilon$ for some $\epsilon > 0$, where each probability goes over random k in K_n and random choices made by A. Is ϵ a negligible function of n? (It should be...)

4 Every non-negligibile advantage is dangerous

Here is another reason why we need to equate security with a negligible bound on adversarial advantage. Consider an attacker A on indistinguishability of encryption E which for some fixed pair of messages m_0, m_1 achieves $\mathsf{Adv}_{A,E}(n) = 1/n^d$ for some d. Note that however large degree d is, n^d is a polynomial, so $\mathsf{Adv}_{A,E}$, even if it is small, is not negligible. Consider for simplicity an attacker A s.t. $Pr[A(E(k, m_1)) = 1] = 1/2 + 1/n^d$ and $Pr[A(E(k, m_0)) = 1] = 1/2 - 1/n^d$ (this attacker achieves $\mathsf{Adv}_{A,E}(n) = 2/n^d$, but that's OK). Consider encryption E' on the same message space as E, which runs encryption E on the same message t times, each time with a fresh random key. (In other words, E' models using E to send the same message t times in a row.) Consider an attacker A' on E' (for the same fixed messages m_0, m_1) which takes ciphertext c_i generated by the i-th instance of E, and runs $A(c_i)$ to output bit B_i, which A' interprets as a vote whether or not the message encrypted in c_i is m_0 or m_1. After gathering such votes for all t ciphertexts (note that they either all encrypt m_0 or they all encrypt m_1), A' outputs its bit as the majority of B_i's, i.e. A' outputs 1 if $\sum_{i=1}^{t} B_i > t/2$ and 0 otherwise (assume t is odd). Show that for t which is some polynomial in n, the advantage of A' against indistinguishability of E' is almost 1, specifically $\mathsf{Adv}_{A',E'}(n) \geq (1 - 2^{-n})$. For what t (as function of n and d) is this true? *Hint: Use a Chernoff bound, and a Wikipedia if you don't remember what a Chernoff bound is...*

5 But negligibile advantage cannot be boosted

In problem 3 you showed that for every encryption E there always exist an efficient (=polynomial-time) attacker A s.t. $Pr[A(E(k, m_0)) = 1] = 1/2 + \epsilon$ and $Pr[A(E(k, m_0)) = 1] = 1/2 - \epsilon$ for some *negligible* function ϵ. Show the converse of problem 4, i.e. that for any polynomial t (polynomial in n), the boosting strategy A' described in problem 4 will fail to convert this negligible advantage of A into a non-negligible advantage.

In other words, while problem 4 showed you that polynomial-time repetition of an attack A with *non-negligible* advantage can result in polynomial-time attack A' with *almost perfect* advantage, here you should show that polynomial-time repetition of an attack A with *negligible* advantage can only result in an attack A' which still has *negligible* advantage.

Put it intuitively:

$$\text{polynomial} \times (1/\text{polynomial}) \approx 1$$

$$\text{polynomial} \times \text{negligible} \leq \text{negligible}$$

The last statement seems like a restatement of problem 1 but it is not, because the "boosting" strategy A' we are considering here cannot be analyzed by a simple union bound argument, i.e. it's not the case that if A' runs A for $t = p(n)$ times, for some polynomial p, and that A' "wins" if and only if A "wins" in any of these t runs. If this was the case and (each of these runs constituted an independent security experiment), and if Adv_A and $\mathsf{Adv}_{A'}$ stand respectively for the probability that A and A' "win" in their respective security experiments, then indeed you could immediately give the following bound using the union bound:

$$\mathsf{Adv}_{A'} \leq p(n) \times \mathsf{Adv}_A$$

But here it is not the case that A' wins iff A wins in one of the t experiments: In every experiment a copy of A outputs a bit "vote", A' collects all these votes and makes its decision by following the majority of the votes, so the relation between $\mathsf{Adv}_{A'}$ and Adv_A is more complex than the simple inequality $\mathsf{Adv}_{A'} \leq p(n)\mathsf{Adv}_A$.