Hard-Core Predicates

Sandip Karmakar and Satrajit Ghosh

January 21, 2009

1 Hard-Core Predicates

In cryptography, a hard-core predicate of a one-way function f is a predicate b (i.e., a function whose output is a single bit) which is easy to compute given x but is hard to compute given $f(x)$. In formal terms, there is no probabilistic polynomial time algorithm that computes $b(x)$ from $f(x)$ with probability significantly greater than one half over random choice of x.
A hard-core predicate captures "in a concentrated sense" the hardness of inverting f. Loosely speaking, a polynomial time predicate b is called a hardcore of a function f if every efficient algorithm, given $f(x)$, can guess $b(x)$ with success probability that is only negligibly better than one-half.

Definition 1. *A polynomial-time-computable predicate $b : \{0,1\}^* \rightarrow \{0,1\}$ is called a* **hard-core** *of a function f if for every probabilistic polynomial time algorithm A',every positive polynomial $p(.)$, and all sufficiently large n's,*

$$Pr[A'(f(U_n)) = b(U_n)] < \frac{1}{2} + \frac{1}{p(n)}$$

Note that, for every $b : \{0,1\}^*\{0,1\}$ and $f : \{0,1\}^* \rightarrow \{0,1\}$ there exist obvious algorithms that guess $b(U_n)$ from $f(U_n)$ with success probability atleast one-half, e.g. the algorithm that obliviously of its input, outputs uniformly choosen bit. Also if b is a hardcore predicate for any function, then $b(U_n)$ must be almost unbiased (i.e.,$|Pr[b(U_n) = 0] - Pr[b(U_n) = 1]|$ must be a negligible function of n).

Since, b itself is polynomial time computable, the failure of efficient algorithms to approximate $b(x)$ from $f(x)$ (with success probability non-negligibly higher than one-half) must be due to,

1. either to an information loss of f. i.e. f not being one-to-one.
2. or the difficulty of inverting f .

For example, the predicate $f(\sigma\alpha) = 0\alpha$ is a hardcore of $b(\sigma\alpha) = \sigma$, hence in this case the fact that b is a hardcore of the function f is due to the fact that f looses information. On the other hand if f looses no information hard-cores of f exist only if f is one way.(Here the case where the hardness of approximating $b(x)$ from $f(x)$ is due to computational reasons and not to information theoretic ones will be considered. As non-bijective functions do not have applications in cryptography, we will consentrate ourselves to the second kind only.

2 One Way Functions

A one-way function is a function that is easy to compute but "hard to invert". "Easy to compute" means that some algorithm can compute the function in polynomial time (in the input size). "Hard to invert" means that no probabilistic polynomial-time algorithm can compute a preimage of $f(x)$ with better than negligible probability, when x is chosen at random. Note that unlike hardness in most of complexity theory (e.g., NP-hardness), "hard" in the context of one-way functions refers to average-case hardness rather than worst-case hardness. Note that just making a function "lossy" (not one-to-one) does not make it a one-way function; inverting a function in this context merely means identifying some preimage of a given value, which does not require the existence of an inverse function. For example, $f(x) = x^2$ is not invertible (for example $f(2) = f(-2) = 4$) but is also not one-way, since given any value, you can compute one of its preimages in polynomial time by taking its square root.

3 Trapdoor oneway functions

One way function that can be easily inverted with a secret information is called the trap-door one way function and the secret information is called trapdoor information.

3.1 Example: RSA

Let $y = x^e mod(pq)$. Clearly this is easy to compute. Given y, e and $N = pq$, we do not know efficient techniques to compute x. But if we have a trap-door $d = e^{-1} mod(p-1)(q-1)$ it becomes easy to compute x and hence to invert the function.

4 Goldreich-Levin Theorem

If there is a family of trapdoor permutations, then there is a family with a hard core predicate.

5 Encryption of one bit b, that is provably secure

The following algorithm is proposed which encrypts a single bit, b. Given $(G, F, I), t_k$ and a hardcore predicate b.

1. Key Generation: G
   -Return (k, t_k)

2. Encryption: $E(b, t_k)$
   -Pick random $X \in \{0,1\}^n$
   -Return $F(X, k), b \oplus B(X, k)$

3. Decryption: $D((z,c),k,t_k)$
   $X = I(z,t_k)$
   Return $c \oplus B(X,k)$

5.1 The Above Encryption is MI Secure

5.2 Example with RSA

1. $B(X,(N,e)) = Xmod2$ is a hardcore predicate for RSA, that is given $(N,e), X^e modN$ it is hard to guess $Xmod2$ with a non-negligibly larger probability than $\frac{1}{2}$.

2. Encrypt $b \in \{0,1\}$ with RSA
   Pick $X \in \{0,1\}^n$
   Compute, $X^e modN$, $XOR(b, Xmod2)$.

6 The Encryption is MI-secure

We have to prove that,
$Pr_{(k,t_k)\in G(n),X\in\{0,1\}^n,b\in\{0,1\}}[A(F(X,k), b\oplus B(X,k),k) = b] \leq \frac{1}{2} + \epsilon(n)$
Proof: Suppose this encryption is not (t,ϵ)-MI secure.
Let,
$Pr_{(k,t_k)\in G(n),X\in\{0,1\}^n,b\in\{0,1\}}[A(F(X,k), b\oplus B(X,k),k) = b] > \frac{1}{2} + \epsilon(n)$
Consider the algorithm $A'(y,k)$.
Pick random $c \in \{0,1\}$.
Return $c \oplus A(y,c,k)$.
Thus,
$Pr_{X\in\{0,1\}^n}[A'(F(X,k),k) = B(X,k)]$
$= Pr_{(k,t_k)\in G(n),X\in\{0,1\}^n,c\in\{0,1\}}[A(F(X,k),c,k) = B(X,k)\oplus c]$
$= Pr_{(k,t_k)\in G(n),X\in\{0,1\}^n,b\in\{0,1\}}[A(F(X,k), b\oplus B(X,k),k) = b] > \frac{1}{2} + \epsilon(n)$
Hence proved.