Theoretical constructions of pseudorandom objects

Leanne Streekstra

We want to built our proofs of security on the mildest assumption possible. The assumption that one-way functions exist is milder than the assumption that pseudorandom objects exist.

What is a one-way function?
A one-way function is easy to compute, but hard to invert. A candidate one-way function is *prime factorization.*

Definition
A function f: $\{0,1\}^* \to \{0,1\}^*$ is a one-way function if:
1. There exists a PPT algorithm computing f.
2. For all PPT algorithms A, there exists a negligible function *negl* s.t.:
$\underset{x \leftarrow \{0,1\}^n}{Pr}[\mathrm{A}(\mathrm{f}(\mathrm{x})) \in \mathrm{f}^{-1}(\mathrm{f}(\mathrm{x}))] \leq \mathrm{negl}(\mathrm{n})$.

To go from one-way functions to pseudorandom objects we first need to define hard-core predicates.

What is a hard-core predicate?
A hard-core predicate is a single bit that is efficiently computable given x, but infeasible given only f(x).

Definition
A function hc: $\{0,1\}^* \to \{0,1\}$ is a hard-core predicate of a function f if:
1. hc can be computed in polynomial time.
2. For all PPT algorithms A, there exists a negligible function *negl* s.t.:
$\underset{x \leftarrow \{0,1\}^n}{Pr}[\mathrm{A}(\mathrm{f}(\mathrm{x})) = \mathrm{hc}(\mathrm{x})] \leq 1/2 + \mathrm{negl}(\mathrm{n})$.

Note that it is always possible to correctly guess hc(x) with probability 1/2.

It is not sure whether there exists a hard-core predicate for every one-way function. The following does hold:
- If f is a one-way function, then there exists a one-way function g along with a hard-core predicate hc:

$$\mathrm{g}(\mathrm{x},\mathrm{r}) \stackrel{def}{=} (\mathrm{f}(\mathrm{x}),\mathrm{r}), \text{ for } |\mathrm{x}|=|\mathrm{r}|.$$
$$\mathrm{hc}(\mathrm{x},\mathrm{r}) \bigoplus_{i=1}^{n} \mathrm{x}_i \cdot r_i$$

Where r is a random string and x_i is the i-th bit of x. Note that hc(x,r) outputs the XOR of a random subset of x. We can see r as selecting these random bits.
We now have the ingredients to construct a pseudorandom generator.

From one-way functions to pseudorandom generators

Pseudorandom generators with expansion factor l(n) = n+1
If f is a one-way permutation, we can construct a pseudorandom generator G in the following way:

$$\mathrm{G}(\mathrm{x}) \stackrel{def}{=} (\mathrm{f}(\mathrm{x}),\mathrm{hc}(\mathrm{x}))$$

Intuitively, G is pseudorandom as hc(x) is infeasible to compute from f(x) and thus looks random (=pseudorandom). f(x) is truly random when x is chosen uniformly at random, by the fact that f is a permutation.

Pseudorandom generators with arbitrary expansion
Take G to be a pseudorandom generator with expansion factor l(n)=n+1, then we can construct a pseudorandom generator $\widetilde{G}$, with expansion factor $\widetilde{l}$(n)=p(n) for any polynomial p(n), by iteration of G.

The idea here is that given a random input, G outputs a pseudorandom string. If we now output one of the n+1 bits, we can use the remaining n bits as input for G again. As these bits are pseudorandom, they are essentially as good as a truly random input. Iterating G in this way will give a pseudorandom $\widetilde{G}$ for any desired polynomial expansion factor.

From pseudorandom generators to pseudorandom functions
Let G be a PRG with expansion factor l(n)=2n. Denote G(k)=(G_0(k), G_1(k)), where |k|=|G_0(k)|=|G_1(k)|. We can now define a pseudorandom function F which takes one bit as input, in the following way:
$F_k(0)=G_0(k)$ $\quad$ $F_k(1)=G_1(k)$

As G is a pseudorandom function, the output of F defined in this way is pseudorandom as well, as it simply outputs half of G's output.
We can now define an F' that takes two bits as input as:
$F'_k(00)=G_0(G_0(k))$ $\quad$ $F'_k(01)=G_1(G_0(k))$ $\quad$ $F'_k(10)=G_0(G_1(k))$ $\quad$ $F'_k(11)=G_1(G_1(k))$

As G(k)=(G_0(k), G_1(k)) is indistinguishable from random, G(G(k)) will look as random as G(r) for a random string r. As F' outputs a part of the output of G(G(k)), it must hold that F' is indistinguishable from a random function. To construct a PRF which takes an input string of length n, we can apply G n times:
$F_k(x_1 x_2 \ldots x_n)=G_{x_n}(\ldots(G_{x_2}(G_{x_1}(k)))\ )$.
By the same reasoning as above, this function is a pseudorandom function.

From pseudorandom functions to pseudorandom permutation
-Combining a pseudorandom function with a 3-round Feistel network yields a pseudorandom permutation.

A strong PRP is indistinguishable from a random permutation even when given oracle access to both the permutation and its inverse.
-Combining a pseudorandom function with a 4-round Feistel network yields a strong pseudorandom permutation.

Concluding remarks
We can now conclude that the existence of one-way functions is a sufficient condition for CCA-secure encryption schemes and MACs that are unforgeable under chosen message attacks.
From the following statements we can conclude that it is also a necessary condition (see K& L for proofs).

-If there exists a pseudorandom generator, then there exists a one-way function.

However, it does not follow from this that one-way function are necessary, as we might be able to construct secure encryption schemes without pseudorandom generators or functions.

- If there exists a private-key encryption scheme that has indistinguishable encryptions in the presence of an eavesdropper, then there exists a one-way function.

We conclude that the existence of one-way functions is both necessary and sufficient for all private-key cryptography.

Reference
- Katz, J., Lindell, Y. (2008). Introduction to modern cryptography, ch 6.