PUBLIC KEY CIPHER (first draft)

1 the big picture

The public key is a set of functions $F_i : 2^n \to 2^n$. Typically there are n such functions, each recovering (indirectly) one bit of the plaintext. But the system is easily extended so that each of $2n$ functions recovers $\frac{1}{2}$ of a bit of the plaintext, and so on. The functions F_i are "heavy" in that many bits are required to represent them, even for $n = 16$. Users might even want to use a lighter version of the system, where each F_i recovers 2 bits of the plaintext, etc.

The set $2^n = \{0, 1, 2, 3, ..., 2^n - 1\}$ is essentially broken up into subsets T_i of equal size, one for each F_i. Each set T_i has a related set of the same size S_i. The sets T_i and S_i are secret. Only the functions F_i are public. The gist is that $F_i(x) \in S_i \implies x \in T_i$. The T_i chosen in such a way that knowledge of $F_i(x)$ for each i gives $x \in \cap_i T_i$, which is always just a single element, so that pre-image is recovered. The sender with the public key sends $F_1(x) \frown F_2(x) \frown ... \frown F_{n-1} \frown F_n(x)$. This is just the concatenation of the $F_i(x)$.

To choose the T_i, we generate a random permutation $\Psi : 2^n \to 2^n$. We can write out Ψ as a matrix where the ith column is the binary representation of $\Psi(i)$. Then the jth row is the binary representation of T_i (its characteristic or indicator function.) While it is possible to choose the T_i to each indicate a particular bit of x, it is more secure to choose instead from $(2^n)!$ equally good possibilities. Recovering x requires a little more work, but the "circuitry" of the F_i is far more obscured this way. This is because each F_i indicates membership in a randomly chosen subset of size 2^{n-1} from a larger set of size 2^n. So there are $\binom{2^n}{2^{n-1}}$, which is already large enough for a key, besides the complexity otherwise in the system. Note that the owner of the public key has to keep the T_i, the S_i, and Ψ secret.

2 the details

Now we discuss the construction of a particular F_i. I'll just use F for F_i, since the process is the same for each of the F_i. We need only the randomly chosen T that is associated with F. We will create 2^n sets R_i,

each of size 2^{n-1}.[1]

First we set $R_1 = T$. This is the only R_i that is not random. The rest of the R_i need only be some random choice of half the elements from 2^n. This choice will imply the choice of R_i^c, the complement of R_i. Note that S_i is defined to be R_{2^n}, or (more generally) the last R_i set. Once the construction is complete, we can throw away everything but the T_i and S_i associated with each F_i.

Now we can build F as a composition. $F = f_{2^n} \circ f_{2^{n-1}} \circ ... \circ f_2 \circ f_1$. Each f_i is created randomly subject to the constraint that $f_i(R_i) \subset R_{i+1}$ and $f_i(R_i^c) \subset R_{i+1}^c$. It's possible to make the f_i permutations that contain "subpermutations", but doing this consistently would make F_i a permutation, reducing security. Instead, just fulfill the weaker constraint. A permutation here and there shouldn't be a problem, but we want the mechanism of the F_i we are constructing to be as obscure as possible.

Note that $x \in T = R_1 \iff f_1(x) \in R_2 \iff f_2(f_1(x)) \in R_2$ and so on. So $x \in T \iff F(x) \in S$, as desired.

Now we "compress" our function F and obscure its mechanism. It can either be written as a function of natural numbers from 2^n to 2^n. Or it can be given (more obscurely) as n Boolean functions of 2^n bits. While those with the public key in Boolean form could also recover the integer understanding of the function, this could become expensive for large n. Note that F is not a permutation. In general, there are $(2^n)^{2^n}$ possibilities confronting a brute force attack. This function F is homophonic. It returns 1 bit of useful information (though this can be increased or decreased.) So it can range wildly within this mild constraint. Moreover the attacker doesn't have the luxury of tracking a particular bit of the plaintext. Instead the function simply maps one unknown half of 2_n to the other half. So $F(T) \subset S$ and $F(T^c) \subset S^c$.

3 generalizations

Note that the system can be expressed in terms of integers, so its easy to use ternary, etc., if one wants to offer the key in terms of ternary functions.

[1] This is not strictly required. Using about half of the set 2^n will work but slightly complicates the construction.