FORMACRYPT

Monday, March 6th, 2006

New Goals
from the Cryptographers' Point of View

David Pointcheval
Ecole normale supérieure
France

Optimality

- Refine reduction cost
- Provide the ***best*** reduction
  - By exhaustive search?
  - With heuristics?

Diffie-Hellman Problems

- Deal with the Diffie-Hellman Problems
- A direction:
  - Trapdoor Hard-to-Invert Group Isomorphisms

Let I be a set of indices. Informally a family of *trapdoor hard-to-invert group isomorphisms* is a family $F = \{f_m : X_m \to Y_m\}_{m \in I}$ satisfying the following conditions:

1. one can easily generate an index m, which provides a description of the function f_m – a morphism –, its domain X_m and range Y_m (which are assumed to be isomorphic, efficiently uniformly samplable finite groups), and a trapdoor t_m;
2. for a given m, one can efficiently sample pairs $(x, f_m(x))$, with x uniformly distributed in X_m;
3. for a given m, one can efficiently decide Y_m, meaning with this that on input m and a candidate value y one can efficiently test if $y \in Y_m$ or not[1].
4. given the trapdoor t_m, one can efficiently invert $f_m(x)$, and thus recover x;
5. without the trapdoor, inverting f_m is hard.

THtIG: Homomorphisms

1 – There exist a polynomial p and a probabilistic polynomial time Turing Machine Gen which on input 1^k (where k is a security parameter) outputs pairs (m, t_m) where m is uniformly distributed in $I \cap \{0,1\}^k$ and $|t_m| < p(k)$. The index m contains the description of X_m and Y_m, which are isomorphic groups, an isomorphism f_m from X_m onto Y_m and a set R_m of uniformly and efficiently samplable values, which will be used to sample $(x, f_m(x))$ pairs. The information t_m is referred as the *trapdoor*.

2.1 – There exists a polynomial time Turing Machine Sample^x which on input $m \in I$ and $r \in R_m$ outputs $x \in X_m$. Furthermore, for any m, the machine $\mathsf{Sample}^x(m, \cdot)$ implements a bijection from R_m onto X_m [2].

2.2 – There exists a polynomial time Turing Machine Sample^y, which on input $m \in I$ and $r \in R_m$ outputs $f_m(x)$ for $x = \mathsf{Sample}^x(m, r)$. Therefore, $\mathsf{Sample}^y(m, r) = f_m(\mathsf{Sample}^x(m, r))$.

3 – There exists a polynomial time Turing Machine Check^y which, on input $m \in I$ and any y, answers whether $y \in Y_m$ or not.

4 – There exists a (deterministic) polynomial time Turing Machine Inv such that, for all $m \in I$ and for all $x \in X_m$, $\mathsf{Inv}(m, t_m, f_m(x)) = x$.

5 – For every probabilistic polynomial time Turing Machine $\mathcal{A}$ we have that, for large enough k,

$$\Pr[(m, t_m) \leftarrow \mathsf{Gen}(1^k)\,;\, x \stackrel{R}{\leftarrow} X_m\,;\, y = f_m(x) :\, \mathcal{A}(m, y) = x] \leq \varepsilon(k),$$

where $\varepsilon(\cdot)$ is a negligible function.

Universal Composability

- Provide proofs in the Universal Composability framework

Gain

- Exhibit de gain of this direct method:
  Provide an example of protocol
  - We can prove under standard assumptions
  - Which requires stronger assumptions
    with other approaches