2011 Paper 4 Question 8

Security I

(*a*) In Windows NTFS, each file can have an associated access control list (ACL). Each entry has a type in $\{allow, deny\} \times \{explicit, inherited\}$.

(*i*) What restriction does the Windows Explorer graphical user interface impose on the order in which these types of access-control entries can appear in an ACL? [4 marks]

(*ii*) Give **one** example of a POSIX file access-control configuration for which an equivalent NTFS ACL violates this GUI restriction. [4 marks]

(*b*) Your colleagues used a pseudo-random function $f : \{0,1\}^{64} \to \{0,1\}^{64}$ in order to construct a permutation $g : \{0,1\}^{192} \to \{0,1\}^{192}$. The argument and return values of g are split into three 64-bit registers, respectively: $g(X_1, X_2, X_3) = (Y_1, Y_2, Y_3)$. The output of g is calculated as $Y_2 = f(X_1) \oplus X_2 \oplus f(X_3)$, $Y_1 = X_1 \oplus f(Y_2)$, and $Y_3 = X_3 \oplus f(Y_2)$, where $\oplus$ denotes bit-wise exclusive or.

(*i*) Show that g is indeed a permutation. [4 marks]

(*ii*) Show how an attacker who does not know f can efficiently distinguish g from most random permutations, after evaluating g on two different inputs. [4 marks]

(*iii*) After you point out this shortcoming to your colleagues, they propose an improved variant $g'(X_1, X_2, X_3) = (Z_1, Z_2, Z_3)$ that adds another round to g: $Z_1 = Y_1$, $Z_2 = f(Y_1) \oplus Y_2 \oplus f(Y_3)$, and $Z_3 = Y_3$.

Show how this variant still does not fix the problem of efficient distinguishability from most random permutations. [4 marks]