Still, there is something interesting to be said for *function problems* arising from *NP* problems. Suppose

\[ L = \{ x \mid \exists y R(x, y) \}\]

where \( R \) is a polynomially-balanced, polynomial time decidable relation. A *witness function* for \( L \) is any function \( f \) such that:

- if \( x \in L \), then \( f(x) = y \) for some \( y \) such that \( R(x, y) \);
- \( f(x) = "no" \) otherwise.

The class \( \textbf{FNP} \) is a collection of witness functions for languages in \( \textbf{NP} \). \( \textbf{FP} \) is the subclass of \( \textbf{FNP} \) of those functions computable in *polynomial-time*. 
A function which, for any given Boolean expression $\phi$, gives a satisfying truth assignment if $\phi$ is satisfiable, and returns “no” otherwise, is a witness function for SAT.

If any witness function for SAT is computable in polynomial time, then \( P = NP \).

If \( P = NP \), then for every language in NP, some witness function is computable in polynomial time, by a binary search algorithm.

\[
P = NP \text{ if, and only if, } FNP = FP
\]

Under a suitable definition of reduction, the witness functions for SAT are FNP-complete.
The *factorisation* function maps a number $n$ to its prime factorisation:

$$2^{k_1} 3^{k_2} \ldots p_m^{k_m}.$$ 

This function is in \textit{FNP}. The corresponding decision problem (for which it is a witness function) is trivial - it is the set of all numbers.

Still, it is not known whether this function can be computed in polynomial time.
Alice wishes to communicate with Bob without Eve eavesdropping.
In a private key system, there are two secret keys
\( e \) – the encryption key
\( d \) – the decryption key
and two functions \( D \) and \( E \) such that:

for any \( x \),

\[
D(E(x, e), d) = x.
\]

For instance, taking \( d = e \) and both \( D \) and \( E \) as exclusive or, we have the one time pad:

\[
(x \oplus e) \oplus e = x
\]
The one time pad is provably secure, in that the only way Eve can decode a message is by knowing the key.

If the original message $x$ and the encrypted message $y$ are known, then so is the key:

$$e = x \oplus y$$
In public key cryptography, the encryption key $e$ is public, and the decryption key $d$ is private.

We still have,

$$D(E(x, e), d) = x$$

If $E$ is polynomial time computable (and it must be if communication is not to be painfully slow), then the function that takes $y = E(x, e)$ to $x$ (without knowing $d$), must be in $\text{FNP}$. Therefore, public key cryptography is not provably secure in the way that the one time pad is. It relies on the existence of functions in $\text{FNP} - \text{FP}$. 
A function $f$ is called a \textit{one way function} if it satisfies the following conditions:

1. $f$ is one-to-one.
2. For each $x$, $|x|^{1/k} \leq |f(x)| \leq |x|^k$ for some $k$.
3. $f \in \text{FP}$.
4. $f^{-1} \notin \text{FP}$.

We cannot hope to prove the existence of one-way functions without at the same time proving $P \neq \text{NP}$.

It is strongly believed that the RSA function:

$$f(x, e, p, q) = (x^e \mod pq, pq, e)$$

is a one-way function.
Though one cannot hope to prove that the RSA function is one-way without separating P and NP, we might hope to make it as secure as a proof of NP-completeness.

**Definition**
A nondeterministic machine is *unambiguous* if, for any input $x$, there is at most one accepting computation of the machine.

**UP** is the class of languages accepted by unambiguous machines in polynomial time.
Equivalently, $\textbf{UP}$ is the class of languages of the form

$$\{ x \mid \exists y R(x, y) \}$$

Where $R$ is polynomial time computable, polynomially balanced, and for each $x$, there is at most one $y$ such that $R(x, y)$.
UP One-way Functions

We have

\[ P \subseteq UP \subseteq NP \]

It seems unlikely that there are any NP-complete problems in UP.

One-way functions exist if, and only if, P ≠ UP.
Suppose $f$ is a \textit{one-way function}.

Define the language $L_f$ by

$$L_f = \{(x, y) \mid \exists z (z \leq x \text{ and } f(z) = y)\}.$$ 

We can show that $L_f$ is in \textsf{UP} but not in \textsf{P}.
Suppose that $L$ is a language that is in UP but not in P. Let $U$ be an unambiguous machine that accepts $L$.

Define the function $f_U$ by

- if $x$ is a string that encodes an accepting computation of $U$, then $f_U(x) = 1y$ where $y$ is the input string accepted by this computation.
- $f_U(x) = 0x$ otherwise.

We can prove that $f_U$ is a one-way function.