Strong, Total, Commutative, Associative
One-way Functions

Sayak Chakraborti, Andrii Osipa, Parker Riley,
Zhuojia Shen, Evan Strohm, and Hassler Thurston

November, 2016
Outline

Build-up to Theorem 2.16
Theorem 2.16
Proof of Theorem 2.16

Outline
Total, 2-Ary functions
Commutativity and Associativity
Honesty
Invertibility
2-Ary One-way Functions
Strong Honesty and Noninvertibility

Useful Definitions:

1. What it means to be 2-ary and total
2. Commutativity
3. Associativity
4. Honesty and Invertibility for 2-ary functions
5. 2-Ary One-way Functions
6. Strong honesty (s-honesty)
7. Strong noninvertibility
Total, 2-Ary functions

- A function $f$ is 2-ary if it maps $\Sigma^* \times \Sigma^*$ to $\Sigma^*$.
- A 2-ary function $f$ is total if $f(x, y)$ is defined for all $x, y \in \Sigma^*$. 
Commutativity

A total, 2-ary function $f$ is commutative if for all $x, y \in \Sigma^*$, $f(x, y) = f(y, x)$. 
A total, 2-ary function $f$ is associative if for all $x, y, z \in \Sigma^*$, 
$$f(f(x, y), z) = f(x, f(y, z)).$$
A 2-ary function \( f \) (does not need to be total) is honest if there is a polynomial \( p \) such that for every \( y \in \text{range}(f) \), there are two strings \( x, x' \) such that:

- \( f(x, x') = y \), and
- \( |x| + |x'| \leq p(|y|) \).

Note that for a given \( y \), as long as there exist two strings \( x \) and \( x' \) that satisfy the above, \( f \) is honest.
A 2-ary function $f$ (does not need to be total) is polynomial-time invertible if there is a polynomial function $g$ such that for every $y \in \text{range}(f)$, the following hold:

- $y \in \text{domain}(g)$
- $g(y) = \langle a, b \rangle$
- $(a, b) \in \text{domain}(f)$, and
- $f(a, b) = y$. 
2-Ary One-way Functions

A 2-ary function \( f \) (does not need to be total) is one-way if:

- \( f \) is computable in polynomial time
- \( f \) is not polynomial-time invertible, and
- \( f \) is honest.
But we want stronger...

- What if, given the output \textit{and} one of the inputs, we still couldn’t determine the other input in polynomial time?
- Need an idea of strong honesty: given one input and the output, the other input can’t be too large.
A 2-ary function $f$ is s-honest if, given the output and one of the arguments, there’s an instance of the other argument that isn’t too large:

- There exists a polynomial $p$ such that for all $y, a \in \Sigma^*$ such that there is at least one $b$ where $f(a, b) = y$, there exists $b' \in \Sigma^*$ such that:
  - $f(a, b') = y$
  - $|b'| \leq p(|y| + |a|)$,

- (similarly), there exists a polynomial $p$ such that for all $y, b \in \Sigma^*$ such that there is at least one $a$ where $f(a, b) = y$, there exists $a' \in \Sigma^*$ such that:
  - $f(a', b) = y$
  - $|a'| \leq p(|y| + |b|)$. 
A 2-ary function $f$ is strongly noninvertible if it is s-honest, and, given the output and one of the arguments, there isn’t a way to figure out the other argument in polynomial time.

Formally: There is no polynomial function $g$ from $\Sigma^* \times \Sigma^*$ to $\Sigma^*$ such that, for all $y \in \text{range}(f)$ and all $(x_1, x_2) \in \text{domain}(f)$ s.t. $f(x_1, x_2) = y$:

- $f(x_1, g(y, x_1)) = y$, and
- $(y, x_1) \in \text{domain}(g)$.

The definition for the other argument is analogous. A function needs to satisfy both to be considered strongly noninvertible.
Motivation

2-ary one-way functions with these properties are of potential to use in the field of cryptography. Imagine that we have a function, \( f \), that takes two arguments, is strongly noninvertible, and is associative. Two parties that want to communicate secretly, Alice and Bob, could use this function to agree on a secret key, \textit{even if the channel used to do the agreement is insecure}. 
Alice chooses two large random strings, $a$ and $k$. She keeps $a$ secret and sends Bob the pair $\langle k, f(a, k) \rangle$.

Bob chooses a large random string, $b$, which he keeps secret, and sends Alice the result of $f(k, b)$.

Alice computes $f(a, f(k, b))$ (remember that $f(k, b)$ was given to her), and Bob computes $f(f(a, k), b)$ (again, $f(a, k)$ was given to him by Alice). Note that, because $f$ is associative, each party has just computed the same string. This will be the key used to encrypt future messages sent between them.
An eavesdropper would only have access to $k$, $f(a, k)$, and $f(k, b)$, but would need to know either $a$ or $b$ in order to learn the key. If $f$ is strongly noninvertible, then there won’t be a function that can discern either given the available information.
Now, for the theorem itself:

**Theorem**

One-way functions exist if and only if strongly noninvertible, total, commutative, associative, 2-ary one-way functions exist.

We will prove both the “if” and “only if” directions.
First, however, it will be useful to observe these equivalencies:

**Proposition**

The following are equivalent:

1. One-way functions exist.
2. 2-ary one-way functions exist.
3. $P \neq NP$

- 1 and 3 are equivalent by Theorem 2.5
- 2 implies 1: given $f$, let $g(\langle x, y \rangle) = f(x, y)$
- 1 implies 2: given $h$, let $h'(x, y) = \langle h(x), y \rangle$
Theorem 2.16, Restated

**Theorem**

One-way functions exist if and only if strongly noninvertible, total, commutative, associative, 2-ary one-way functions exist.

From Proposition 2.17, we see that the “if” direction is trivial: given any 2-ary one-way function, we can easily construct a one-way function of a single argument using our standard pairing function with nice properties. We will now show that the “only if” direction also holds.
Theorem 2.16 cont’d.

The hypothesis for the “only if” direction is that one-way functions exist. However, we know that this is equivalent to $P \neq NP$, so we will proceed with the proof by first assuming this, and continuing as follows:

- Let $L$ be in $NP - P$, and let $N'$ be a NPTM accepting $L$.
- By standard machine manipulation, we can construct a machine $N$ that accepts the same language but pads all accepting paths of input $x$ to be $p(|x|)$ bits long, where $p$ is a bounding polynomial for $N'$ and such that $p(n) > n$ for all $n$. 
Theorem 2.16 cont’d.

Definition (Witness String)

A string $w$ is a witness for the inclusion of $x$ in $L(M)$ if and only if $w$ represents an accepting path for $M(x)$.

- Let $W(x)$ be the set of all witnesses for the inclusion of $x$ in $L(N)$. Note that $x$ is in $L(N)$ if and only if $W(x)$ is non-empty, and that each element of $W(x)$ will have length exactly equal to $p(|x|)$. This means that no string can be its own witness, because $p(n) > n$.

- Let $t$ be a fixed string that is not in $L(N)$.

We now have what we need to prove the existence of functions with the desired properties.
Theorem 2.16 cont’d.

We will use our assumptions and definitions to construct such a function:

**Definition (Function $f$)**

$$f(\langle a, b \rangle, \langle c, d \rangle) =$$

- $\langle a, \text{lexmin}(b, d) \rangle$ IF $a = c \land b \in W(a) \land d \in W(a)$
- $\langle a, a \rangle$ IF $a = c \land \{b, d\} = \{a, w\}$ for some $w \in W(a)$
- $\langle t, t1 \rangle$ OTHERWISE

Recall that $t$ is a fixed string not in $L(N)$. $t1$ is $t$ with a 1 appended. $\text{lexmin}(a, b)$ returns the lexicographically lesser of $a$ and $b$. 
Theorem 2.16 cont’d.

Definition (Function \( f \))

\[
\begin{align*}
  f(\langle a, b \rangle, \langle c, d \rangle) &= \\
  &\quad \langle a, \text{lexmin}(b, d) \rangle \text{ IF } a = c \land b \in W(a) \land d \in W(a) \\
  &\quad \langle a, a \rangle \text{ IF } a = c \land \{b, d\} = \{a, w\} \text{ for some } w \in W(a) \\
  &\quad \langle t, t1 \rangle \text{ OTHERWISE}
\end{align*}
\]

\( f \) takes two arguments, each of which is a pair. It expects the first element of each pair to be the same string, and each second element should either be a witness for that string or the string itself. Note that testing \( w \in W(x) \) can be done in polynomial time.
Theorem 2.16 cont’d.

We argue that \( f \) has all of the properties required by Theorem 2.16:

- **\( f \) is total:** The third case maps all inputs that don’t match what the function expects to a garbage string.
- **\( f \) is 2-ary:** This is trivial; \( f \) takes two arguments.
- **\( f \) is commutative:** In all three cases, reordering the inputs does not affect the output:
  - In the first case, \( a = c \) and lexmin is not affected by the order of its inputs.
  - In the second, \( a = c \) and the function compares the set \( \{b, d\} \), so order is not important.
  - In the third, the output is fixed.
Theorem 2.16 cont’d.

- **$f$ is strongly noninvertible:**
  - **$f$ is s-honest:** For non-garbage cases, recall that witness strings are of polynomial length. For the garbage case, any input $\langle a, b \rangle$ can be paired with $\langle a1, b \rangle$ to produce the garbage string.
  - Assume $f$ is not strongly noninvertible. Therefore, a polynomial-time function can recover an input given the other and the output. Let $g$ be this function. For any $x \in L$, $g(\langle x, x \rangle, \langle x, x \rangle)$ must output $\langle x, w \rangle$ for some $w$ in $W(x)$. However, because witness testing is polynomial-time, this allows us to build a polynomial-time algorithm for $L$, but $L$ is in $NP - P$. Therefore, $f$ is strongly noninvertible.
Theorem 2.16 cont’d.

- \( f \) is **one-way**: Recall our definition of one-way:
  - \( f \) is **polynomial-time computable**: Witness checking is a polynomial-time operation.
  - \( f \) is **not polynomial-time invertible**: Much like the strong noninvertibility argument, inverting the output \( \langle x, x \rangle \) will give us a witness for \( x \) if it exists, allowing a polynomial-time test of membership in \( L(N) \).
  - \( f \) is **honest**: The non-garbage cases are easy: witness strings are polynomially-bounded in length, and our pairing function is polynomial-time in both directions. Because the garbage case has a fixed output, we can pick any input producing it and choose our bounding polynomial to be large enough to cover that one case.
f is associative: We must show that
\[ f(f(z, z'), z'') = f(z, f(z', z'')). \]
There are multiple cases:

- At least one of \( z, z', z'' \) is not of the form \( ⟨x, x⟩ \) or \( ⟨x, w⟩ \) for some \( w ∈ W(x) \): That input will produce the garbage string when passed to \( f \), and the garbage string will also produce itself when passed to \( f \), so the equality holds.

- Exactly two are of the form \( ⟨x, x⟩ \): If both are passed to \( f \), the result is the garbage string. If one is combined with the third input (of form \( ⟨x, w⟩ \)), that result will be \( ⟨x, x⟩ \) by case 2 of \( f \)'s definition, and so the final call will again be with two instances of \( ⟨x, x⟩ \), producing the garbage string.

- All three are of the form \( ⟨x, x⟩ \): Like in the previous case, all results will be the garbage string.
Theorem 2.16 cont’d.

- **f** is associative (continued):
  - The first elements of the three inputs are not all identical: The non-garbage cases require that \( a = c \), so both results will be the garbage string.
  - Exactly two are of the form \( \langle x, w \rangle \) for some \( w \in W(x) \): Note that the witness strings can be distinct, but don’t need to be. The third input, of the form \( \langle x, x \rangle \), will produce \( \langle x, x \rangle \) when passed with either other input. If the two \( \langle x, w \rangle \) inputs are passed together, they will produce as output whichever of them has the lexicographically lesser witness as its second element, and \( \langle x, x \rangle \) will still be produced when combined with the third input.
Theorem 2.16 cont’d.

- **f** is associative (continued):
  - *All three are of the form* \(\langle x, w \rangle\): Again, the witness strings need not be identical, but may be. Here, all calls to \(f\) will fall into the first case, which will preserve the lexicographically lesser of the two passed witnesses. Because \(\text{lexmin}\) is itself associative, both results in this case will be \(\langle x, w_0 \rangle\), where \(w_0\) is the lexicographically least of all three witness strings.

Therefore, \(f\) is a strongly noninvertible, total, commutative, associative, 2-ary one-way function.
Theorem 2.16 cont’d.

By assuming that $P \neq NP$, we constructed function $f$ and proved that it has all of the properties required by Theorem 2.16. Because the question of whether $P \neq NP$ is equivalent to the question of whether one-way functions exist, we have proven the ”only if” direction. Recall that the “if” direction is trivial. Therefore, we have proven Theorem 2.16.