Two Party Computation

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Alex Ledger

May 2015

Approved for the Division
(Mathematics)

_____

Adam Groce

Acknowledgements

I want to thank a few people.

Preface

This is an example of a thesis setup to use the reed thesis document class.

List of Abbreviations

You can always change the way your abbreviations are formatted. Play around with it yourself, use tables, or come to CUS if you'd like to change the way it looks. You can also completely remove this chapter if you have no need for a list of abbreviations. Here is an example of what this could look like:

ABC	American Broadcasting Company
CBS	Columbia Broadcasting System
CDC	Center for Disease Control
CIA	Central Intelligence Agency
CLBR	Center for Life Beyond Reed
CUS	Computer User Services
FBI	Federal Bureau of Investigation
NBC	National Broadcasting Corporation

Table of Contents

List of Tables

List of Figures

Abstract

The preface pretty much says it all.

Dedication

You can have a dedication here if you wish.

Introduction

Introduction goes here

Chapter 1

Background

Multiparty computation (MPC) is the study and creation of protocols for computing a function between multiple parties, such that no party learns the input of any other party.

The idea is best communicated through an example: suppose Alice and Bob are millionaires and wish to determine who is wealthier, but Alice and Bob are also secretive, and do not want to disclose their exact amount of wealth. Is there some method by which they can determine who has more money?

The goal of MPC is to design a protocol which will help Alice and Bob solve their problem. The desired properties of a secure MPC scheme can be informally described as follows:

- **Privacy:** Each party's input is kept secret.
- **Correctness:** The correct answer to the computation is computed.

Originally, the goal was to come up with a protocol that was secure and prove that the protocol was secure. In more recent times, the focus has shifted to making the MPC faster, fast enough that it can be used regularly in the real world.

If MPC can be made fast enough, it could serve a wide range of applications. For example, imagine that two companies who operate in a similar industry want to work together, but they don't want to disclose any company research which the other doesn't know. These companies could a run set intersection function (a function that given two inputs finds their intersection, or overlap), to determine what information they can disclose without giving away important information.

Another interesting example of MPC is to improve the outsourcing of computation. As it is right now, cloud computing companies, such as Amazon and Google, have really nice computers which they will rent out to you. You can pay them someone money, write a program, and run it on their computers. The problem is that you may not trust the cloud computing company, and you want some guarantees that they are going to respect the privacy of your computation. An MPC protocol, in this setting, would allow you to run computation in the cloud, with the guarantee that the inputs to your computation are disguised.

Since the research into MPC has focused on creating a method by which an arbitrary function can be computed securely, the application of MPC beyond what we

can presently conceive of. It's not unlikely that MPC protocols will become a standard in the internet, where when you access the internet, behind the scenes your access is being is plugged into an MPC protocol, sent off to another computer to do some processing. As cryptography improves, research in MPC and other areas of cryptography, the hope is that the security of our computer systems will improve as well. However, there is no guarantee. The modern cryptography needs to implemented and used, perhaps in some cases built into low-level standards, and used correctly. At this point, the outlook of cryptography is bright, but the future will only be realized positvely if it is actively worked towards.

1.1 Tools for MPC

Imagine again that millionaires Alice and Bob wish to determine who has more wealth. Suppose that Alice has x_a dollars and Bob has y_b dollars. Then the function that they wish to compute is $f : \{0,1\}^n \to \{0,1\}$ as defined by:

$$f(x_a, x_b) = \begin{cases} 0, & x_a \leq x_b; \\ 1, & \text{otherwise.} \end{cases} \tag{1.1}$$

Why is this here? If Alice and Bob successfully compute $f(x, y)$ and each gets the output, then each party has gained some knowledge about the nature of the other's input. For example, if $f(x_a, x_b)$ outputs 0, then Alice and Bob know that Alice has more money. Alice has learned that Bob has less than x dollars, and Bob has learned that Alice has more x_b dollars. Therefore it's not possible for MPC to guarantee that *no* information about the other parties' inputs is learned, only that nothing more is learned than what can be inferred from a single party's input and the output of f.

MPC requires a combination of several cryptographic tools. The next few sections will describe these basic tools, and then MPC will described in section x. tools discussed are

1. Boolean Circuit
2. Encryption
3. Oblivious Transfer (need to explain semihonest and honest)

1.1.1 Encryption

Encyprtion is the process of encoding a message such that only parties with the key can read the message. An encryption protocol is composed of two parts. The first part is the encryption function which disguises the message, and the second part is the decryption function which unobfuscates the disguised message. We notate encryption and decryption with

$$\begin{aligned} \text{Enc}_k(pt) &= ct \\ \text{Dec}_k(ct) &= pt \end{aligned} \tag{1.2}$$

where *pt* stands for plaintext and is the original message, *ct* stands for ciphertext and is the encrypted message, k is the secret key, that only authorized readers of the message hold, and is a random sequence of λ 0s and 1s, and λ is a security parameter of our protocol (As λ increases, the size of the key increases, and so the encryption becomes harder to break.)

The encryption decribed above is more precisely called symmetric-key encryption, as opposed to public-key encryption. The difference between the two is that for symmetric-key encryption, there is a single key and all communicating must have the same key in order to acheive secure communication, whereas in public-key encryption, the encryption key is published publicly, and anyone can send a message using the public key, and the message can only be decrypted by those with the secret key. For more information on encryption, we encourage the reader to peruse the many great online resources.

The MPC protocols that will be examined here exclusively use symmetric-key encryption.

1.1.2 Digital Signatures/MACS

kA Do we need this?

1.1.3 Boolean Circuit

A function for an MPC protocol is represented by a boolean circuit. A boolean circuit takes as input a sequence of n 0s and 1s, (i.e. a value in $\{0,1\}^n$), performs a series of small opreations on the inputs, and outputs a sequence of m 0s and 1s (i.e. a value in $\{0,1,\}$). You may have encountered circuits and logical operators in another context, where the inputs and outputs were True and False. For our usage, True will correspond to the value 1, and False will coresond to the value 0.

The small operations performed inside of a circuit are performem by an object called a *gate*. A gate is composed of three wires: two input wires and one output wire, where a *wire* can have a value either 0 or 1. A gate performs a simpler operation on the two inputs, resulting in a single output bit. Table 1.1.3 gives the mapping of an XOR gate.

x	y	xor(x,y)
1	1	0
1	0	1
0	1	1
0	0	0

Table 1.1: The mapping of an XOR gate.

A circuit is a combination of gates. In fact, a circuit built out of only AND gates and XOR gates can compute any function. **Find details and citation** In other words, if there's some algorithm that do it, then there is some circuit that can do it

as well. Hence, a circuit is a sufficient representation of the function f. Figure 1.1.3 shows the circuit representation of a circuit that computes the less than function, the function f, specified in equation 1.1 that millionaires Alice and Bob wanted to compute.

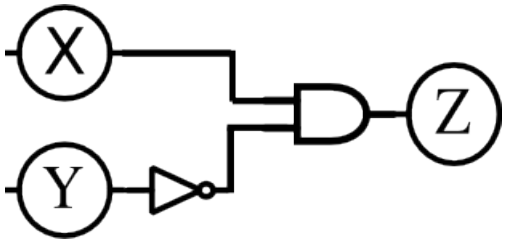

Figure 1.1: A circuit that computes the less or equal to function, equivalent to f for input of two one-bit values. **TODO, add truth table?**

A circuit **with what constraints???** can compute any function.

1.1.4 Technical Definition of a Circuit

A circuit, which I will refer to as g, is formalized by a 6-tuple $g = (n, m, q, A, B, G)$, where n is the number of inputs, m is the number of outputs and q is the number of gates. We define $r = n+q$ to the number of wires inside of the circuit. We let Wires $= \{1, \ldots, n+q\}$, InputWires $= \{1, \ldots, n\}$, OutputWires $= \{n+1-m+1, \ldots, n+q\}$, and Gates $= \{n+1, \ldots, n+q\}$. Then A : Gates $\rightarrow$ Wires $\setminus$ OutputWires identifies each gate's first incoming wire. And B : Gates $\rightarrow$ Wires $\setminus$ OutputWires identifies each gate's second incoming wire. Finally, G : Gates $\times \{0,1\}^2 \rightarrow \{0,1\}$ identifies the functionality of each gate.

For example, the less than circuit shown in figure 1.1.3 has values:

¡¡¡¡¡¡¡ HEAD

To evaluate a circuit, we use a canonical algorithm ev_{circ}. ev_{circ} takes as input a string f and a string $x = x_1 x_2 \ldots x_n$ and does the following:

Algorithm 1 ev_{circ}

Input: Function f and String $x = x_1, \ldots x_n$.
Output: Output of function f on input x.

```
(n, m, a, A, B, G) ← f                    ▷ parse f as a circuit
for g = n + 1 to n + q do                 ▷ loop over all gates.
    a ← A(g)                              ▷ compute each gate
    b ← B(g)
    x_g ← G_g(x_a, x_b)
end for
return x_{n+q−m+1} ... x_{n+q}
```

Definition 1 Topological Circuit A topological circuit, denoted f^{-}, is the 5-tuple

(n, m, q, A, B) for some circuit $f = (n, m, q, A, B, G)$. Furthermore, define $\text{Topo}(f)$ such that $f^- = \text{Topo}(f)$. ♢

A topological circuit is the same as its conventional circuit f, except that the functionality of the gates is not specified. If one holds a topological circuit, then they know the structure of the function, but not what is being computed. =======

1.1.5 Encryption

Encryption is a method of encoding a message such that only parties with the key can read the message. An encryption protocol is composed of two parts. The first part is the encryption function which disguises the message, and the second part is the decryption function which unobfuscates the disguised message. We notate encryption and decryption with

$$\begin{aligned} \text{Enc}_k(pt) &= ct \\ \text{Dec}_k(ct) &= pt \end{aligned} \tag{1.3}$$

where pt stands for plaintext and is the original message, ct stands for ciphertext and is the encrypted message, k is the secret key, that only authorized readers of the message hold, and is a random sequence of λ 0s and 1s, and λ is a security parameter of our protocol (As λ increases, the size of the key increases, and so the encryption becomes harder to break.)

The encryption decribed above is more precisely called symmetric-key encryption, as opposed to public-key encryption. The difference between the two is that for symmetric-key encryption, there is a single key and all communicating must have the same key in order to acheive secure communication, whereas in public-key encryption, the encryption key is published publicly, and anyone can send a message using the public key, and the message can only be decrypted by those with the secret key. For more information on encryption, we encourage the reader to peruse the many great online resources.

The MPC protocols that will be examined here exclusively use symmetric-key encryption.

1.1.6 Digital Signatures/MACS

Do we need this?

1.1.7 The DDH Assumption

When a cryptographic scehe is said to be *secure*, cryptographers actually mean something much more precise. When a cryptographic scheme is considered secure, it actually means that an adversary can beat the scheme only if the adversary can tackle the hardness assumptions on which the scheme is based. A common hardness assumption in crytoraphy is the Decisional Diffie-Helman Assumption (DDH Assumption), an assumption about solving a problem concerning discrete logs.

Alice				Bob		
Secret	Public	Calculus		Secret	Public	Calculus
m_0, m_1		Messages to be sent				
d	N, e	Generate RSA key pair and send public portion to Bob	$\Rightarrow$		N, e	Receive public key
	x_0, x_1	Generate two random messages	$\Rightarrow$		x_0, x_1	Receive random messages
				k, b		Choose $b \in \{0, 1\}$ and generate random k
	v		$\Leftarrow$		$v = (x_b + k^e) \mod N$	Compute the encryption of k, blind with x_b and send to Alice
$k_0 = (v - x_0)^d \mod N$ $k_1 = (v - x_1)^d \mod N$		One of these will equal k, but Alice does not know which.				
	$m_0' = m_0 + k_0$ $m_1' = m_1 + k_1$	Send both messages to Bob	$\Rightarrow$		m_0', m_1'	Receive both messages
				$m_b = m_b' - k$		Bob decrypts the m_b' since he knows which x_b he selected earlier.

Figure 1.2: Semi-honest Naor-Pinkas oblivious transfer.

Informally, the DDH assumption is:

$$\begin{aligned} &\text{Let } G \text{ be a group of order } q \text{ with generator } g. \\ &\text{Let } a, b \text{ and } c \text{ be random elements from } \mathbb{Z}_q. \\ &\text{Then, } (g^a, g^b, g^{ab}) \approx_D (g^a, g^b, b^c). \end{aligned} \tag{1.4}$$

Define computationally indistinguishability somewhere

1.1.8 Oblivious Transfer

Oblivious Transfer is a special method of communicating a message between two parties where a sender sends one of two messages the receive, and the sender remains oblivious as which message was sent. The setup is the following: Alice has two messages, m_1 and m_2, and he wants to send a message to Bob under the following conditions: First, Alice sends either m_1 or m_2 but not both. Second, Alice does not know which message he sent to Bob. Third, Bob selects which message he wants to receive.

Here we give the Naor-Pinkas protocol of $1-2$ oblivious transfer. The protocol relies on the DDH assumption (see section 1.1.7) and is secure in the semi-honest setting (see section **??**).

Researchers have also developed methods for performing k-out-of-n oblivious transfer, where the sender sends exactly k messages out of a possible n. The method described above is called $1-2$ oblivious transfer, and is the OT used in MPC protocols.

Since the first proposal of OT in **year**, several improvements have been developed. Preprocessing. OT extension. Semi-honest/Malicious.

1.1.9 Other

TODO Paper has topological circuit next to real circuit change arbitrary function to arbitrary boolean function.

1.2 Classic MPC

MPC was first proposed by Andrew Yao in an oral presentation on secure function evaluation. After Yao's presentation, two methods for performing MPC were developed. One method was contributed by Yao himself, and the other was contributed by a group of researchers, Beaver, Micali and Widgerson, (GMW). The two methods are premised on a similar idea: encrypt a circuit by encrypting its gates, with has since been termed garbled circuit. At this point, it is unclear which method is better, both in terms of security and in terms of speed speed. As a result, research is still being done on both protocols.

This section will first give a new definition of a garbling scheme, first presented by BHR in 2012, and then give Yao's garbled circuit protocol, and finally GMW's garbled circuit protocol.

1.2.1 Garbling Schemes

Recall millionaires Alice and Bob who want to determine who has more wealth. Alice and Bob start with a function f (defined in **??**) and their inputs, x_a and x_b, which corresond to their amount of wealth. In order to compute who has more money, the function f needs to be transformed such that they can compute the function by sending messages between each other.eir inputs, x_a and x_b, which corresond to their amount of wealth. Informally, a garbling scheme transforms a specification of a function into a collection of algorithms which can in combination can compute the function.

Definition 2 Garbling Scheme A garblering scheme is a 5 tuple of algorithms $G = (\mathrm{Gb}, \mathrm{En}, \mathrm{De}, \mathrm{Ev}, \mathrm{ev})$ where the algorithms are

- $\mathrm{Gb}(f, k) \to (F, e, d)$.
- $\mathrm{En}(e, x) \to X$.
- $\mathrm{Ev}(F, X) \to Y$.
- $\mathrm{De}(d, Y) \to y$.
- $\mathrm{ev}(f, x) \to y$. where
  - f is a function.
  - $k \in \mathbb{N}$ is the security parameter.
  - $x \in \{0,1\}^n$ is the initial input.
  - X is the garbled input
  - Y is the garbled output
  - y is the final output
  - F is a string that describes the operation of the Ev algorithm.

  - e is a string describing the operation of the En algorithm. In particular, e describes how to obscure the input x.
  - d is a string describing the operation of the De algorithm. In particular, d describes how to unobscure the output Y.

- A garbling scheme guarantees correctness if $\mathrm{De}(d, \mathrm{Ev}(F, \mathrm{En}(e, x))) = \mathrm{ev}(f, x)$.

A garbling schemes map the information:

$$(f, k, x) \to^{\mathrm{Gb}} (F, e, d, x) \to^{\mathrm{En}} (F, d, X) \to^{\mathrm{Ev}} (d, Y) \to^{\mathrm{De}} y \tag{1.5}$$

♢

To make the idea a garbling scheme more concrete, below is an example of how Alice and Bob would compute who is wealthier. The inputs to their computation are f, the function to be computed which is defined in equation 1.1, and their inputs x_a and x_b.

1. Alice has the function f to be computed, her input x_a, and Bob has his input x_b.
2. Alice hard-codes her inputs into the function, resulting in $\hat{f}$.
3. Alice runs the probabilistic algorithm $\mathrm{Gb}(\hat{f}, k)$. She now has (F, e, d).
4. Alice encodes all possible inputs that Bob could, all possible sequences of 0s and 1s, into X using En.
5. Alice sends (F, X, d) to Bob.
6. Bob runs $\mathrm{Ev}(F, X)$ to get Y. This computes the function, but the output, Y, is obscured.
7. Bob runs $\mathrm{De}(d, Y)$ to get y. This unobscure the output Y. Bob now has $y \in \{0, 1\}$. If $y = 0$, then Bob knows that Alice is wealthier. If $y = 1$, then Bob knows that he is wealthier.
8. Bob sends y to Alice.

The definition of a garbling scheme is a recently created definition, proposed for the first time ****, *** years after Yao's garbled circuit was propsed. The benefits of thinking of a garbled scheme in terms of this defintion are signficant. For one, it easier to map a theoretically garbling protocol into practice, i.e program it. The methods are already separated, whereas before the algorithm was generally one giant idea. When the methods are separated, the programmer does not need to intimately understand the security guarantees and pitfalls of the method, as the sequence of events that need to happen is explicit in the construction. A second reason that this definition of a garbling scheme is good is that it presents garbling as a larger cryographic abstraction. Garbling schemes are simply methods for garbling a function and input, and this definition matches the intuition.

1.2.2 Projective Garbling Schemes

Perhaps a helpful abstraction to have later.

1.2.3 Security of Garbling Schemes

This section will give the definition of security of a garbling scheme.

1.2.4 MPC Security Definitions

A number of definitions which formalize what it means for a mulitparty computation have been formalized. Yao, in his 1982 paper on secure function evaluation, says a protocol is secure if it has the property:

> If one participant behaves according the the protocol, the probability that the other participant successfully cheats is at most γ for $\gamma \in \{0, 1\}$.

As more complex protocols were developed, both for MPC, and in other areas of cryptography, property based definitions became difficult to work with. The main problem that it was unclear precisely what assumptions were being made about what each party could and couldn't do. For example, in Yao's definition above, it's not clear that the parties' inputs are kept secret if both parties behave according to the protocol. At the least, the definition doesn't give any guarantee. For a while, cryptographers added in additional properties to definitions, but eventually it became clear that there was always the possibility that a certain property was being missed.

This led to the creation to indistinguishability and simulation based definitions of security. The idea is that the adversary cannot tell what is going on, based on the information that they receive, so the best that the adversary can possible do is guess what is happening. For example, a definition of a secure encryption scheme could be: **Might give the definition of encryption in the section above. If we do, just reference that section. That might be preferable**

- Adversary picks two messages, m_0 and m_1.
- We pick a random bit $b \in \{0, 1\}$. Encrypt m_b, and send the encrypted version to the adversary (i.e. send $c = \text{Enc}(m_b)$).
- Adversary outputs a bit b' where $b' = 0$ if they believe the encrypted messages was m_0, and $b' = 1$ if they think the encrypted message was m_1.
- Finally, we consider the encryption scheme secure if

$$Pr[\text{Adversary outputs } b' = b] \approx_C \frac{1}{2}.$$

The cryptography research community now tends to prefer simulation based definitions. The idea behind simulation based definitions is that we have ideal world, in which the protocol is run perfectly and securely, and then we have the real world

protocol, the one that Alice and Bob run. Then, for some adversary, we randomly choose either the real ideal or real world and run the MPC computation with the adversary in that world. The adversaries attempts to determine which world the protocol is being run in. Finally, the protocol is secure if the adversary can't do any better than guessing which world they are in. As a consequence, the real world is indistinguishable from the ideal world, hence cryptographers feel confident that the protocol is secure. The idea being that if the adversary could learn more information in the real world, say information about other parties' inputs, then they would know that they are in the real world, hence the protocol is insecure.

The biggest problem with simulation based definitions is that it is difficult and labor intensive to prove with the definition. The property based definitions are the most straightforward to prove, and the most familiar to proofs used in mathematics.

1.2.5 Goldreich's Definition of Security for 2PC

Here we give Goldreich's definition of MPC security, which is presented in his textbook, *Foundations of Cryptography Volume II* **?**. The definition is designed for static, semi-honest adversaries.

Setup:

- Let $f = (f_1, f_2)$ be a probabilistic, polynomial time functionality.
- Let Π be a two party protocol for computing f.
- Define $\mathsf{view}_i^{\Pi}(n, x, y)$ (for $i \in \{1, 2\}$) as the view of the ith party on input (x, y) and security parameter n. $\mathsf{view}_i^{\Pi}(n, x, y)$ equals the tuple $(1^n, x, r^i, m_1^i, \ldots, m_t^i)$, where r^i is the contents of the ith party's internal random tape, and m_j^i is the jth message that the ith party received.
- Define $output_i^{\Pi}(n, x, y)$ as the output of the ith party on input (x, y) and security parameters n. Also denote

$$\mathsf{output}^{\Pi}(n, x, y) = (\mathsf{output}_1^{\Pi}(n, x, y), \mathsf{output}_2^{\Pi}(n, x, y)).$$

- Note that view_i^{Π} and output_i^{Π} are random variables whose probabilities are taken over the random tapes of the two parties. Also note that for two party computation.

Definition: We say that Π securely computes f in the presence of static semi-honest adversaries if there exists probabiliistic polynomial time algorithms S_1 and S_2 such that for all $x, y \in \{0, 1\}^*$, where $|x| = |y|$, the following are true:

$$\{(S_1(x, f_1(x, y), f(x, y)))\}_{x,y} \equiv^C \{(\mathsf{view}_1^{\Pi}(x, y), \mathsf{output}^{\Pi}(x, y))\}_{x,y} \tag{1.6}$$

$$\{(S_2(x, f_2(x, y), f(x, y)))\}_{x,y} \equiv^C \{(\mathsf{view}_2^{\Pi}(x, y), \mathsf{output}^{\Pi}(x, y))\}_{x,y} \tag{1.7}$$

Intuition: We think of view_i^Π as all of the information that the ith has to operate with, such that any conclusion that the ith party can come to could be determined from $view_i^\Pi$. Moreover, output_i^Π is simply a complicated way of writing the output of the ith party. The value of output_i^Π is computable from the tuple view_i^Π.

Let's dig a little deeper into the meanings of equation 1.6 and 1.7. They state that a proabilistic, polynomial time algorithm, denoted S_1 and S_2, which is given access *only* to the party's input and output can compute the view of a party. For example, the defintion requires that S_1 on input $(x, f(x, y))$ must be able to compute $\mathsf{view}_i^\Pi(x, y)$, in particular the messages received by party 1, such that the generated view is indistinguishable from the actual view. If there exists an algorithm that can perform the aforementioned task, then Π does not adequately conceal information, so we should not consider Π to be secure.

Finally, the definition requires that $|x| = |y|$; however, this constraint can be overcome in practice by making the shorter input longer by adding padding.

The definition provided here is only secure when adveraries are semi-honest (see **??**).DDefinitions of security in settings with malicious adversaries require substantially more complexity. As a result, these definitions are often simulation based definitions. They imagine an ideal world, where the function f must be computed securely, and by a series of comparisons, show that the real world where Π computes f is essentially the same as the ideal world. For an easy to understand security definition of 2PC with malicious adversaries, we refer to reader to **?**.

1.2.6 Yao's Garbled Circuit

We now give an implementation of a generic 2PC protocol created by Yao **?**. The protocol works for two parties; we will call party 1 Alice, denoted A, and party 2 Bob, denoted B, who have inputs x and y respectively. Suppose f is the function that Alice and Bob wish to compute.

Yao's protocol depends on first encoding the function f as a circuit, as discussed in more detail in section **??**, and then Alice and Bob together evaluate the circuit.

Step 0: Setup

Alice and Bob want to compute the function $f(x, y)$, where $x, y \in \{0, 1\}^*$. Alice is the garbler, and will create the circuit. Bob is the evaluator, and will compute the circuit. Alice first hardwires her inputs into the circuit, yielding a circuit that computes $f(x, \cdot)$.

Step 1: Garbling the Circuit

For each wire w_i in the circuit,

For every gate g in the circuit, say that g has input wire w_i and w_j and output wire w_k, Alice creates a table T_g. T_g encrypts the output wire using the input wires

Algorithm 2 Garble Circuit

Input: Circuit $f(x, \cdot)$
Output: Populate garbled tables $f(x, \cdot).tables$.
  for wire w_i in $f(x, \cdot).wires$ **do**
    Generate two encryption keys, called garbled values, W_i^0 and W_i^1.
    Assign (W_i^0, W_i^1) to w_i.
  end for

  for gate g in $f(x, \cdot).gates$ **do**
    Let w_i be g's first input wire.
    Let w_j be g's second input wire.
    Let w_k be g's output wire.
    for $(u, v) \in \{(0,0), (0,1), (1,0), (1,1)\}$ **do**
      $T_g(u, v) = \text{Enc}_{W_i^u}(\text{Enc}_{W_j^v}(W_k^{g(u,v)}))$
    end for
    $f(x, \cdot).tables[g] = T_g$.
  end for

as keys. The table enables the computation of the gate g given either K_i^0 or K_i^1 and either K_j^0 or K_j^1. Algorithm 2 gives the pseudocode for Alice's garbling. The algorithm essentially produces keys, and encrypts keys using other keys. Figure **??** gives an example of a table for an AND gate. **TODO**

Step 2: Bob's Input

Alice sends the garbled circuit to bob, which consists of the garbled tables, $f(x, \cdot).tables$, and the rules for connecting the gates together. In order for Bob to compute the circuit, he needs garbled values of all input wires. Once he has the garbled values of the input wires, he can decrypt the first few gates, and acquire the decryption keys of the other gates, until he has the keys to decrypt all of the gates in the circuit, yieliding the outp. Bob can acquire the garble values of the input wires from Alice using 1-out-of-2 oblivious transfer on each input wire. Section **??** explains oblivious transfer.

Step 3: Computing the Circuit

After receiving the garbled values of the input wires, Bob decrypts the input gates, works through the rest of the circuit, decrypting gates as he goes, until he finally receives the output. At this point, Bob has learned $f(x, y)$, but Alice has learned nothing. To complete the protocol, Bob sends Alice $f(x, y)$, the functionality requires it (i.e. if $f_1(x, y) = f_2(x, y)$).

Explanation of the security of Yao's protocol

To do.

Notes about complexity

1 OT per (input?) wire. How much encryption? Not good enoug hfor practice.

1.2.7 GMW

Algorithm 3 Garble Circuit

Input: Circuit $f(x, \cdot)$

For Person 1:

for wire w_i in $f(x, \cdot).wires$ **do**

Assign $a^1_{w_i} \leftarrow \{0, 1\}$ ▷ a uniform random selection of 0 or 1.

Assign $b^1_{w_i} = x_i a^1_{w_i}$

end for

1.3 Improving MPC

1.3.1 Overview of history

working on specific function verse general functions working on number of parties making it work for sugarbeet auction election potential? auctions

1.3.2 Point and Permute

1.3.3 Garbled Row Reduction 3

1.3.4 Free XOR

1.3.5 Garbled Row Reduction 2

1.3.6 FlexXOR

1.3.7 Half Gates

1.4 Overview of what's going to be covered

Mode Transition Times Duration (Typical)	**Size (xλ)**		**Eval Cost**		**Garble Cost**		**Assumption**
	XOR	**AND**	**XOR**	**AND**	**XOR**	**AND**	
Classical	1365	1260	1024	μ s	a	b	a
Point and Permute	TBD	TBD	TBD	μ s	a	b	a
GRR3	TBD	TBD	TBD	μ s	a	b	a
Free XOR	TBD	TBD	TBD	μ	a	bs	a
GRR2 XOR	TBD	TBD	TBD	μ	a	bs	a
FleXOR	TBD	TBD	TBD	μ	a	bs	a
Half Gates	TBD	TBD	TBD	μ	a	bs	a

Table 1.2: Summary of Garbled Circuit Improvements. GRR3 stands for garbled row reduction 3 and GRR2 stands for garbled row reduction 2

Chapter 2

The First

This is the first page of the first chapter. You may delete the contents of this chapter so you can add your own text; it's just here to show you some examples.

2.1 References, Labels, Custom Commands and Footnotes

It is easy to refer to anything within your document using the `label` and `ref` tags. Labels must be unique and shouldn't use any odd characters; generally sticking to letters and numbers (no spaces) should be fine. Put the label on whatever you want to refer to, and put the reference where you want the reference. LaTeX will keep track of the chapter, section, and figure or table numbers for you.

2.1.1 References and Labels

Sometimes you'd like to refer to a table or figure, e.g. you can see in Figure 4.2 that you can rotate figures . Start by labeling your figure or table with the label command (`\label{labelvariable}`) below the caption (see the chapter on graphics and tables for examples). Then when you would like to refer to the table or figure, use the ref command (`\ref{labelvariable}`). Make sure your label variables are unique; you can't have two elements named "default." Also, since the reference command only puts the figure or table number, you will have to put "Table" or "Figure" as appropriate, as seen in the following examples:

As I showed in Table 4.1 many factors can be assumed to follow from inheritance. Also see the Figure 4.1 for an illustration.

2.1.2 Custom Commands

Are you sick of writing the same complex equation or phrase over and over?

The custom commands should be placed in the preamble, or at least prior to the first usage of the command. The structure of the `\newcommand` consists of the name of the new command in curly braces, the number of arguments to be made in square

brackets and then, inside a new set of curly braces, the command(s) that make up the new command. The whole thing is sandwiched inside a larger set of curly braces.

In other words, if you want to make a shorthand for H_2SO_4, which doesn't include an argument, you would write: `\newcommand{\hydro}{H$_2$SO$_4$}` and then when you needed to use the command you would type `\hydro`. (sans verb and the equals sign brackets, if you're looking at the .tex version). For example: H_2SO_4

2.1.3 Footnotes and Endnotes

You might want to footnote something.[1] Be sure to leave no spaces between the word immediately preceding the footnote command and the command itself. The footnote will be in a smaller font and placed appropriately. Endnotes work in much the same way. More information can be found about both on the CUS site.

2.2 Bibliographies

Of course you will need to cite things, and you will probably accumulate an armful of sources. This is why BibTeX was created. For more information about BibTeX and bibliographies, see our CUS site (`web.reed.edu/cis/help/latex/index.html`)[2]. There are three pages on this topic: *bibtex* (which talks about using BibTeX, at `/latex/bibtex.html`), *bibtexstyles* (about how to find and use the bibliography style that best suits your needs, at `/latex/bibtexstyles.html`) and *bibman* (which covers how to make and maintain a bibliography by hand, without BibTeX, at at `/latex/bibman.html`). The last page will not be useful unless you have only a few sources. There used to be APA stuff here, but we don't need it since I've fixed this with my apa-good natbib style file.

2.2.1 Tips for Bibliographies

1. Like with thesis formatting, the sooner you start compiling your bibliography for something as large as thesis, the better. Typing in source after source is mind-numbing enough; do you really want to do it for hours on end in late April? Think of it as procrastination.

2. The cite key (a citation's label) needs to be unique from the other entries.

3. When you have more than one author or editor, you need to separate each author's name by the word "and" e.g.
   `Author = {Noble, Sam and Youngberg, Jessica},.`

4. Bibliographies made using BibTeX (whether manually or using a manager) accept LaTeX markup, so you can italicize and add symbols as necessary.

[1] footnote text
[2] **?**

5. To force capitalization in an article title or where all lowercase is generally used, bracket the capital letter in curly braces.

6. You can add a Reed Thesis citation[3] option. The best way to do this is to use the phdthesis type of citation, and use the optional "type" field to enter "Reed thesis" or "Undergraduate thesis". Here's a test of Chicago, showing the second cite in a row[4] being different. Also the second time not in a row[5] should be different. Of course in other styles they'll all look the same.

2.3 Anything else?

If you'd like to see examples of other things in this template, please contact CUS (email cus@reed.edu) with your suggestions. We love to see people using LaTeX for their theses, and are happy to help.

[3] **?**
[4] **?**
[5] **?**

Chapter 3

Mathematics and Science

3.1 Math

TEX is the best way to typeset mathematics. Donald Knuth designed TEX when he got frustrated at how long it was taking the typesetters to finish his book, which contained a lot of mathematics.

If you are doing a thesis that will involve lots of math, you will want to read the following section which has been commented out. If you're not going to use math, skip over this next big red section. (It's red in the .tex file but does not show up in the .pdf.)

$$\sum_{j=1}^{n}(\delta\theta_j)^2 \leq \frac{\beta_i^2}{\delta_i^2+\rho_i^2}\left[2\rho_i^2+\frac{\delta_i^2\beta_i^2}{\delta_i^2+\rho_i^2}\right] \equiv \omega_i^2$$

From Informational Dynamics, we have the following (Dave Braden):
After n such encounters the posterior density for θ is

$$\pi(\theta|X_1 < y_1, \ldots, X_n < y_n) \propto \pi(\theta)\prod_{i=1}^{n}\int_{-\infty}^{y_i}\exp\left(-\frac{(x-\theta)^2}{2\sigma^2}\right)\,dx$$

Another equation:

$$\det\begin{vmatrix} c_0 & c_1 & c_2 & \ldots & c_n \\ c_1 & c_2 & c_3 & \ldots & c_{n+1} \\ c_2 & c_3 & c_4 & \ldots & c_{n+2} \\ \vdots & \vdots & \vdots & & \vdots \\ c_n & c_{n+1} & c_{n+2} & \ldots & c_{2n} \end{vmatrix} > 0$$

Lapidus and Pindar, Numerical Solution of Partial Differential Equations in Science and Engineering. Page 54

$$\int_t\left\{\sum_{j=1}^{3}T_j\left(\frac{d\phi_j}{dt}+k\phi_j\right)-kT_e\right\}w_i(t)\,dt = 0, \qquad i = 1,2,3.$$

L&P Galerkin method weighting functions. Page 55

$$\sum_{j=1}^{3} T_j \int_0^1 \left\{ \frac{d\phi_j}{dt} + k\phi_j \right\} \phi_i \; dt = \int_0^1 k\, T_e \phi_i dt, \qquad i = 1, 2, 3$$

Another L&P (p145)

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta,\zeta) = \sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{i=1}^{n} w_i w_j w_k f(\xi,\eta,\zeta).$$

Another L&P (p126)

$$\int_{A_e} (\,\cdot\,) dx dy = \int_{-1}^{1}\int_{-1}^{1} (\,\cdot\,) \det[J] d\xi d\eta.$$

3.2 Chemistry 101: Symbols

Chemical formulas will look best if they are not italicized. Get around math mode's automatic italicizing by using the argument `$\mathrm{formula here}$`, with your formula inside the curly brackets.

So, $Fe_2^{2+}Cr_2O_4$ is written `$\mathrm{Fe_2^{2+}Cr_2O_4}$`
Exponent or Superscript: O^-
Subscript: CH_4

To stack numbers or letters as in Fe_2^{2+}, the subscript is defined first, and then the superscript is defined.
Angstrom: Å
Bullet: $CuCl \bullet 7H_2O$
Double Dagger: ‡
Delta: Δ
Reaction Arrows: $\longrightarrow$ or $\xrightarrow{solution}$
Resonance Arrows: $\leftrightarrow$
Reversible Reaction Arrows: $\rightleftharpoons$ or $\xrightleftharpoons{solution}$ (the latter requires the chemarr package)

3.2.1 Typesetting reactions

You may wish to put your reaction in a figure environment, which means that LaTeX will place the reaction where it fits and you can have a figure legend if desired:

$$C_6H_{12}O_6 + 6O_2 \longrightarrow 6CO_2 + 6H_2O$$

Figure 3.1: Combustion of glucose

3.2.2 Other examples of reactions

$NH_4Cl_{(s)} \rightleftharpoons NH_{3(g)} + HCl_{(g)}$
$MeCH_2Br + Mg \xrightarrow[below]{above} MeCH_2 \bullet Mg \bullet Br$

3.3 Physics

Many of the symbols you will need can be found on the math page (`http://web.reed.edu/cis/help/latex/math.html`) and the Comprehensive LaTeX Symbol Guide (enclosed in this template download). You may wish to create custom commands for commonly used symbols, phrases or equations, as described in Chapter 2.1.2.

3.4 Biology

You will probably find the resources at `http://www.lecb.ncifcrf.gov/~toms/latex.html` helpful, particularly the links to bsts for various journals. You may also be interested in TeXShade for nucleotide typesetting (`http://homepages.uni-tuebingen.de/beitz/txe.html`). Be sure to read the proceeding chapter on graphics and tables, and remember that the thesis template has versions of Ecology and Science bsts which support webpage citation formats.

Chapter 4

Tables and Graphics

4.1 Tables

The following section contains examples of tables, most of which have been commented out for brevity. (They will show up in the .tex document in red, but not at all in the .pdf). For more help in constructing a table (or anything else in this document), please see the LaTeX pages on the CUS site.

Table 4.1: Correlation of Inheritance Factors between Parents and Child

Factors	Correlation between Parents & Child	Inherited
Education	-0.49	Yes
Socio-Economic Status	0.28	Slight
Income	0.08	No
Family Size	0.19	Slight
Occupational Prestige	0.21	Slight

If you want to make a table that is longer than a page, you will want to use the longtable environment. Uncomment the table below to see an example, or see our online documentation.

Table 4.2: Chromium Hexacarbonyl Data Collected in 1998–1999

Chromium Hexacarbonyl			
State	Laser wavelength	Buffer gas	Ratio of $\frac{\text{Intensity at vapor pressure}}{\text{Intensity at 240 Torr}}$
$z^7P_4^\circ$	266 nm	Argon	1.5
$z^7P_2^\circ$	355 nm	Argon	0.57
$y^7P_3^\circ$	266 nm	Argon	1
$y^7P_3^\circ$	355 nm	Argon	0.14
$y^7P_2^\circ$	355 nm	Argon	0.14
$z^5P_3^\circ$	266 nm	Argon	1.2
$z^5P_3^\circ$	355 nm	Argon	0.04
$z^5P_3^\circ$	355 nm	Helium	0.02
$z^5P_2^\circ$	355 nm	Argon	0.07
$z^5P_1^\circ$	355 nm	Argon	0.05
$y^5P_3^\circ$	355 nm	Argon	0.05, 0.4
$y^5P_3^\circ$	355 nm	Helium	0.25
$z^5F_4^\circ$	266 nm	Argon	1.4
$z^5F_4^\circ$	355 nm	Argon	0.29
$z^5F_4^\circ$	355 nm	Helium	1.02
$z^5D_4^\circ$	355 nm	Argon	0.3
$z^5D_4^\circ$	355 nm	Helium	0.65
$y^5H_7^\circ$	266 nm	Argon	0.17
$y^5H_7^\circ$	355 nm	Argon	0.13
$y^5H_7^\circ$	355 nm	Helium	0.11
a^5D_3	266 nm	Argon	0.71
a^5D_2	266 nm	Argon	0.77
a^5D_2	355 nm	Argon	0.63
a^3D_3	355 nm	Argon	0.05
a^5S_2	266 nm	Argon	2
a^5S_2	355 nm	Argon	1.5
a^5G_6	355 nm	Argon	0.91
a^3G_4	355 nm	Argon	0.08
e^7D_5	355 nm	Helium	3.5
e^7D_3	355 nm	Helium	3
f^7D_5	355 nm	Helium	0.25
f^7D_5	355 nm	Argon	0.25
f^7D_4	355 nm	Argon	0.2
f^7D_4	355 nm	Helium	0.3
Propyl-ACT			

State	Laser wavelength	Buffer gas	Ratio of $\frac{\text{Intensity at vapor pressure}}{\text{Intensity at 240 Torr}}$
$z^7P_4^\circ$	355 nm	Argon	1.5
$z^7P_3^\circ$	355 nm	Argon	1.5
$z^7P_2^\circ$	355 nm	Argon	1.25
$z^7F_5^\circ$	355 nm	Argon	2.85
$y^7P_4^\circ$	355 nm	Argon	0.07
$y^7P_3^\circ$	355 nm	Argon	0.06
$z^5P_3^\circ$	355 nm	Argon	0.12
$z^5P_2^\circ$	355 nm	Argon	0.13
$z^5P_1^\circ$	355 nm	Argon	0.14
Methyl-ACT			
$z^7P_4^\circ$	355 nm	Argon	1.6, 2.5
$z^7P_4^\circ$	355 nm	Helium	3
$z^7P_4^\circ$	266 nm	Argon	1.33
$z^7P_3^\circ$	355 nm	Argon	1.5
$z^7P_2^\circ$	355 nm	Argon	1.25, 1.3
$z^7F_5^\circ$	355 nm	Argon	3
$y^7P_4^\circ$	355 nm	Argon	0.07, 0.08
$y^7P_4^\circ$	355 nm	Helium	0.2
$y^7P_3^\circ$	266 nm	Argon	1.22
$y^7P_3^\circ$	355 nm	Argon	0.08
$y^7P_2^\circ$	355 nm	Argon	0.1
$z^5P_3^\circ$	266 nm	Argon	0.67
$z^5P_3^\circ$	355 nm	Argon	0.08, 0.17
$z^5P_3^\circ$	355 nm	Helium	0.12
$z^5P_2^\circ$	355 nm	Argon	0.13
$z^5P_1^\circ$	355 nm	Argon	0.09
$y^5H_7^\circ$	355 nm	Argon	0.06, 0.05
a^5D_3	266 nm	Argon	2.5
a^5D_2	266 nm	Argon	1.9
a^5D_2	355 nm	Argon	1.17
a^5S_2	266 nm	Argon	2.3
a^5S_2	355 nm	Argon	1.11
a^5G_6	355 nm	Argon	1.6
e^7D_5	355 nm	Argon	1

4.2 Figures

If your thesis has a lot of figures, LaTeX might behave better for you than that other word processor. One thing that may be annoying is the way it handles "floats" like tables and figures. LaTeX will try to find the best place to put your object based on the text around it and until you're really, truly done writing you should just leave it where it lies. There are some optional arguments to the figure and table environments

to specify where you want it to appear; see the comments in the first figure.

If you need a graphic or tabular material to be part of the text, you can just put it inline. If you need it to appear in the list of figures or tables, it should be placed in the floating environment.

To get a figure from StatView, JMP, SPSS or other statistics program into a figure, you can print to pdf or save the image as a jpg or png. Precisely how you will do this depends on the program: you may need to copy-paste figures into Photoshop or other graphic program, then save in the appropriate format.

Below we have put a few examples of figures. For more help using graphics and the float environment, see our online documentation.

And this is how you add a figure with a graphic:

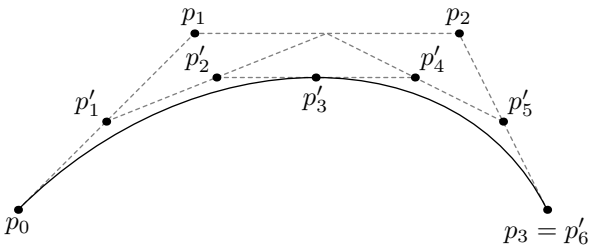

Figure 4.1: A Figure

4.3 More Figure Stuff

You can also scale and rotate figures.

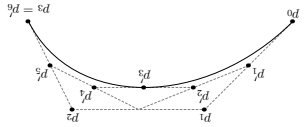

Figure 4.2: A Smaller Figure, Flipped Upside Down

4.4 Even More Figure Stuff

With some clever work you can crop a figure, which is handy if (for instance) your EPS or PDF is a little graphic on a whole sheet of paper. The viewport arguments are the lower-left and upper-right coordinates for the area you want to crop.

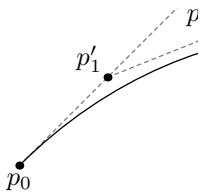

Figure 4.3: A Cropped Figure

4.4.1 Common Modifications

The following figure features the more popular changes thesis students want to their figures. This information is also on the web at `web.reed.edu/cis/help/latex/graphics.html`.

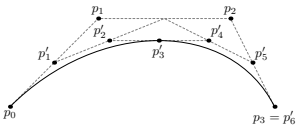

Figure 4.4: Subdivision of arc segments. You can see that $p_3 = p'_6$.

Conclusion

Here's a conclusion, demonstrating the use of all that manual incrementing and table of contents adding that has to happen if you use the starred form of the chapter command. The deal is, the chapter command in LaTeX does a lot of things: it increments the chapter counter, it resets the section counter to zero, it puts the name of the chapter into the table of contents and the running headers, and probably some other stuff.

So, if you remove all that stuff because you don't like it to say "Chapter 4: Conclusion", then you have to manually add all the things LaTeX would normally do for you. Maybe someday we'll write a new chapter macro that doesn't add "Chapter X" to the beginning of every chapter title.

4.1 More info

And here's some other random info: the first paragraph after a chapter title or section head *shouldn't be* indented, because indents are to tell the reader that you're starting a new paragraph. Since that's obvious after a chapter or section title, proper typesetting doesn't add an indent there.

Appendix A

The First Appendix

Appendix B

The Second Appendix, for Fun

References

Bellare, M., Hoang, V. T., Keelveedhi, S., & Rogaway, P. (2013). Efficient garbling from a fixed-key blockcipher. In *IEEE Symposium of Security and Privacy*, (pp. 478–492).

Bellare, M., Hoang, V. T., & Rogaway, P. (2012). Foundations of garbled circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, (pp. 784–796). ACM.

Lindell, Y., & Pinkas, B. (2009). Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, *1*(1), 5.