2011

A SECURE ONLINE PAYMENT SYSTEM

Shristi Pant
University of Kentucky, shristi.pant@gmail.com

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained and attached hereto needed written permission statements(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine).

I hereby grant to The University of Kentucky and its agents the non-exclusive license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless a preapproved embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's dissertation including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Shristi Pant, Student

Dr. Mukesh Singhal, Major Professor

Dr. Raphael A. Finkel, Director of Graduate Studies

A SECURE ONLINE PAYMENT SYSTEM

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science in the
College of Engineering at the University of Kentucky

By

Shristi Pant

Lexington, Kentucky

Director: Dr. Mukesh Singhal, Professor of Computer Science

Lexington, Kentucky

2011

ABSTRACT OF THESIS

A SECURE ONLINE PAYMENT SYSTEM

An online payment system allows a customer to make a payment to an online merchant or a service provider. Payment gateways, a channel between customers and payment processors, use various security tools to secure a customer's payment information, usually debit or credit card information, during an online payment. However, the security provided by a payment gateway cannot completely protect a customer's payment information when a merchant also has the ability to obtain the payment information in some form. Furthermore, not all merchants provide a secure payment environment to their customers and, despite having a standard payment policy, adhere to it. Consequently, this exposes a customer's payment information to risks of being compromised or misused by merchants or stolen by hackers and spammers. In this thesis we propose a new approach to payment systems in which a customer's payment information cannot be obtained by a merchant. A customer sends his payment information directly to a payment gateway and a payment gateway, upon verifying the transaction, sends a payment to the appropriate merchant. We use the Pedersen commitment scheme along with dual signatures to securely transfer funds to a merchant and protect a customer's payment information from any Internet vulnerabilities.

KEYWORDS: Online Payment Systems, Payment Gateways, Pedersen Commitment, Secure Electronic Transaction, Dual Signatures

Shristi Pant

September 2011

A SECURE ONLINE PAYMENT SYSTEM

By

Shristi Pant

Dr. Mukesh Singhal
Director of Thesis

Dr. Raphael A. Finkel
Director of Graduate Studies

November 15, 2011

Dedicated to my grand-mom and my parents

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my committee members, Dr. Dakshnamoorthy Manivannan, Dr. Zongming Fei, and especially, my advisor Dr. Mukesh Singhal. It would not have been possible for me to successfully complete my thesis without the guidance and expertise of my advisor. He inspired and motivated me all along my research with regular input and constant support. I would also like to thank my family for their love and encouragement throughout my master's program. Last but not the least, I thank god for showering his kind blessings on me.

Table of Contents

List of Figures

Chapter 1: Introduction

This thesis proposes a new approach to electronic payment in which a customer's payment information cannot be obtained by a merchant. A customer's payment information is usually a debit or credit card detail, and providing it to a merchant during e-payment exposes this sensitive financial information to various risks. Some of these widely known risks are data tampering, stealing credit card details and credit card fraud. A merchant may or may not exploit customer data but can definitely store it. In that case, if a merchant's server or system is not secure enough to prevent intrusion of data stealers, spammers, spyware, malware and hackers, customer data may be stolen and misused. Hence, to avoid the issue of data mishandling or unsecured data on the merchant side, we propose a payment method that does not send customer payment information to merchants and allows only payment gateways to deal with it. Payment gateways are secure and reliable, because they comply with the standard data security rules and communicate with banks and credit card companies using the most secure methods and technologies. To strengthen data security, we implement the Pedersen commitment scheme along with dual signatures in our proposed online payment system.

The buying and selling of goods and services over the Internet is known as electronic commerce (e-commerce). The concept of e-commerce is, however, not just limited to buying and selling of goods. It also includes the entire purchase process of developing, marketing, selling, delivering, servicing and paying for products and services [6].

With the development of e-commerce, payment systems and protocols have been developed. The current payment system consists of customers, merchants and payment gateways such that a merchant receives a customer's payment information and forwards it to a payment gateway to process the payment. This, however, exposes a customer's payment information to risks, because a merchant can save the customer's payment information in either plain or encrypted form and may misuse it later. It is also possible that a merchant's server, through which a customer's payment information is forwarded to a payment gateway, is compromised and the merchant is unaware of it.

In 2006, five noted payment brands, VISA, MasterCard, Amex, JCB, International and Discover Financial Services, formed a council of security standards called the Payment Card Industry Data Security Standard (PCI DSS). PCI DSS is an open global forum that works on developing, managing, spreading awareness and educating merchants and customers about PCI Security Standards. They also work on Payment Application Data Security Standard (PA-DSS) and Pin Transaction Security (PTS) requirements. Online merchants who use these five payment brands for payment in their websites are required to follow the PCI DSS policies. Violation of the policies may result in revoking a merchant's right to sell, but an online merchant knowingly or unknowingly may not follow all the PCI DSS policies. Furthermore, violation of a policy is usually tracked when it is reported by someone or a breach is identified with a merchant.

Taking all this into consideration, we propose a secure online payment system where a customer's payment information is directly sent to a payment gateway and a merchant does not obtain a customer's payment information, not even in encrypted/hashed form.

1.1 Problem Statement

Despite security standards, sending a customer's payment information to a payment gateway through a merchant is one of the key reasons for customer data theft in the e-commerce system today. When a customer's payment information is sent to an online merchant, the merchant has the capability to obtain the customer's payment information like credit card number, credit card issuer, expiration date, last four digits of a credit card, and card holder's name from a payment gateway. Even if a merchant receives a customer's payment information in an encrypted form, he can save the encrypted information and decrypt it later. The current payment systems allow a merchant to obtain some form of a customer's payment information so that a merchant can claim the validity of a transaction in case of dispute/charge backs. However, a merchant does not necessarily need a customer's payment information to prove the validity of a transaction. Other information related to a purchase can be used to prove the validity of a transaction.

Frauds that occur on the Internet today are mostly from hackers, fraud merchants, spammers, phishers, malware, spyware and data thieves who place attacks on networks and personal computers to corrupt and steal information. Hence, to avoid these risks, it is desirable not to send a customer's payment information to a merchant at all, because it creates the possibilities of security breach and information leaks from a merchant's side.

1.2 Thesis Overview

In this thesis, we propose an online payment system in which a customer's payment information is sent directly to a payment gateway, instead of sending it through a merchant. This approach prevents a customer's payment information from being manipulated and compromised by a merchant. It differs from current approaches where a customer's payment information is first sent to a merchant and the merchant forwards it to a payment gateway. In our proposed payment system, we use a commitment scheme called Pedersen commitment to ensure that only the correct merchant receives payment from the payment gateway. Our proposed payment system also verifies the identity of a merchant before initiating the payment process. The identity check is performed by a trusted third party called the identity provider, which prevents unauthorized or fraudulent merchants from obtaining a customer's payment. Hence, in this thesis, we develop a new approach to an online payment system in which a customer's payment information is directly sent to a payment gateway that allows only the rightful merchant to obtain the payment, and verifies each merchant as genuine before proceeding with the payment process.

Chapter 2: Related Work

One of the most notable payment systems developed is the Secure Electronic Transaction (SET), developed by SETco and led by Visa and MasterCard in 1996. SET was not just a payment system; it was a standard protocol for securing credit card transactions over the Internet. However, it failed due to various reasons, such as the need to install client-side software, cost and complexity, and client-side certificate distribution logistics.

After the failure of SET, Secure Sockets Layer (SSL) and Transport Layer Security (TLS) were used to secure e-communications and e-commerce. VISA and MasterCard have developed their own protocols for secure electronic payments, namely, 3-D Secure as Verified by Visa and MasterCard SecureCode. In addition to Visa and MasterCard, American Express uses SafeKey, and JCB (Japan Credit Bureau) implements J/Secure. To provide security for payment, these providers tie financial authorization with online authentication, where online authentication means authenticating both client and server, and financial authorization means authorizing the payment information using a secret password tied to the debit or credit card.

2.1 SET

The invention of the Internet, and the growth and development of e-commerce, reflected a greater need for secure payment systems. For this reason, Visa and MasterCard started the development of SET in cooperation with various other companies like GTE, IBM, Microsoft, Netscape, RSA, Safelayer and VeriSign. SET was intended to be the standard

protocol for online payment systems. But due to network effect, cost and complexity, and client-side certificate distribution logistics, it was not successful. Despite the failure, SET introduced the dual signature --- an important innovation in cryptography.

2.2 Verified by Visa & SecureCode

With the failure of SET, e-commerce companies had to rely on Secure Sockets Layer (SSL) for secure information flow over the Internet. SSL creates a uniquely encrypted channel for private communication over public channels, and it checks the server's certificate for authenticity before any information is sent. Transporting sensitive data over SSL helped in achieving security, but with complex technologies in use, Transport Layer Security (TLS) was introduced as its successor. TLS is primarily used over transport protocols like Transmission Control Protocol (TCP) for encapsulating the application-specific protocols such as HTTP, FTP, SMTP, NNTP and XMPP. A prominent use of TLS is for securing World Wide Web traffic carried by HTTP to form HTTPS.

Although SSL and TLS are currently in use and provide secure transportation of information over the Internet, e-commerce and credit card companies required a more reliable payment method or system to secure their customer's financial information. For this reason, Visa developed its own secure payment protocol, called Verified by Visa, and MasterCard developed SecureCode. Verified by Visa and SecureCode are an added security level provided by VISA and MasterCard to prevent fraudulent transactions. They confirm the identity of a cardholder through a password-based method. Both these

methods are currently in use and reliable. However, as with other methods, a customer's payment information is sent to the payment gateway via a merchant.

The only difference between Visa and MasterCard's security implementation is the generation of a Universal Cardholder Authentication Field (UCAF): MasterCard uses the Accountholder Authentication Value (AAV) and Visa uses the Cardholder Authentication Verification Value (CAVV). Both the AAV and CVV are 3-digit security codes used to authenticate the cardholder and thereby prevent fraud. The 3-digit security code is an added layer of protection implemented by credit card companies to verify that the credit card is held by the appropriate cardholder.

In both Verified by Visa and SecureCode, a transaction initiates a redirect from a merchant's website to the website of the card-issuing bank to authorize a transaction. These protocols don't take into consideration what authentication method is being used by the card-issuing bank, although most card-issuing banks use a password-based method. This is done so that the bank can be held responsible for any identified security breaches. The presence of the Personal Assurance Message (PAM) on the password page, chosen by a customer when registering with the bank, is their confirmation that the page is coming from that bank. However, this still leaves some possibility of a man-in-the-middle attack, if the customer's browser cannot verify the SSL Server Certificate for the password page.

For a Visa or MasterCard member bank to use the Verified by Visa or SecureCode services, a bank has to operate compliant software that supports the latest protocol specifications. Once compliant software is installed, the member bank will perform product integration testing with the payment system server before it rolls out the system. The ACS, which is usually on a bank's side, is outsourced to a third party by the bank and this condition is not covered by any of the existing payment protocols, including Visa and MasterCard. Commonly, in a customer's browser, the domain name of the ACS provider is shown rather than the bank's domain, but this feature can again be customized to show the bank's domain. This means that when an online transaction is said to be authorized by a bank, it may actually be that a third party is checking all the details of a customer's financial information. Moreover, when Visa and MasterCard leave authentication of an online transaction to a bank, customers are actually relying on the authentication method and security of their bank when giving out their sensitive financial information, whereas, most of the banks are relying on a third party that runs the ACS for them.

Visa and MasterCard also don't license their merchants for sending a payment request to their servers. They isolate their servers by licensing software providers, called merchant plug-in (MPI) providers, and the merchants have to purchase MPI, which is expensive (setup fee, monthly fee and per-transaction fee).

The newer recommendation to use an inline frame (IFrame, when redirecting to the bank for authorization) instead of a popup has reduced user confusion, at the cost of making it

harder for a customer to verify that the page it is redirecting to is genuine in the first place. Also, as of 2010, most web browsers do not provide a simple way to check the security certificate for the contents of an IFrame.

This is a significant disadvantage for these methods as a customer is able to see a merchant's website redirect to unfamiliar domain names due to some vendors' Merchant plug-in (MPI) implementations and the use of outsourced Access Control Server (ACS) implementations by issuing banks, which may make it easier to perform phishing attacks on cardholders.

In some cases, the Verified by Visa system has been mistaken by users for a phishing scam, and has itself become the target of some phishing scams. In addition to all of this, when most of the banks are incorporating mobile banking into their line of services, most mobile browsers particularly present problems for 3-D Secure, due to the common lack of certain features such as iframes and pop-ups. Even if the merchant has a mobile Web site, unless the issuer is also mobile-aware, the authentication pages may fail to render properly, or even at all. In the end, many analysts have concluded that the Activation During Shopping (ADS) protocols invite more risk than they remove and, furthermore, transfer this increased risk to customers.

2.3 Payment systems and Payment Gateways

The demands in online shopping and security have given rise to various payment gateways and systems. Among them, some very secure payment systems like Skipjack provide less customer payment information (debit/credit card information) to merchants.

Skipjack provides two methods of online payment: merchant initiated and customer initiated. The merchant initiated method is the current approach to payment, where merchants collect payment information from customers and send it to payment gateways. When a merchant uses Skipjack's merchant initiated method, Skipjack encrypts a customer's payment information from the merchant's payment form and sends it to the issuing bank for authorization. In Skipjack's customer initiated method, a merchant creates a secure payment form in Skipjack and embeds it in his website for payment. When a customer wants to make payment, he enters his payment information directly on Skipjack's secure payment form. This method, like our proposed payment approach, provides a customer's payment information directly to a payment gateway. But Skipjack provides its merchants with a few customer payment details, like credit card type and the last 4 digits of a debit/credit card, and allows a merchant to choose if a customer's debit/credit card details should be verified through a CCV number.

CCV is the standard security code present on a debit/credit card, and is used to verify that the person using the debit/credit card has the card with him when using it. The CCV security code also helps a credit card issuer to verify the details of a credit card and the identity of the card owner. It is used by all credit/debit card companies as an added protection against fraud. Therefore, when a merchant does not require his customers to provide their credit card CCV number, he is preventing the card issuer from verifying the

card and the card owner's information. These days most credit card companies reject a transaction without a CCV number and most merchants do not approve payment without a CCV number. Though Skipjack provides a secure approach, it does not provide a completely secure environment for online payment.

2.4 Sequence of Steps in a Payment System

The standard sequence of steps used in the current payment system is as follows:

a) A customer visits a merchant's website and selects the items he wants to buy. He adds these items into his online shopping cart.

b) When ready to purchase, a customer provides his payment information to the merchant. Payment information includes a customer's debit or credit card information.

c) The merchant redirects the customer's payment information to a payment gateway for authorizing the customer's payment.

d) The payment gateway checks the customer's payment information, and if correct, authorizes the payment. The payment gateway then sends a payment capture token to the merchant. A payment capture token is a message indicating the authorization of a payment to a merchant. The merchant needs to provide the payment capture token information when requesting payment from the payment gateway.

e) After receiving the payment capture token from the payment gateway, the merchant sends a message to the customer indicating the authorization of the payment.

f) The merchant sends the customer's purchased item to the customer and requests the payment gateway for payment of the same. A merchant sends the payment capture information, received from the payment gateway, to request payment for a purchase.

g) The payment gateway confirms the payment capture information sent by the merchant. If verified, the payment gateway sends the payment to the merchant.

Chapter 3: A Secure Online Payment System

3.1 Introduction

When a customer wants to make an online purchase from a merchant, he visits the merchant's website, chooses the items he is willing to buy, and places those items in his online shopping cart (within the merchant's website). When he is ready to buy those items, he proceeds to checkout. During checkout, a customer provides his shipping and billing address, and payment information (e.g., debit or credit card information) to the merchant. In the online payment systems used today, like Verified by Visa, SecureCode, J/Secure and SafeKey, a customer provides his payment information to a merchant when making an online purchase. This payment information is sent to the merchant in an encrypted or hashed form so that the merchant cannot obtain it. In order to receive payment for his sale, the merchant forwards the customer's payment information to the payment gateway.

For security reasons, it is not safe for a customer to provide his payment information directly to an online merchant over the World Wide Web (WWW), even in encrypted or hashed form. Providing sensitive financial information to an online merchant, even in an encrypted form, makes it vulnerable to risks of financial exploitation/fiduciary abuse.

When an online merchant sells his items, he is only concerned with receiving his payment from a customer for the items sold. So, if the customer provides his payment information directly to the payment gateway, and the payment gateway sends the payment to the merchant, the merchant does not require the customer's payment

information. In this thesis, we propose a secure online payment system in which a customer need not provide his payment information to the merchant for the merchant to get paid for the online purchase made by the customer.

3.2 An Architecture of a Secure Online Payment System

The architecture of the proposed payment system consists of the following entities:

1. Customers
2. Merchants (M)
3. Identity Provider (IP)
4. Payment Gateway (PGW)

3.2.1 Customers

A customer is a person making an online purchase from a merchant. He visits the merchant's website through the WWW, selects the items he wants to buy, puts them in his online shopping cart, and buys them from the merchant. During this procedure, when making payment for a purchase, a customer interacts with a payment gateway to provide his payment information. Each customer has a unique customer id, a public key and a private key pair; the private key is known only to the customer, and a public key certificate.

3.2.2 Merchants

A merchant, denoted by M, is an online seller who has a website listing the items he sells. He has a unique merchant id, a public and private key pair (the private key is known only to the merchant), and a public key certificate.

Every merchant in our payment system is registered with an Identity Provider (IP), and the IP uses this registration to verify if a merchant is legitimate. This helps provide security to customers from fake merchants.

3.2.3 An Identity Provider (IP)

In our secure payment system, an identity provider is an independent trusted third-party that validates a merchant's identity, computes a commitment for a transaction (i.e., commitment of a transaction's identity attribute), and issues a certified identity token for a transaction. The identity token of a transaction is sent to the merchant.

The identity provider checks the merchant's registration to validate its identity. Once validated, it uses the identity attribute provided by the merchant to compute a commitment. The IP, then, issues an identity token for the transaction, which consists of merchant id, identity attribute for which the commitment is computed, commitment signed by the IP and, a random number used in computing the commitment. This identity token plays an important role during the payment process, so a merchant stores all of its information.

3.2.4 Payment Gateway

A Payment Gateway is an e-commerce application service provider that provides tools to process a payment between a customer, merchant and banks over the World Wide Web (WWW). It helps secure a purchase and a customer's payment information in a transaction. A payment gateway protects payment information by encrypting sensitive information, such as credit/debit card details, to ensure that information is passed securely between a customer and, the payment processor. Besides encrypting the payment information, a payment gateway also helps in authorizing payments and protect against financial frauds. Many online merchants use payment gateways for its security, reliability and immediate authorization of payment.

3.3 Why not send payment information to merchants?

The development of e-commerce demonstrated a greater need for payment system to secure customers' financial data when sending it over the Internet. Since then, various payment systems have been developed. But, with the advancement in cyberspace and technology, protecting customers' financial data is not the only issue. Financial frauds reported over the years due to hackers, spammers, fraud merchants and network breaches have become a major concern. To address these problems, current online payment systems like Verified by Visa and SecureCode rely on cutting-edge technologies for identifying fake transactions and fraud merchants. In doing so, they follow an approach where customers' payment information is first sent to merchants and then the merchants redirect it to payment gateways. This approach in conjunction with other technologies works well as merchants and payment gateways communicate directly with each other

and are aware of customers, purchases and payments. But independently, this approach does not facilitate in protecting customers' financial data from fraud merchants. Instead it makes the entire system susceptible to infringement/ intrusion.

When customers send their payment information (hashed/encrypted) to merchants, they are not aware how that information travels over the Internet and is processed by a merchant. Merchants can save the encrypted/hashed financial information of customers and later decrypt it. It might also be possible that a merchant's server is being compromised and he is completely unaware of it. On the other hand, most banks rely on password-based access and encryption methods to secure customers' data. A bank, however, cannot guarantee that a customer's financial information has not been compromised if that information has travelled through a merchant's server before reaching a payment gateway/bank.

Taking these factors into consideration, we infer that sending customers' financial information to payment gateways via merchants exposes it to additional networks susceptible to information leaks and attacks. Hence, to avoid this vulnerability, we develop a different approach for payment systems in which customers directly send their payment information to payment gateways. This way a merchant gets paid for his sold items without receiving a customer's payment information, not even in encrypted/hashed form.

3.4 Design issues when payment information is not sent to merchant

When customers' payment information is sent to a payment gateway via merchants, certain information of the customers along with their purchase and payment details are retained by merchants. The current approach to a payment system allows merchants to store some information related to customers' purchases including some parts of payment information (like credit/debit card type or issuer, last four digits of a credit/debit card or encrypted credit/debit card details). This is done so that merchants can prove a transaction's validity or existence in case of dispute and charge backs. When merchants do not obtain any payment information from customers because, customers provide their payment information directly to a payment gateway, several challenges as listed below need to be addressed.

1. For sending a payment, the payment gateway should be able to identify the merchant and ensure his authenticity;
2. validity of a purchase should be determined by a payment gateway before authorizing its payment to merchants;
3. when sending a payment, the payment gateway should ensure that only the rightful merchant receives payment;
4. And most importantly, merchants should be able to obtain purchase information when required and prove the legitimacy of its payment in case of dispute/charge back.

3.5 Addressing design issues through Pedersen commitment scheme

As mentioned in the previous section, when customers send their payment information directly to payment gateways instead of sending it through merchants, a payment approach faces several design issues. The first issue is identifying merchants by the payment gateway and ensuring their authenticity. In our secure online payment system, however, customers send the merchants' certificate (obtained from the merchants) to the payment gateways along with their payment information. This helps a payment gateway identify which merchant receives the payment. The core issue here is also to verify the merchants' authenticity for payment gateways. For this, each merchant in our payment system registers with an identity provider (IP) to obtain a unique commitment, called a Pedersen commitment. A Pedersen commitment is unique to each transaction and is computed at the start of a payment process by an IP. But, before an IP computes a Pedersen commitment for a merchant's transaction, it checks the merchant's registration. This procedure helps to ensure the authenticity of a merchant to a payment gateway.

The second issue is confirming the validity of a purchase before authorizing its payment to merchants. The reason we need to validate purchases from merchants is to make sure that both merchants and customers have the same purchase details and no information is different on either side. In the current payment approach, merchants verify the order details of a purchase before forwarding the customers' payment information to payment gateways. But, when customers do not send any information to merchants during or before payment, merchants have no way to confirm that their purchase details match customers' purchase details. Therefore, a payment gateway needs to confirm with the

merchant if the purchase details received from a customer are same as the merchant's. In order to achieve this, payment gateways in our proposed payment system use dual signature because they provide the ability to link information and check their individual correctness. Payment gateways create a dual signature of a transaction id and order details as received from customers and send it along with the transaction's Pedersen commitment to merchants. A Pedersen commitment allows a merchant to track its corresponding transaction id and order details for verifying the dual signature. If the dual signature is verified, a merchant sends a transaction id to a payment gateway as verification. This way, a payment gateway validates the transaction and authorizes its payment to merchants.

The third challenge we face is ensuring only the right merchant receives the payment. To guarantee this, payment gateways use a Pedersen commitment's strong computational ability and binding capacity for generating an encryption key. This encryption key is used to encrypt the payment capture token sent by payment gateways to merchants. Merchants require the exact identity attribute of a transaction used when computing the Pedersen commitment in order to decrypt the payment capture token. Since the identity attribute of a transaction is computed by merchants and provided to IP for computing Pedersen commitment, only the right merchant will be able to decrypt the payment capture token. This ensures receipt of payment only by the rightful merchant.

All three entities, i.e., merchant, customer and payment gateway, participating in a transaction know its corresponding Pedersen commitment. This enables each entity to obtain their purchase and payment details when required, especially in case of a dispute or charge back. Since a Pedersen commitment allows tracking the id of a transaction, its order details, identity of the customer making a purchase and identity of the merchant selling items to a customer, each entity can obtain this information when required and prove the legitimacy of a purchase. This solves our last but most important issue of proving the legitimacy of a payment in case of dispute/charge back by merchants.

Hence, the payment system we propose, which has customers provide their payment information directly to payment gateways, addresses all the aforementioned design issues. Furthermore, network breaches are not uncommon today; therefore, we make sure that our payment system protects customers' payment information (debit/credit card details) from network attackers. We depend on a commitment scheme to achieve all of these features. The commitment scheme we choose to implement has to be quick (as payment processing is real-time), efficient, easy to work, and trustworthy. At the same time, we require the commitment's computation to be strong enough that anyone who receives the purchase's payment capture token (a message specifying authorization of a payment), will not be able to decrypt it. With these factors in consideration, we choose the Pedersen commitment scheme for our proposed payment system along with SHA-1 encryption and dual signature implemented over TLS (Transport Layer Security).

3.6 Pedersen Commitment Scheme

Pedersen Commitment scheme is defined as an unconditionally hiding and computationally binding commitment scheme which is based on the intractability of the discrete logarithm problem. It is a type of cryptographic commitment which allows a user to commit to a value while keeping it hidden and preserving the user's ability to reveal the committed value later. This property of a Pedersen commitment protects our proposed payment system against fraudulent merchants and attackers receiving payments of authorized transactions. Moreover, it helps us overcome the most important design issue of allowing merchants to track a purchase's information and prove its validity in case of disputes and charge backs.

Next, we discuss the security features achieved using a Pedersen commitment in our payment system.

a) Since a Pedersen commitment is unconditionally hiding, it hides the attributes used in its computation such that, even with complex and strong computational powers, an entity not required to know the attributes cannot obtain them. For example, customers provide a Pedersen commitment of a transaction to PGWs so that PGWs can check the transaction's validity from a merchant. In this process, although a purchase's order details are contained in its Pedersen commitment, a PGW cannot obtain it. Since PGWs need not know the order details of a purchase, we are successfully able to hide those details using a Pedersen commitment.

b) A Pedersen commitment is also computationally binding and we use this feature of a Pedersen commitment during the most important phase of a payment system, i.e., payment of a transaction. A Pedersen commitment binds a purchase to a merchant such that only the stated merchant can receive payment using the right Pedersen commitment and identity attribute used to compute the commitment. This also means the decryption code of a payment requires knowledge of both the Pedersen commitment and identity attribute of a transaction by merchants. This assures receipt of customers' payments by rightful merchants and protection against network attackers.

c) A Pedersen commitment, which is not computed by merchants, is computed by a trusted third party called an identity provider, only for registered merchants. To obtain a Pedersen commitment, merchants have to provide their merchant id to an IP for verification. Only after verifying the merchant's identity and confirming his registration, an IP computes his requested Pedersen commitment. This prevents fraud merchants from obtaining a Pedersen commitment, without which a payment cannot be processed in our payment system. Hence, this approach protects customers from being tricked by fraud merchants.

d) A Pedersen commitment is computed using a transaction's identity attribute. An identity attribute is unique to each transaction and contains its transaction id, customer id and order details. When merchants receive a Pedersen commitment for an identity attribute, they send it to customers and customers forward it to PGWs. PGWs,

however, cannot obtain customers' order details from a Pedersen commitment despite verifying the same order details from merchants before authorizing payments. This provides confidentiality to customers' purchases from payment gateways.

3.7 How is Pedersen commitment computed?

In our Secure Online Payment System, Pedersen commitment is computed by a trusted third party called identity provider (IP). We believe that computation of Pedersen commitment by a trusted third party instead of merchants, customers, or payment gateways help us prevent online frauds in e-commerce systems. We here describe how Pedersen commitment is computed in our proposed payment system.

Setup: *IP* takes a security parameter t and implements the setup algorithm for Pedersen commitment. *IP* chooses a finite cyclic group G of large prime order p. It chooses two large prime numbers p and q such that q divides $p-1$. Typically, p is 1024 bits and q is 160 bits. We consider working in the subgroup of order-q inside Z_p. IP accordingly takes g as a generator of G_q, the unique order-q subgroup of Z_p and randomly chooses x from Z_q. IP then computes $h = (g^x \bmod p)$ and makes p, q, g *and* h public as the system's parameters while keeping value of x secret.

Commit: An IP computes a commitment, c, for value A received from a merchant, M. For computing the commitment, an IP chooses a random number, $r \leftarrow Z_q$, and computes commitment $c = (g^A h^r \bmod p)$. After computing c, IP sends c and r to M and PGW receives c from customer (customer receives c from M).

Open: A PGW and merchant agree on a predicate *A0*. The PGW uses *c* and *A0 to* create an encryption key, *σ*. PGW creates a message, *m,* encrypts it using *H [σ]* and sends it to a merchant, M. To decrypt *m*, M needs to open a commitment, *c,* using *A* and *r* such that *A=A0*. A merchant is able to obtain a message, *m,* if, and only if, the predicate *A0* he committed to PGW is the same as *A*, the value used to compute the commitment.

Pedersen commitment scheme is said to be unconditionally hiding because an adversary cannot know the value of *A* from *c*. This is because "the commitment of any two numbers in Z_q has exactly the same distribution" [1]. Pedersen commitment is also said to be computationally binding because with the discrete logarithm assumption an adversarial cannot open the commitment, *c*, with any value other than the committed value *A*. "Suppose an adversary finds *a'* other than *a* and *r'* such that $g^{a'}h^{r'} \equiv g^a h^r \bmod p$, then she can compute $(a'-a)/(r-r') \bmod q$, which is $\log_g(h)$, the discrete logarithm of h with respect to *g*. Therefore, breaking the commitment scheme is equivalent to solving $\log_g(h)$ in discrete logarithm" [1].

3.8 Sequence of events in the Proposed Secure Online Payment System

The step-wise overview of our proposed payment system is as follows. The payment system begins when a customer is ready to purchase and pay for items in his online shopping cart.

1. A *merchant, M,* generates a *transaction id* for a purchase and computes its *identity attribute,* $\boldsymbol{A} = (H(\text{Transaction Id})||H(\text{Order Details})||H(\text{Customer Id}))$. He sends the *identity attribute, A*, along with his *merchant id*, in an encrypted form, to *an identity provider, IP*.
2. The *IP* uses the *merchant's id* to check the merchant's registration. After verifying the registration, *IP* computes *Pedersen commitment, c,* for the *identity attribute, A*, sent by the *merchant*. Then-after, the *identity provider* signs the computed *Pedersen commitment* with its digital signature and creates an *identity token*. The *identity token* is an encrypted message, sent by an IP to a *merchant*, consisting of *merchant id, identity attribute - A*, *Pedersen commitment - c* (signed by IP), and a *random number - r* used in computation of the Pedersen commitment.
3. After receiving the Pedersen commitment from the IP, the merchant creates a message for the customer. The message includes a *transaction id*, *identity attribute – A*, *Pedersen commitment – c, order details* and *payment amount* of the purchase, a *dual signature of the transaction id and order details,* DS_{TO}, (so that customers can verify both information), and the *merchant's certificate*. This message provides all essential information of a purchase to allow the customer make payment for the purchase.

4. The customer verifies the *transaction id* and *order details* of the purchase using the *dual signature, DS_{TO}*, sent by the merchant. Then-after, the customer creates an encrypted message containing his *payment and purchase information*, DS_{OP}, and *certificates of the merchant and customer*. This message is intended for the payment gateway, permitting it to process and authorize the customer's payment.
5. The payment gateway receives the customer's payment information for authorizing and processing the payment. But, before authorizing the payment, payment gateway verifies the customer's purchase and payment information. The verification process consists of – a) confirming the purchase's *order details* and customer's *payment information*; and b) validating the transaction from the merchant. The payment gateway confirms the correctness of *order details* and customer's *payment information* using the *dual signature, DS_{TO}*. To validate the transaction, the payment gateway sends the *Pedersen commitment* of a purchase, received from the customer, and the *dual signature of transaction id and order details, DS_{TO},* to the merchant.
6. The merchant uses the *Pedersen commitment* received from the payment gateway to track details of the transaction. Then, using the *dual signature, DS_{TO},* sent by the payment gateway, the merchant validates the correctness of *transaction id* and its corresponding order details. If correct, in response to the payment gateway's message, the merchant sends the identity attribute, *A,* of the transaction.
7. The payment gateway uses the *identity attribute, A,* received from the merchant to compute an *encryption key, σ*. During the computation of σ, payment gateway names the *identity attribute, A,* received from a merchant in the previous step as *A0*. The payment gateway then computes the *hash of σ, represented as $H[\sigma]$,* and encrypts the

payment capture token using *H[σ]*. This encrypted *payment capture token* is sent to the merchant.

8. A merchant can decrypt the *payment capture token* if, and only if, *the identity attribute, A,* sent by the merchant in step 6 is the same as *the identity attribute, A,* computed in step 1 by the merchant. If the merchant is able to decrypt the *payment capture token* successfully, he sends a transaction complete message to the customer or else he sends a transaction failed message.
9. If the merchant was able to decrypt the *payment capture token* in the previous step, he requests the payment gateway to send him the *payment amount*. Usually, merchants request *payment* from payment gateways only after mailing customers' purchased items.
10. Following the merchant's *payment request*, the payment gateway sends the liable payment amount to the merchant. Next, the payment gateway informs both the customer and merchant that payment of the purchase has been posted into the merchant's account.

3.9 A Detailed Algorithm for Secure Online Payment System

In this section, we present a formal detailed algorithm of our proposed Secure Online Payment System. We also show the computation of the Pedersen commitment and dual signature as they come in the process. The procedure begins when a customer is ready to make a payment for his online purchase.

Step 1: Transaction Token Request

A *transaction token* is an *identity token* of a transaction/purchase, provided by a trusted third party, called an *identity provider (IP)*, to a merchant. A merchant sends an *identity attribute* of a transaction to an *IP* for obtaining a *transaction token*. But, before sending the *identity attribute*, a merchant generates a unique identity of a transaction, called a *transaction id*. He then uses the *transaction id* along with the *customer id* and *order details* of a purchase to create an *identity attribute, A,* as follows:

A = (H(Transaction Id)||H(Order Details)||H(Customer Id));

such that, *Transaction Id* is an identification of the transaction (generated by the merchant); *Order Details* are details of items purchased by the customer; and *Customer Id* is the identification of a customer with merchant.

After computing a transaction's *identity attribute*, a merchant encrypts the identity attribute (*identity attribute*) with his private key and sends it to an identity provider along with his unique *merchant id*. This entire message is encrypted with the identity provider's public key so that only the corresponding private key can decrypt it. The *transaction token* request message sent by the *merchant, M,* to an *IP* is as follows:

$$M \rightarrow IP: \{(A)_{M(Pr)}, Merchant\_id\}_{IP(Pu)}$$

where,

A = identity attribute of a transaction

Merchant_id = Merchant's unique identification with the Identity Provider;

M(Pr) = Merchant's Private Key

IP(Pu) = Identity Provider's Public Key

Including a merchant's id in a *transaction token request* message helps the *IP* identify a merchant. Since a merchant has to be registered with an *IP*, the *Merchant_id* allows the *IP* to check the merchant's registration and obtain his public key for decrypting *A*.

Step 2: Transaction Token Issuance

A *transaction token* is issued by an identity provider to a merchant for his transaction. It is created on the basis of the *transaction token request* message received from the merchant. So when an IP receives a *transaction token request* message, it decrypts the message using its private key to obtain an encrypted *identity attribute* and a *merchant's id*. Using the merchant's id, the identity provider checks the registration of the merchant and obtains his public key. The IP uses this public key to decrypt the *identity attribute, A*.

An important role of an identity provider is to compute a Pedersen commitment. After acquiring *identity attribute, A,* of a transaction, the IP computes its *Pedersen commitment, c*, as follows:

$c = (g^A h^r \bmod p)$

where, g = generator of G_q;

A = identity attribute of a transaction

$h = g^x \bmod p$ where, x = uniformly randomly chosen from Z_q; $h \neq 1$

r = number randomly picked from Z_q;

p and q = two large prime numbers such that q divides (p-1)

Following the computation of the Pedersen commitment, the IP creates a transaction token and sends it to the merchant. The transaction token consists of a *merchant id*, *identity attribute - A*, *Pedersen commitment - c,* signed by the identity provider, and a *random number - r,* used in computing the Pedersen commitment. All of this information is encrypted using the merchant's public key and sent to the merchant as follows:

$$IP \rightarrow M: \{ Merchant\_id, A, (c)_{IP(Pr)}, r \}_{M(Pu)}$$

Where,

Merchant_id = Merchant's identification with Identity Provider;

A = (H(Transaction Id)||H(Order Details)||H(Customer Id));

c = Pedersen Commitment;

IP(Pr) = IP's private key

r = Random number used in the computation of c;

M(Pu) = Merchant's Public Key

Step 3: Payment Initiation

A merchant receives a *transaction token* from an IP and uses his private key to decrypt it. In the *transaction token*, a merchant receives back his *merchant id* and the *identity*

attribute, A, of the transaction along with *c*, a Pedersen commitment of the transaction, and a random value, *r*, used in computing *c*. The merchant id and identity attribute, *A*, are sent back to the merchant to let the merchant verify that information and acknowledge that the Pedersen commitment, *c*, belongs to the mentioned identity attribute.

After receiving the transaction token and its information, the merchant prepares a payment initiation message for the customer. The merchant sends this message so that the customer knows his purchase details and, if acceptable, initiates the payment process. The payment initiation message, encrypted with the customer's public key, includes the *transaction id*, *identity attribute – A*, *Pedersen commitment – c*, *order details* of a purchase, total *payment amount*, *dual signature of transaction id and order details*, and the *merchant's certificate* as follows:

$$M \rightarrow \text{Customer: } \{\text{Transaction\_id, A, c, Order Details, Payment Amount, } DS_{TO}, \text{ Merchant's Certificate}\}_{C(Pu)}$$

where,

Transaction_id = identification of a transaction as generated by the merchant

A = identity attribute i.e. (H(Transaction Id)||H(Order Details)||H(Customer Id));

c = Pedersen Commitment;

Order Details = details of items purchased by the customer;

Payment Amount = total cost of the customer's purchase;

DS_{TO} *= Dual Signature of transaction id and order details;*

Merchant's Certificate = a set of information certifying the authenticity of a merchant;

C(Pu) = Customer's Public Key

The merchant sends Pedersen commitment, *c*, to the customer so that the customer can forward it to the payment gateway along with his payment information. The customer sends the Pedersen commitment of the transaction to the payment gateway because it helps the payment gateway authenticate the transaction before sending payment to the merchant. Also included in *payment initiation message* is *dual signature of transaction id and order details*. The purpose of dual signature is to link *transaction id* with *order details* and also check their individual correctness (of transaction id and order details).

To compute a *transaction id and order detail's dual signature*, $\boldsymbol{DS_{TO}}$,

a) merchant takes hash (SHA-1) of *Transaction Id* to obtain *TIMD (Transaction Id Message Digest)* i.e. *H(Transaction_id)*;

b) merchant takes hash of Order Details to obtain *OIMD (Order Details Message Digest)* i.e. *H[Order Details]*

c) *TIMD* and *OIMD* are concatenated i.e. *H(Transaction_id)|| H[Order Details]*

d) Hash of the result is computed i.e. *H(H(Transaction_id)|| H[Order Details])*

e) Merchant signs the final hash with his private key as follows

$\boldsymbol{DS_{TO} = [H(H(Transaction\_id)||H(Order\_Details))]_{M(Pr)}}$, *where M(Pr) is merchant's private key.*

Step 4: Processing of Payment

When a customer receives *payment initiation message* from a merchant, it means that his items are ready for purchase and are awaiting payment. Before making a payment, a

customer, however, verifies the purchase details of a transaction. For this, a customer first decrypts the *payment initiation message* using his private key and confirms the *order details* and *payment amount* of a purchase. Then-after, he verifies the *dual signature,* DS_{TO}, using *transaction id* and *order details*. To conduct this verification, a customer is provided with the purchase's *transaction id* and *order details* separately in the *payment initiation message*. The procedure for verifying the dual signature is as follows:

a) Compute hash of transaction id to obtain *TIMD i.e. H(Transaction Id)*
b) Compute hash of order details to obtain *OIMD i.e. H(Transaction Id)|| H[Order Details]*
c) Concatenate *TIMD* and *OIMD* i.e. *H(Transaction Id)|| H(Order Details)*
d) Compute hash of the concatenated value *i.e. H(H(Transaction Id)||H(Order Details))*
e) Decrypt the dual signature, DS_{TO}, received from the merchant using merchant's certificate key and obtain *H(H(Transaction_id)||H(Order_Details))*
f) If the values computed in *d)* and *e)* match, the *transaction id* and *order details* sent by the merchant to the customer are verified and correct else, it is incorrect.

Verification of dual signature implies that the *order details* and *transaction id* sent individually in a *payment initiation message* are the ones used to compute the dual signature, DS_{TO}. Only after the verification of dual signature, the customer sends a message to payment gateway for processing the purchase's payment.

The message sent by a customer to a payment gateway for processing a payment contains the customer's payment information. But, payment information alone is not sufficient for

the payment gateway to authorize and send payment to the merchant. So, the payment message created by a customer consists of two parts.

The first part of the message contains *payment information, payment amount, order detail's message digest (ODMD), Pedersen commitment – c, transaction id message digest (TIMD)* and a *dual signature of order details and payment information* (DS_{OP}), all encrypted with *one time symmetric key*, K_s. *ODMD* is the hash of order details required by the payment gateway to verify the dual signature, DS_{OP}. The process of computing the dual signature for DS_{OP} is similar to computing DS_{TO} (Step 3) except that DS_{TO} has *transaction id* and *order details* with the entire hashed result encrypted with *merchant's private key*, whereas DS_{OP} has *order details* and *payment information* with the entire hashed result being encrypted with *payment gateway's private key*. The sending *ODMD* instead of *order details* helps in preserving a customer's privacy to purchases. Similarly, *TIMD* is the *hash of transaction id*; used for computing the dual *signature of transaction id and order details*, DS_{TO}. DS_{TO} is computed and sent by a payment gateway to a merchant for authenticating a transaction.

The second part of the message contains a *digital envelope*, the *customer's certificate and* the *merchant's certificate*. The d*igital envelope* contains a *one time symmetric key*, K_s, encrypted with the *payment gateway's public key* so that only the stated payment gateway can decrypt the customer's *payment information* using K_s. Hence, by using a digital envelope we provide an additional encryption-based security to the customer's payment

information. The certificates of the customer and merchant allow a payment gateway to identify them and send messages to them when required.

The message sent by the customer to payment gateway for processing the payment is as follows:

Customer →PGW: {(Payment_Info, Payment Amount, ODMD, c, TIMD, DS_{OP})$_{Ks}$, Digital Envelope, Customer's Certificate, Merchant's Certificate} $_{PGTW(Pu)}$

where,

Payment Info = Credit/ debit card or account details of a customer for making payment,

Payment Amount = Total cost of the purchase made by the customer,

ODMD = Order Details Message Digest i.e. H [Order Details]

c = Pedersen Commitment,

TIMD = Hash of Transaction Id i.e. H[Transaction Id]

DS_{OP} = Dual Signature of Order Details & Payment Information = (H(H[OD]||H[PI]))$_{C(Pr)}$

Ks = One time symmetric Key,

Digital Envelope = (Ks) $_{PGW(Pu)}$

PGW(Pu) = Payment Gateway's Public Key

Step 5: Transaction Check

When a payment gateway receives a *processing of payment* message from a customer, it uses its private key to decrypt those messages. From the first level of decryption, a

payment gateway obtains *certificates of both the customer and merchant* involved in a purchase. Next, PGW decrypts the *digital envelope* using its private key and gets the *one time symmetric key, K_s*. This K_s is used to decrypt the *payment information* of a customer within the *processing of payment* message along with the *payment amount*, *ODMD* and DS_{OP}.

However, before proceeding with payment authorization, a payment gateway verifies the dual signature, DS_{OP}, using *ODMD* and *payment information* of the customer. The process of verifying the dual signature for DS_{OP} is similar to verifying DS_{TO} (as shown in Step 4). When verifying $DS_{TO,}$ a customer requires the merchant's public key to decrypt it whereas, DS_{OP} is encrypted using customer's private key and so the *PGW* requires customer's public key to decrypt it. But, verifying the dual signature, DS_{OP}, is not the only thing a payment gateway does before sending payment to a merchant. Through the Pedersen commitment, it verifies from a merchant the *identity attribute, A,* of the transaction. To implement this, a payment gateway sends a message to the merchant such that, the merchant send back the identity attribute, ***A,*** of the provided Pedersen Commitment, *c*.

PGW → M: {Payment amount, c, DS_{TO} , Customer's Certificate, PGW's Key Exchange Certificate}$_{M(Pu)}$

where,

Payment Amount = Total cost of purchase liable to be paid by a customer

c = Pedersen Commitment,

DS_{TO} = Dual signature of transaction id and order details

Customer's Certificate = a certificate that identifies a customer and his public key

PGW's Key Exchange Certificate = a certificate containing payment gateway's public key information such that the receiving party can send back a message and use PGW's public key for encryption

M(Pu) =Merchant's public key

Step 6: Transaction verification

After receiving and decrypting the *transaction check message*, a merchant uses the *Pedersen commitment* to track its corresponding transaction. The *payment amount* of the transaction and the *customer's certificate* are provided as additional information to help the merchant identify the correct transaction. Following the identification of the transaction, the merchant verifies the dual signature, DS_{TO}, to ensure the correctness of *transaction id* and *order details* sent by a customer to the payment gateway. The verification of DS_{TO} confirms that both merchant and customer are referring to the same transaction with exact order details.

In response to a *transaction check message*, the merchant sends the transaction's identity attribute, *A*, encrypted with the payment gateway's public key, to the payment gateway. This insures that the identity attribute, *A*, is obtained only by the payment gateway. Here, sending a transaction's identity attribute by merchant validates the transaction as genuine. Therefore, we call this message a "transaction verification" message and the notation is as follows:

$$M \rightarrow PGW: \{A\}_{PGW(Pu)}$$

where,

A = H(Transaction_id)||H(Order_Details)||H(Customer_id)

PGW(Pu) = Payment Gateway's Public Key

A payment gateway calls the identity attribute, *A*, sent by the merchant in the transaction verification message *"A0"*.

Step 7: Payment Capture Token

The payment gateway receives the identity attribute, *A*, sent by the merchant in the *transaction verification message* and names it, *A0*. Thereafter, the payment gateway computes the encryption key, **σ**, as follows:

PGW randomly picks $y \leftarrow Z_q$, *where* Z_q *is a cyclic group of order-q; (p, q, g, h) are public attributes used by an IP for computing a Pedersen commitment, c.*

$\sigma = (cg^{-A0})^y$

$= (h^r)^y$ (*Since* $c = g^A h^r$, $cg^{-A0} = g^A h^r g^{-A0} = h^r g^{A-A0} = h^r$)

$= (h^y)^r$

$\sigma = (n)^r$, *where* $n = h^y$

After computing the encryption key, the payment gateway further computes *"hash of σ"* and encrypts the *payment capture token* using it. The *payment capture token* is a message

generated by a payment gateway specifying the authorization of a payment for a transaction. The encrypted payment capture token, ***C***, and is computed as follows:

$\boldsymbol{C = (m)_{H[\sigma]}}$ *; such that m= Payment Capture Token, H[σ] = Hash of σ,* $\sigma = (cg^{-A0})^{y}$

Lastly, the payment gateway encrypts *n*, *C* and *c* using the *merchant's public key* and sends it to the corresponding merchant. The Pedersen commitment, *c*, helps the merchant identify the corresponding random number, *r*, used in computing *c*. A merchant requires the random number, *r*, for decrypting the payment capture token, *C*.

$\boldsymbol{PGW \rightarrow M: \{n, C, c\}_{M(Pu)}}$

where,

$n = h^{y}$ *such that* $(cg^{-A0})^{y}$

c = Pedersen commitment

C = encrypted payment capture token

M(Pu) = Merchant's Public key

Step 8: Receipt of Payment Capture Token

A merchant decrypts the message received from a payment gateway and obtains the encrypted payment capture token, *C*. To decrypt *C*, the merchant requires the random number, *r*, used in the computation of the Pedersen commitment, *c*. Therefore, the merchant uses *c*, sent along with the payment capture token, *C*, to trace its corresponding *r*. Using value *r* and *n*, sent by the payment gateway along with *C*, the merchant computes the decryption key, σ' as follows:

$\sigma' = n^r$

where,

n = value sent by payment gateway along with C, $n = h^y$

r = random number used by IP to compute Pedersen commitment; Merchant receives r from IP in the transaction token message (step 2)

In the Pedersen commitment scheme, the encryption and decryption keys are symmetric; and the proof is as follows:

$\sigma = (cg^{-A0})^y$

$= (g^A h^r g^{-A0})^y$

$= (h^r g^{A-A0})^y$

$= (h^r)^y$

$= (h^y)^r$

$= n^r$

$\sigma = \sigma'$

Thus, $\sigma = \sigma'$, where σ is encryption key and σ' is decryption key.

From the above proof of symmetry we see that computing a decryption key σ' is straight forward for merchants. The value of *n* is sent by *PGWs* to merchants and value of *r* is received from the *IP*.

A merchant receives the payment capture token if, and only if, the identity attribute, *A0*, sent by merchants to PGWs in Step 6 is same as the *identity attribute*, *A*, used to compute

the Pedersen commitment in Step 2. Therefore, the most important factor in computing the encryption key, *σ*, and the decryption key, σ', is the value of *A0* (an identity attribute sent by a merchant in step 6).

Finally, if the merchant is successful in decrypting the *payment capture token*, he sends a transaction complete message to the customer as follows:

M → Customer: {Transaction_id, Payment amount, Merchant's Certificate, successful}$_{C(Pu)}$

Else it sends the following transaction failure message:

M → Customer: {Transaction_id, Payment amount, Merchant's Certificate, failed}$_{C(Pu)}$

Step 9: Payment Request

A payment request message is sent by a merchant to a *PGW* requesting payment for the stated *payment capture token, received from the PGW.* A merchant sends a payment request message to payment gateway only after shipping customers' purchased items.

While requesting a payment, the merchant sends total *payment amount* of a purchase, *capture token information* received from the *PGW* and *merchant's certificate*. All of this is encrypted using the *payment gateway's public key*.

M → PGW: {Payment Amount, Capture Token Info, Merchant's Certificate}$_{PGW(Pu)}$

Step 10: Payment

The payment gateway receives the *payment request* message from the merchant and decrypts it using its private key. A PGW checks the correctness of *capture token information*, *payment amount* and *merchant id* before sending payments to merchants. After verifying all this information, PGW sends the merchant's requested payment and informs the customer about it. To the merchant, it sends the *payment* and *transaction id* encrypted with *merchant's public key*. To the customer it sends the *payment amount*, *merchant id* and *transaction id* all encrypted with *customer's public key*.

(a) PGTW → Merchant: {Payment, Transaction Id}$_{M(Pu)}$
(b) PGTW → Customer: {Payment Amount, Merchant's Certificate, Transaction_id}$_{C(Pu)}$

where,

Payment = Payment of a customer's purchase;

Transaction Id = Identification of a transaction;

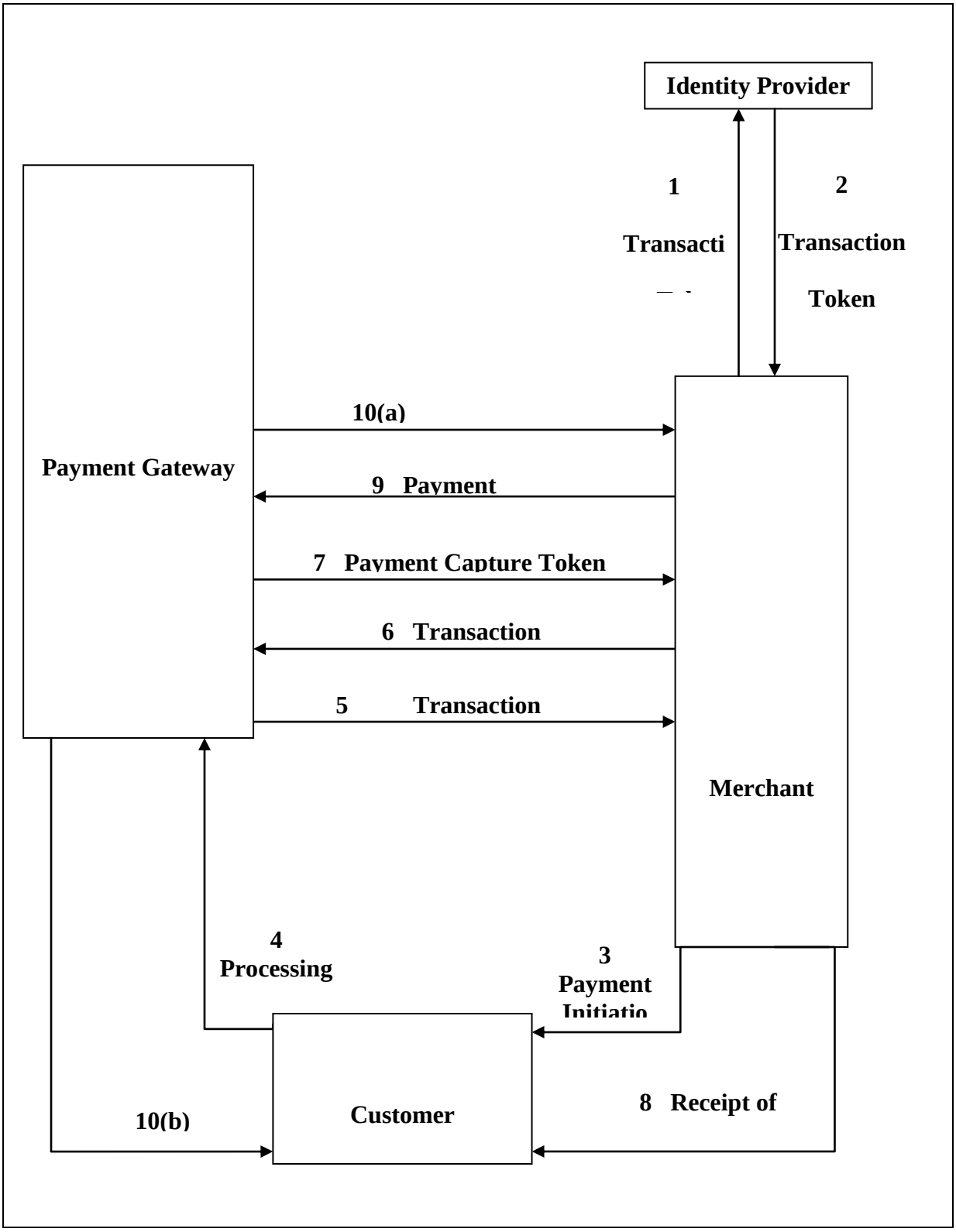
M(Pu) = Merchant's Public Key;

Payment Amount = total amount of a customer's purchase;

Merchant's Certificate = a set of information certifying the authenticity of a merchant;

C(Pu) = Customer's Public Key.

Figure 3.1: Flowchart of "A Secure Online Payment System"



3.10 Security Analysis

The sending of a customer's payment information directly to a payment gateway prevents a merchant from obtaining a customer's sensitive financial information. This protects a customer's payment information from risks of data theft and data infringement on the merchant's side. In this section we show that our approach for online payment is secure and protects both a customer's payment and payment information through security claims in the following ways.

1. ***Only a registered merchant can obtain an identity token from an Identity Provider and initiate the payment process***

   Each merchant in our payment system is required to register with an Identity Provider (IP). An IP checks the identity of a merchant and creates a Pedersen commitment for the merchant's transactions. A Pedersen commitment, which is a unique identity of a transaction, is encrypted within the identity token and is provided to the merchant. A merchant cannot request and receive payment from a customer without an identity token. This is done to secure a customer's payment information and insure that the transaction is being implemented by a registered merchant and not an imposter.

   For instance, let us assume that an attacker acquires the merchant id of a merchant. To initiate a payment process of a transaction, an attacker requires the Pedersen commitment of a transaction. To obtain a Pedersen commitment, an attacker needs to send an identity attribute, *A*, of the transaction to an IP as follows:

$$M \rightarrow IP: \{(A)_{M(Pr)}, Merchant\_id\}_{IP(Pu)}$$

Now, even if an attacker computes an identity attribute, *A*, of a transaction, it has to encrypt *A* with the merchant's private key, which is known only to the merchant. This prevents an attacker from obtaining a transaction's identity token from an IP without which an attacker cannot request payment from a customer.

2. Only the rightful merchant can verify a transaction and obtain its payment

When a payment gateway receives a request for payment from a customer, it does not authorize the customer's payment immediately to a merchant. It sends the Pedersen commitment of the transaction and the dual signature of order details and transaction id to the merchant. A merchant, in response to the payment gateway's message, is required to send the identity attribute, *A*, of a transaction. The sending of a transaction's identity attribute by the merchant to the payment gateway validates the correctness of the transaction. The other reason why a payment gateway verifies a transaction from a merchant is to confirm that a customer's order details and transaction id have not been manipulated and match with the merchant's record.

Let us assume that an attacker captures the *transaction check message* sent by a payment gateway to a merchant for the verification of a transaction. Since the transaction check message is encrypted with the merchant's public key, it can be decrypted only by the merchant's private key. This prevents an attacker from obtaining the information within the transaction check message. However, even if we assume that an attacker knows the merchant's private key, decryption of the transaction check message does not provide an attacker with any information related to the transaction.

The *transaction check message* contains the Pedersen commitment, *c*, and dual signature, DS_{TO}. The Pedersen commitment, *c*, unconditionally hides the identity attribute, *A*, of a transaction, and the dual signature of the transaction id and order details, DS_{TO}, also does not provide any information to the attacker about the transaction. Therefore, the attacker will not be able to send back the identity attribute, *A*, of the transaction for validation purposes. The payment gateway, without receiving a transaction's identity attribute, does not authorize its payment to a merchant. This protects a customer's payment from being obtained by any receiver other than the rightful merchant.

3. A customer's payment information is protected

The main objective of our proposed payment approach is to protect a customer's payment information from being stolen or misused. To achieve this, unlike the current payment approaches, we do not send a customer's payment information to a payment gateway through a merchant. We send a customer's financial information directly to a payment gateway and prevent a merchant from obtaining a customer's financial information even in encrypted form.

When a customer shops online, he is exposed to various risks on the Internet. To avoid these risks, a customer provides his payment information to a payment gateway directly on a payment gateway's server. Payment gateways are more secure and trust-worthy. They also communicate with banks for authorizing and issuing payments. Hence, providing a customer's payment information directly onto a payment gateway's server will protect a customer's payment information from being tampered

with or compromised. A customer sends his payment information to a payment gateway as follows:

Customer $\rightarrow$ *PGW: {(Payment_Info, Payment Amount, ODMD, c, TIMD,* $DS_{OP})_{Ks}$ *,*

Digital Envelope, Customer's Certificate, Merchant's Certificate}

$_{PGTW(Pu)}$

As shown in the message above, a customer provides his payment information and payment amount directly to the payment gateway. The message is encrypted with the payment gateway's public key allowing only the payment gateway to decrypt the message using its private key.

4. Merchants can prove the validity of their transactions in case of dispute/charge backs.

When a customer submits a dispute to a payment gateway against a transaction, the PGW initiates an inquiry with the associated merchant. Current payment approaches allow a merchant to store some details of a customer's debit/credit card to prove the validity of a transaction. Our payment approach, however, restricts a merchant from obtaining any part of a customer's payment information. Instead, a merchant stores Pedersen commitment, *c*, of a transaction, a random number, *r*, used to compute the Pedersen commitment, an identity attribute, *A*, of a transaction and the components used to compute *A*, which includes order details, customer id and transaction id.

In security analysis 1) and 2) we proved that a fraudulent merchant cannot initiate a transaction and obtain a customer's payment. Likewise, a merchant does not obtain any financial information of a customer which makes it very unlikely for a customer's

payment information to be compromised from the merchant's side. Therefore, the only dispute a customer would have in our payment system is regarding a transaction's order details and payment amount.

However, before approving a payment, a payment gateway sends a transaction check message (Step 5) to the merchant to check the correctness of a transaction's order details and payment amount as follows:

$PGW \rightarrow M: (Payment\ amount,\ c,\ DS_{TO}\ ,\ Customer's\ Certificate,\ PGW's\ Key\ Exchange\ Certificate)_{M(Pu)}$

The significance of this message is to let a merchant verify all details of the transaction, provided by a customer, before a payment gateway approves the transaction's payment. This way a payment gateway confirms before authorizing a payment to the merchant that the purchase details of a transaction with both the customer and merchant are exactly the same. In case a dispute arises regarding the transaction's order details and payment amount, a payment gateway then holds the merchant responsible for approving the wrong details of the transaction.

A merchant uses the Pedersen commitment, *c*, to identify the transaction. After identification, a merchant verifies the *order details* and *transaction id* of a transaction using the dual signature, DS_{TO}. During this process of verification and transaction check, a merchant identifies a customer through the customer's certificate. A merchant also confirms the correctness of the payment amount corresponding to the Pedersen commitment, c, and dual signature, DS_{TO}, by comparing it with the payment amount sent by the PGW in the transaction check message. A merchant is supposed to send an

identity attribute, *A*, of the transaction to the PGW only if all details of the transaction match.

In case they vary, a merchant does not send back the identity attribute, *A*, in response to the transaction check message and the payment is not authorized. If a merchant compromises on the purchase information and sends back *A* to the payment gateway, the dispute will prove his violations of the payment system's policies.

Therefore, to avoid disputes and charge backs, our proposed payment system validates all required information before sending a customer's payment to a merchant. However, in case of a dispute, a merchant provides all purchase details of a transaction to a payment gateway for re-verifying them with the information provided by a customer. This also proves that the merchant does not require a customer's payment information to prove the genuineness of a transaction in case of dispute.

Chapter 4: Conclusion and Directions for Future Work

4.1 Conclusion

In this thesis, we discussed current online payment systems, primarily VISA and MasterCard. We discussed their working mechanisms, which are the same except for the generation of UCAF, which secures a customer's debit or credit card information when used on the Internet. In addition to VISA and MasterCard, we also discussed SET and its role in the development of other payment systems and in cryptography.

In the current payment systems, a customer's payment information is sent to a payment gateway via a merchant. This makes the payment system vulnerable to intrusions and information leaks, causing customer data theft, identity theft and fraudulent transactions. To protect a customer's financial information from being compromised, we developed an approach for online payment systems in which a customer's payment information is directly provided to a payment gateway rather than sent through a merchant. This approach, however, introduced some design issues which were addressed by the use of a trusted third party called identity provider, IP, and a commitment scheme called Pedersen commitment. An IP verifies the identity of a merchant before processing the payment. The Pedersen commitment scheme helps to validate a transaction, ensuring receipt of a payment only by the right merchant, and allowing merchants to prove the legitimacy of a transaction in case of disputes/charge backs. Hence, we show that our proposed payment system is secure and protects a customer's payment information and payment against network intruders or attackers.

4.2 Directions for Future Work

In our proposed payment system, an identity provider checks the identity of a merchant and computes a Pedersen commitment for each transaction. The identity of a merchant is checked to ensure that the merchant is not fraudulent. However, validating a merchant for each transaction might be difficult for an IP with a large number of merchants. Therefore, an efficient approach for checking the identity of a merchant and computing the Pedersen commitment at the same time could reduce time and cost of the payment system.

A payment gateway processes payments on the basis of a Pedersen commitment computed by an identity provider. The computation of a Pedersen commitment consists of some public attributes (p, q, g, h) which are later used by payment gateways for computing an encryption key. However, each identity provider can have its own defined set of public attributes (p, q, g, h) for computing a Pedersen commitment. This means that, in case of more than one IP, a payment gateway should know which public values to use while computing an encryption key. To help a payment gateway identify those values, some form of an identity provider's identification should be sent to the payment gateway. A payment gateway should also be able to contact an identity provider in case of disputes or investigations.

Our proposed payment system protects a customer's payment from being compromised by sending it directly to a payment gateway. However, in case of an existing identity theft, our payment system will not be able to determine if a customer is using his own payment information for purchase or somebody else's. On the other hand, a card holder will discover that his debit/credit card information is being used by someone else only after receiving an email from the merchant and the bank regarding the purchase/payment.

So, to identify an existing identity theft, a payment gateway should implement some method to secure a customer's payment that depends on the customer's personal information, like address. When payment gateways obtain payment information from customers, they also have the ability to obtain the IP address of the device used in purchasing. This allows a payment gateway to identity in real time if the purchase is made by the customer himself or someone is using his financial information. Some banks and payment systems currently use this approach and contact a customer for confirmation when they identify a purchase being made from somewhere else.

Bibliography

[1] Jiangtao Li and Ninghui Li. OACerts : Oblivious Attribute Certificates, CERIAS and Department of Computer Science, Purdue University

[2] Mohamed Nabeel, Elisa Bertino. CloudMask : Private Access Control in the Cloud, Purdue University, West Lafayette, Indiana, USA

[3] Ning Shang, Mohamed Nabeel, Federica Paci, Elisa Bertino. A Privacy-Preserving Approach to Policy-Based Content Dissemination, Purdue University, West Lafayette, Indiana, USA

[4] Torben Pryds Pedersen. Non-interactive and information theoretic secure verifiable secret sharing in CRYPTO '91: *Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology, London, UK: Springer-Verlag, 1992, pp 129-140*

[5] http://www.bestpaymentgateways.com/articles3.html

[6] http://www.psbill.com/3-d-secure-verified-by-visa-mastercard-securecode.html

[7] http://www.cs.uky.edu/~singhal/CS687-Au11/ch17.ppt

[8] Shuchen Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing, Dept of ECE, Worchester Polytechnic Institute, Dept of ECE, Illinois Institute of Technology

[9] Mohamed Nabeel, Ning Shang, John Zage, Elisa Bertino. Mask: A System for Privacy-Preserving Policy-Based Access to Published Content, Purdue University, West Lafayette, Indiana, USA

[10] http://groupprops.subwiki.org/wiki/Groups_of_prime_power_order

[11] http://www.informit.com/articles/article.aspx?p=26857&seqNum=3

[12] http://www.gokiosk.net/kiosk/2009/03/card-compromise-statistics-prove-that-pci-dss-compliance-protects-businesses-and-customers.html

[13] http://groupprops.subwiki.org/wiki/Group_of_prime_order

Vita

Date of Birth: 7 February, 1986

Place of Birth: Dhangadi, Nepal

Education

Bachelor of Engineering in Computer Science, Kathmandu University, Nepal, 2007

Shristi Pant