INTE1071

Secure Electronic Commerce

Semester 2, 2020

Assignment 2 – Assignment 2

Name: Duncan Do

Student Number: s3718718

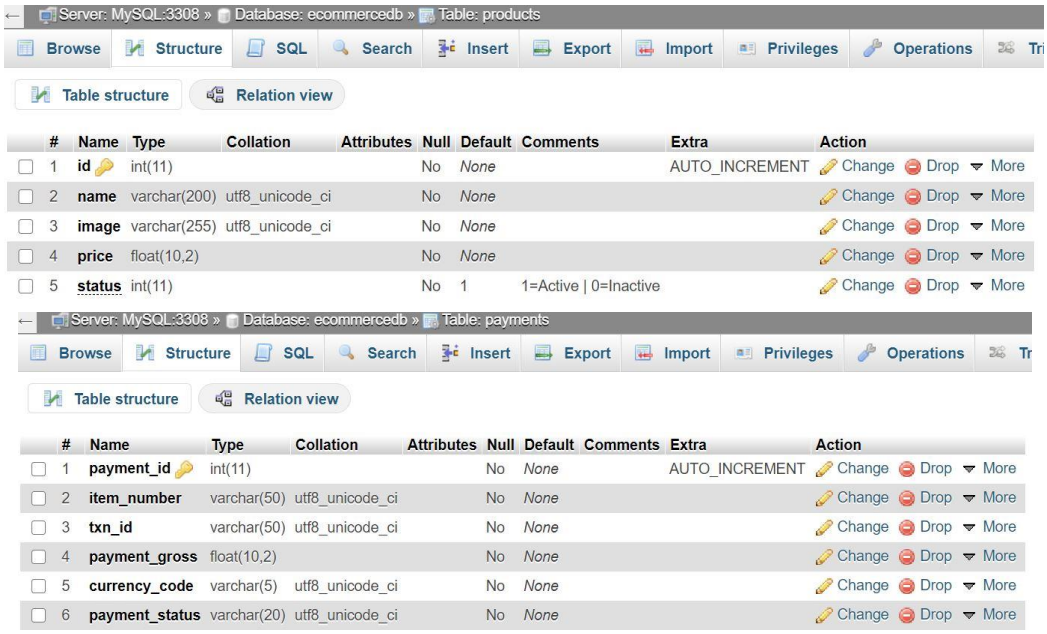
Contents Page

Secure Payment Method Integration

Integrating PayPal payment into an E-Commerce Website.

Step 0: Have a PayPal account (The example implementation uses PayPal's sandbox environment)

For the example, a database housing a table for the E-Commerce product information as well as payment information was created.



Step 1: Create a file to house the configurations of the websites PayPal services (The database configurations will also be housed here)

IN THE FIGURE BELOW

- Line [8-14] define the constant variables for the PayPal configuration
  - **PAYPAL_ID** – Specify the email of the PayPal Business account.
  - **PAYPAL_SANDBOX** – Specify whether you use Sandbox environment (TRUE/FALSE). In your case, it should be TRUE as you are using Sandbox.
  - **PAYPAL_RETURN_URL** – Specify the URL where the buyer will be redirected after payment.
  - **PAYPAL_CANCEL_URL** – Specify the URL where the buyer will be redirected after payment cancellation.
  - **PAYPAL_NOTIFY_URL** – Specify the URL where the transaction data will be sent for verification through PayPal IPN.
  - **PAYPAL_CURRENCY** – Specify the currency code.
- Line [17-20] define the constant variables for the database configuration
  - **DB_HOST** – Specify the database host.
  - **DB_USERNAME** – Specify the database username.
  - **DB_PASSWORD** – Specify the database password.
  - **DB_NAME** – Specify the database name.

```php
// PayPal configuration
define('PAYPAL_ID', 'duncanndo@gmail.com');
define('PAYPAL_SANDBOX', TRUE); //TRUE or FALSE

define('PAYPAL_RETURN_URL', 'http://localhost/assignment2/assignment2_q1/success.php');
define('PAYPAL_CANCEL_URL', 'http://localhost/assignment2/assignment2_q1/cancel.php');
define('PAYPAL_NOTIFY_URL', 'http://localhost/assignment2/assignment2_q1/ipn.php');
define('PAYPAL_CURRENCY', 'AUD');

// Database configuration
define('DB_HOST', 'localhost:3308');
define('DB_USERNAME', 'root');
define('DB_PASSWORD', '');
define('DB_NAME', 'ecommercedb');
```

Step 2: Connect your website to the database itself

IN THE FIGURE BELOW

- The database connection file which links the database to the website

```php
<?php
// Connect with the database
$db = new mysqli(DB_HOST, DB_USERNAME, DB_PASSWORD, DB_NAME);

// Display error if failed to connect
if ($db->connect_errno) {
    printf("Connect failed: %s\n", $db->connect_error);
    exit();
}
?>
```

Step 3: Input necessary code segments into the index/products page to facilitate PayPal payment

IN THE FIGURE BELOW

- Line [35] Is an example where a PayPal variable needs to be identified in the products page to collect the payments through PayPal (Even though it does not need to be displayed on screen)

```php
<form action="cart.php" method="post">
    <input type="hidden" name="business" value="<?php echo PAYPAL_ID; ?>">

    <!-- Specify a Buy Now button. -->
    <input type="hidden" name="cmd" value="_xclick">

    <input type="hidden" name="item_name" value="<?php echo $row['name']; ?>">
    <input type="hidden" name="item_number" value="<?php echo $row['id']; ?>">
```

Step 4: Input the necessary code segments to pass the payment information to PayPal services. Which handle the actual payment themselves

IN THE FIGURE BELOW

- The form which collects the information about the payment transaction
- Line [70] is the submission of the form to PayPal's payment service which is handled offsite

```php
<!-- Paypal payment -->
<!-- PayPal payment form for displaying the buy button -->
PAYPAL<form action="<?php echo PAYPAL_URL; ?>" method="post">
    <!-- Identify your business so that you can collect the payments. -->
    <input type="hidden" name="business" value="<?php echo PAYPAL_ID; ?>">

    <!-- Specify a Buy Now button. -->
    <input type="hidden" name="cmd" value="_xclick">

    <!-- Specify details about the item that buyers will purchase. -->
    <input type="hidden" name="item_name" value="<?php echo $item_name; ?>">
    <input type="hidden" name="item_number" value="<?php echo $item_number; ?>">
    <input type="hidden" name="amount" value="<?php echo $amount; ?>">
    <input type="hidden" name="currency_code" value="<?php echo PAYPAL_CURRENCY; ?>">

    <!-- Specify URLs -->
    <input type="hidden" name="return" value="<?php echo PAYPAL_RETURN_URL; ?>">
    <input type="hidden" name="cancel_return" value="<?php echo PAYPAL_CANCEL_URL; ?>">
    <input type="hidden" name="notify_url" value="<?php echo PAYPAL_NOTIFY_URL; ?>">

    <!-- Display the payment button. -->
    <input type="image" name="submit" border="0" src="https://www.paypalobjects.com/en_US/i/btn/btn_buynow_LG.gif">
</form>
```

Step 5: Create the pages which PayPal redirects to, given the outcome of the payment

IN THE FIGURE BELOW

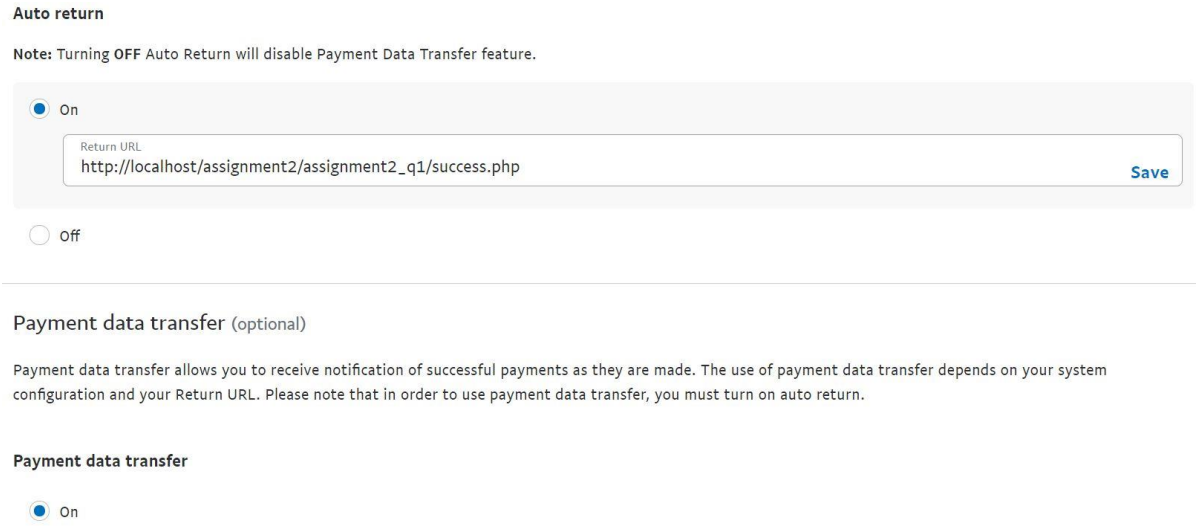
- The page that the website will redirect you to from PayPal after a successful payment
- *Other pages exist for the other behaviours like “Cancel” which simply provide the option to redirect back to the other pages on the website*

```php
    <title>Integrating PayPal Payment System in E-Commerce Website</title>
    <script src='https://www.google.com/recaptcha/api.js' async defer></script>
  </head>
<body>
    <h1>Paypal Integration</h1>
<div class="container">
    <div class="status">
        <?php
        if(!empty($payment_id)){
        ?>
            <h1 class="success">Your Payment has been Successful</h1>

            <h4>Payment Information</h4>
            <p><b>Reference Number:</b> <?php echo $payment_id; ?></p>
            <p><b>Transaction ID:</b> <?php echo $txn_id; ?></p>
            <p><b>Paid Amount:</b> <?php echo $payment_gross; ?></p>
            <p><b>Payment Status:</b> <?php echo $payment_status; ?></p>

            <h4>Product Information</h4>
            <p><b>Name:</b> <?php echo $productRow['name']; ?></p>
            <p><b>Price:</b> <?php echo $productRow['price']; ?></p>
        <?php
        }else{
        ?>
            <h1 class="error">Your Payment has Failed</h1>
        <?php
        }
        ?>
    </div>
    <a href="index.php" class="btn-link">Back to Products</a>
```

Step 6: Modify PayPal settings to interact with your website

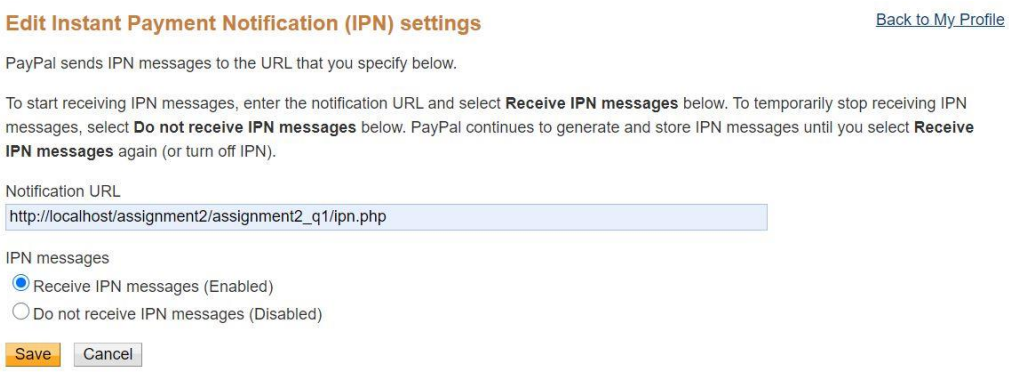
IN THE FIGURE BELOW

- Path to page: Account Settings → Website payments → Website payments → Update
- Set the auto return URL to your payment success page



IN THE FIGURE BELOW

- Path to page: Account Settings → Website payments → Instant Payment Notifications → Update
- Set the notification URL to your IPN page (explained in next step)
- Enable “Receive IPN Message”



Step 7: Create a page for Instant Payment Notification

IN THE FIGURES BELOW

- Figure 1: Reads Payment data from the PayPal system
- Figure 2: Post IPN data back to PayPal to validate
- Figure 3: Act on IPN validation results

Figure 1

```php
 * Read POST data
 * reading posted data directly from $_POST causes serialization
 * issues with array data in POST.
 * Reading raw POST data from input stream instead.
 */
$raw_post_data = file_get_contents('php://input');
$raw_post_array = explode('&', $raw_post_data);
$myPost = array();
foreach ($raw_post_array as $keyval) {
    $keyval = explode ('=', $keyval);
    if (count($keyval) == 2)
        $myPost[$keyval[0]] = urldecode($keyval[1]);
}

// Read the post from PayPal system and add 'cmd'
$req = 'cmd=_notify-validate';
if(function_exists('get_magic_quotes_gpc')) {
    $get_magic_quotes_exists = true;
}
foreach ($myPost as $key => $value) {
    if($get_magic_quotes_exists == true && get_magic_quotes_gpc() == 1) {
        $value = urlencode(stripslashes($value));
    } else {
        $value = urlencode($value);
    }
    $req .= "&$key=$value";
}
```

Figure 2

```php
 * Post IPN data back to PayPal to validate the IPN data is genuine
 * Without this step anyone can fake IPN data
 */
$paypalURL = PAYPAL_URL;
$ch = curl_init($paypalURL);
if ($ch == FALSE) {
    return FALSE;
}
curl_setopt($ch, CURLOPT_HTTP_VERSION, CURL_HTTP_VERSION_1_1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER,1);
curl_setopt($ch, CURLOPT_POSTFIELDS, $req);
curl_setopt($ch, CURLOPT_SSLVERSION, 6);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 1);
curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 2);
curl_setopt($ch, CURLOPT_FORBID_REUSE, 1);

// Set TCP timeout to 30 seconds
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 30);
curl_setopt($ch, CURLOPT_HTTPHEADER, array('Connection: Close', 'User-Agent: Saidur Test'));
$res = curl_exec($ch);
```

Figure 3

```php
 * Inspect IPN validation result and act accordingly
 * Split response headers and payload, a better way for strcmp
 */
$tokens = explode("\r\n\r\n", trim($res));
$res = trim(end($tokens));
if (strcmp($res, "VERIFIED") == 0 || strcasecmp($res, "VERIFIED") == 0) {

    // Retrieve transaction info from PayPal
    $item_number    = $_POST['item_number'];
    $txn_id         = $_POST['txn_id'];
    $payment_gross     = $_POST['mc_gross'];
    $currency_code     = $_POST['mc_currency'];
    $payment_status = $_POST['payment_status'];

    // Check if transaction data exists with the same TXN ID
    $prevPayment = $db->query("SELECT payment_id FROM payments WHERE txn_id = '".$txn_id."'");
    if($prevPayment->num_rows > 0){
        exit();
    }else{
        // Insert transaction data into the database
        $insert = $db->query("INSERT INTO payments(item_number,txn_id,payment_gross,currency_code,payment_status) VALUES('".$item_number."'
    }
```

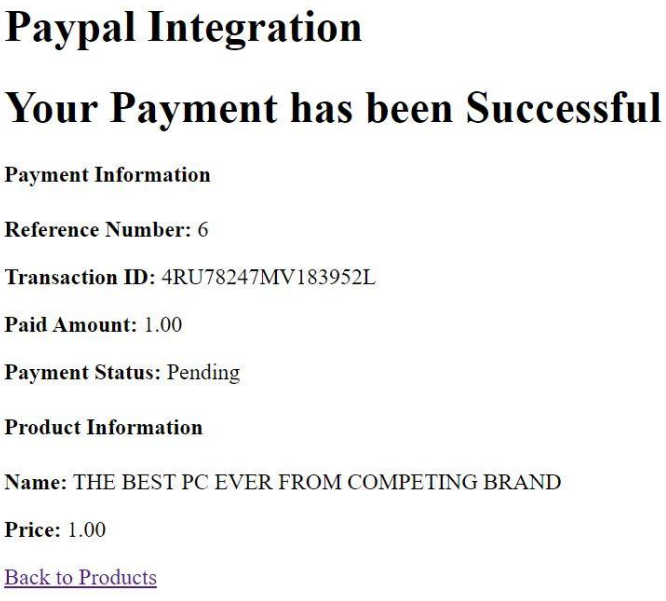
Demo with sample page screenshots

Step 1: Select one of the items on the products page (The button will add the item to the cart and the user will be redirected to the cart page)

Step 2: After examining the cart, the user can click the button and proceed to checkout

Step 3: User can click on the PayPal "Buy Now" button to be redirected to PayPal's payment service

Step 4: Once the user logs into PayPal, enters their card information, and pays, a successful payment will have the user redirected back to the website and to its success page



Secure Payment Method Integration

Integrating Google Pay payment into an E-Commerce Website.

Step 0: Create a JavaScript file with the following…

Step 1: Define Google Pay API version

```
const baseRequest = {
  apiVersion: 2,
  apiVersionMinor: 0
};
```

Step 2: Choose a payment tokenization method

```
const tokenizationSpecification = {
  type: 'PAYMENT_GATEWAY',
  parameters: {
    'gateway': 'example',
    'gatewayMerchantId': 'exampleGatewayMerchantId'
  }
};
```

Step 3: Define supported payment card networks

- Define card networks supported by Google Pay

```
const allowedCardNetworks = ["AMEX", "DISCOVER", "INTERAC", "JCB", "MASTERCARD", "VISA"];
```

- Define card authentication method supported by the website and Google Pay

```
const allowedCardAuthMethods = ["PAN_ONLY", "CRYPTOGRAM_3DS"];
```

Step 4: Describe allowed payment methods

- Combine supported authentication methods and supported card networks to describe the supported card payment methods

```
const baseCardPaymentMethod = {
  type: 'CARD',
  parameters: {
    allowedAuthMethods: allowedCardAuthMethods,
    allowedCardNetworks: allowedCardNetworks
  }
};
```

- Extend the base card payment method object to describe information you expect to be returned to your application

```
const cardPaymentMethod = Object.assign(
  {},
  baseCardPaymentMethod,
  {
    tokenizationSpecification: tokenizationSpecification
  }
);
```

Step 5: Initialize a Payment Client type object in the test environment

```
function getGooglePaymentsClient() {
  if ( paymentsClient === null ) {
    paymentsClient = new google.payments.api.PaymentsClient({
        environment: 'TEST',
      paymentDataCallbacks: {
        onPaymentAuthorized: onPaymentAuthorized
      }
    });
  }
  return paymentsClient;
}
```

Step 6: Determine readiness to pay with the Google Pay API

```
function onGooglePayLoaded() {
  const paymentsClient = getGooglePaymentsClient();
  paymentsClient.isReadyToPay(getGoogleIsReadyToPayRequest())
    .then(function(response) {
      if (response.result) {
        addGooglePayButton();
      }
    })
    .catch(function(err) {
      // show error in developer console for debugging
      console.error(err);
    });
}
```

Step 7: Add a Google Pay payment button

```
function addGooglePayButton() {
  const paymentsClient = getGooglePaymentsClient();
  const button =
      paymentsClient.createButton({onClick: onGooglePaymentButtonClicked});
  document.getElementById('container').appendChild(button);
}
```

Step 8: Create a Payment Data Request type object

- Build an object that describes your website's support for Google Pay

```
const paymentDataRequest = Object.assign({}, baseRequest);
```

- Add the payment methods

```
paymentDataRequest.allowedPaymentMethods = [cardPaymentMethod];
```

- Define the meta data of your E-Commerce products

```
function getGoogleTransactionInfo() {
  return {
      displayItems: [
      {
        label: "Subtotal",
        type: "SUBTOTAL",
        price: "1.00",
      },
     {
        label: "Tax",
        type: "TAX",
        price: "0.00",
      }
    ],
    countryCode: 'AU',
    currencyCode: "AUD",
    totalPriceStatus: "FINAL",
    totalPrice: "1.00",
    totalPriceLabel: "Total"
  };
}
```

Step 9: Register an event handler for user gestures

- Line [243] loads the “load payment data” function to handle user gestures

```
238 function onGooglePaymentButtonClicked() {
239     const paymentDataRequest = getGooglePaymentDataRequest();
240     paymentDataRequest.transactionInfo = getGoogleTransactionInfo();
241
242     const paymentsClient = getGooglePaymentsClient();
243     paymentsClient.loadPaymentData(paymentDataRequest);
244 }
```

Step 10: Insert the Google Pay widget into your html of the payment page (Checkout) and load the JavaScript file as well as the Google Pay external payment method.

```
73          <!-- Google Pay payment -->
74          <div id="container">
75              GOOGLE PAY
76              <div class="row">
77                  <div class="col-sm-6">
78                      <script src="checkout.js"></script>
79                      <script async src="https://pay.google.com/gp/p/js/pay.js" onload="onGooglePayLoaded()"></script>
80                  </div>
81              </div>
82          </div>
```

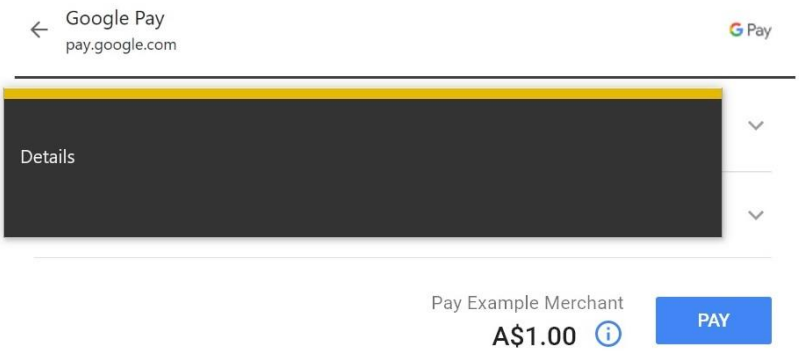
Demo with sample page screenshots

Step 1: Select one of the items on the products page (The button will add the item to the cart and the user will be redirected to the cart page)

Step 2: After examining the cart, the user can click the button and proceed to checkout

Step 3: User can click on the PayPal “Buy with G Pay” button to be redirected to Google Pay’s payment service

Step 4: User will enter their card details and click “Pay” to complete the transaction



Secure Payment Method Integration

Integrating Stripe's multiple card payment services into an E-Commerce Website.

Step 0: Stripe's payment gateway into multiple card payment services requires a bootstrap of stripe-php-<LATEST VERSION>

- Can be obtained from Stripe's GitHub Repository: https://github.com/stripe/stripe-php a

Step 1: Create a file that processes the payment of the user once they click pay

IN THE FIGURE BELOW

- Line [4-6] sets the secret key and publishable key (Provided by Stripe)
- Line [9] Initializes the Stripe API with the secret key

```
4   $stripe = [
5       "secret_key"      => "sk_test_uEvN3RsigPquFvjSMNCIqtpv",
6       "publishable_key" => "pk_test_InwOqy624uXwfN2dgqlCR2gI",
7   ];
8
9   \Stripe\Stripe::setApiKey($stripe['secret_key']);
```

Step 2: Create the Stripe Widget that will take the payment information and send it securely to Stripe

```
84      <!-- Direct Card Payment -->
85      CREDIT CARD DIRECT
86      <form action="paymentProcess.php?pid=1" method="POST">
87          <script
88              src="https://checkout.stripe.com/checkout.js" class="stripe-button"
89              data-key="pk_test_InwOqy624uXwfN2dgqlCR2gI"
90              data-amount="<?php echo $amount*100; ?>"
91              data-name="<?php echo $item_name; ?>"
92              data-description="Some PC"
93              data-image="https://stripe.com/img/documentation/checkout/marketplace.png"
94              data-locale="auto"
95              data-currency="aud">
96          </script>
97      </form>
```

Step 3: In Charge.php we can use the token POST parameter to direct the payment to the specific card service and charge the card

- Stripe-php-<LATEST VERSION> → lib → Charge.php

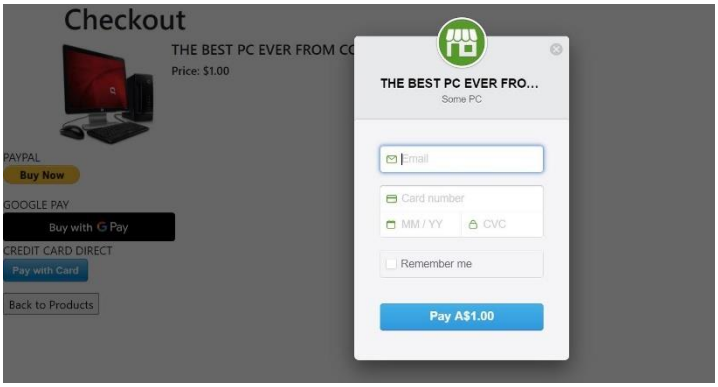
Demo with sample page screenshots

Step 1: Select one of the items on the products page (The button will add the item to the cart and the user will be redirected to the cart page)

Step 2: After examining the cart, the user can click the button and proceed to checkout

Step 3: User can click on the PayPal “Buy with Card” button to be prompted with Stripe’s payment service where they can pay with multiple third-party credit card services (E.g. VISA, AMEX)

Step 4: User will enter their card details and click “Pay” to complete the transaction



Multi-Signature

Scenario 1: 3 Separate users sign a check and get separately verified by the Trusted Third Party

RSA Signature: Signing the message/check (3 separate users)

Step 1: Calculating the public keys

$M = 10000$

USER 1

$p = 443$

$q = 503$

$n = 443 \times 503 = 222829$

$\varphi(n) = (443 - 1) \times (503 - 1) = 442 \times 502 = 221884$

Generate e such that $\gcd(e, \varphi(n)) = 1$

$\gcd(e, 221884) = 1$

$e = 5$

Public Key $= (n, e) = (221884, 5)$

USER 2

$p = 449$

$q = 503$

$n = 449 \times 503 = 225847$

$\varphi(n) = (449 - 1) \times (503 - 1) = 448 \times 502 = 224896$

Generate e such that $\gcd(e, \varphi(n)) = 1$

$\gcd(e, 224896) = 1$

$e = 5$

Public Key $= (n, e) = (225847, 5)$

USER 3

$p = 457$

$q = 503$

$n = 457 \times 503 = 229871$

$\varphi(n) = (457 - 1) \times (503 - 1) = 456 \times 502 = 228912$

Generate e such that $\gcd(e, \varphi(n)) = 1$

$\gcd(e, 228912) = 1$

$e = 5$

Public Key = (n, e) = (229871, 5)

Step 2: Calculating the private keys

USER 1

$d = e^{-1} \bmod(\varphi(n)) = 5^{-1} \bmod(221884) = 44377$

Private Key = (d) = (44377)

USER 2

$d = e^{-1} \bmod(\varphi(n)) = 5^{-1} \bmod(224896) = 179917$

Private Key = (d) = (179917)

USER 3

$d = e^{-1} \bmod(\varphi(n)) = 5^{-1} \bmod(228912) = 91565$

Private Key = (d) = (91565)

Step 3: Signing the message

USER 1

$s = m^d \bmod(n) = 10000^{44377} \bmod(222829) = 206688$

USER 2

$s = m^d \bmod(n) = 10000^{179917} \bmod(225847) = 204173$

USER 3

$s = m^d \bmod(n) = 10000^{91565} \bmod(229871) = 114136$

RSA Signature: Verifying the signature to accept or reject the message/check (3 separate users)

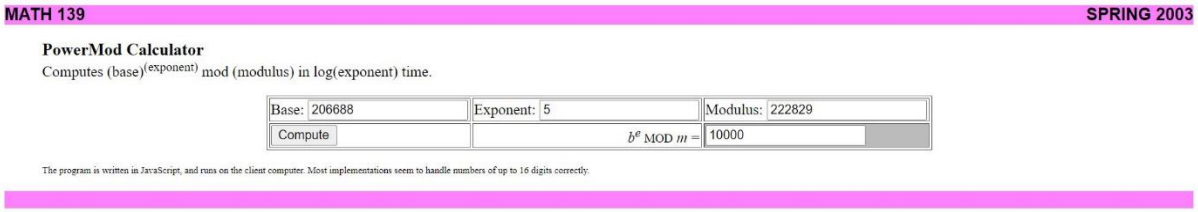
Step 1/1: Verifying the signature

USER 1

$m' = s^e \bmod(n) = 206688^5 \bmod(222829) = 10000$

$m = m'$

VERIFIED

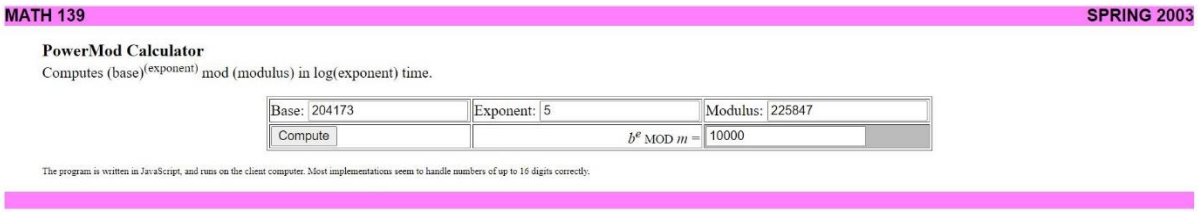


USER 2

$m' = s^e \bmod(n) = 204173^5 \bmod(225847) = 10000$

$m = m'$

VERIFIED



USER 3

$m' = s^e \bmod(n) = 114136^5 \bmod(229871) = 10000$

$m = m'$

VERIFIED

Multi-Signature

Scenario 2: 3 users sign a check using Multi-Signature. Each signing the same check over the prior's signature to create a 3-tiered signed check

RSA Multi-Signature: Signing the message/check (3 users)

Step 1: Calculating the private keys

$M = 10000$

$p = 443$

$q = 503$

$n = 11 \times 17 = 222829$

$\varphi(n) = (443 - 1) \times (503 - 1) = 442 \times 502 = 221884$

Generate t keys such that $t = (\text{number of users/signatures required}) + 1$.

$k_1 \times k_2 \times k_3 \times k_4 = 1 \bmod(\varphi(n))$

Select key values for all keys -1 (Last one is the calculated private key) such that all public keys are prime numbers.

$3 \quad \times 5 \times 7 \times k_4 = 1 \bmod(221884)$

Step 2: Calculate public key

$105 \times k_4 = 1 \bmod(221884)$

$k_4 = 105^{-1} \bmod(221884) = 23245$

Step 3: User 1 signing

$s^1 = m^{k1} \bmod(n) = 10000^3 \bmod(222829) = 46566$

Step 4: User 2 signing

$s^2 = (s^1)^{k2} \bmod(n) = 46566^5 \bmod(222829) = 1346$

Step 5: User 3 signing

$s^3 = (s^2)^{k3} \bmod(n) = 1346^7 \bmod(222829) = 17897$

RSA Signature: Verifying the signature to accept or reject the message/check (3 users)

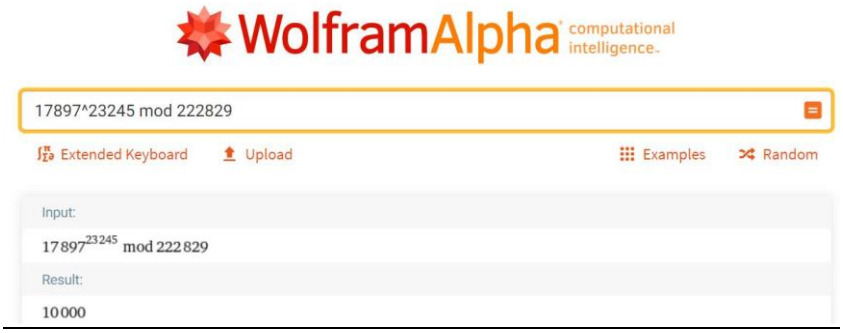
Step 1/1: Verifying the signature

$\text{Signature} = (s, m) = (17897, 10000)$

$m' = s^{k4} \bmod(n) = 17897^{23245} \bmod 222829 = 10000$

$m = m'$

VERIFIED



Multi-Signature

Implement the multi-signature scenario

PATH: SEC_Assign2_Code.zip → assignment2_q2c → multiSignature.py

Language: Python 3

Stock user inputs: [10000, 443, 503, 3, 5, 7]

- Use the above user inputs in sequence to replicate the scenario in the previous question.
- Input 1: Message
- Input [2,3]: p and q Primes
- Input [3,4,5]: Private Key Primes

Designing Reliable E-Commerce Systems

Calculated the number of servers needed per cluster to obtain desired reliabilities

- All Servers: 80% (0.8) Reliability
- Web-server cluster desired reliability: 99.999% (0.99999) Reliability
- Application-server cluster desired reliability: 99.99% (0.9999) Reliability
- Database-server cluster desired reliability: 99.9% (0.999) Reliability

Step 1: Determining the reliability of each individual server (Connected in parallel)

$r_i = 0.8$

Step 2: Calculating number of servers required in parallel to achieve desired reliability rate

We know $R_p = 1 - \Pi^n_{i=1}(1 - r_i)$

WEB-SERVER CLUSTER

To achieve a reliability of 99.999%, probability that the system is working (R_p) should be $R_p = 99.999\% = 0.99999$

Therefore, $0.99999 = 1 - \Pi^n_{i=1}(1 - r_i)$

Or, $0.99999 = 1 - (1 - r_i)^n$

Or, $0.99999 = 1 - (1 - 0.8)^n = 1 - 0.2^n$

Or, $0.2^n = 1 - 0.99999 = 0.00001$

Or, $n \ln 0.2 = \ln 0.00001$

Or, $n = (\ln 0.00001 / \ln 0.2) = \text{Ceiling}(7.1534) = 8$

Therefore, to achieve a reliability of 99.999%, we need **8 servers** connected in parallel.

APPLICATION-SERVER CLUSTER

To achieve a reliability of 99.99%, probability that the system is working (R_p) should be $R_p = 99.99\% = 0.9999$

Therefore, $0.9999 = 1 - \Pi^n_{i=1}(1 - r_i)$

Or, $0.9999 = 1 - (1 - r_i)^n$

Or, $0.9999 = 1 - (1 - 0.8)^n = 1 - 0.2^n$

Or, $0.2^n = 1 - 0.9999 = 0.0001$

Or, $n \ln 0.2 = \ln 0.0001$

Or, $n = (\ln 0.0001 / \ln 0.2) = \text{Ceiling}(5.7227) = 6$

Therefore, to achieve a reliability of 99.99%, we need **6 servers** connected in parallel.

DATABASE-SERVER CLUSTER

To achieve a reliability of 99.9%, probability that the system is working (R_p) should be $R_p = 99.9\% = 0.999$

Therefore, $0.999 = 1 - \Pi^{n}_{i=1}(1 - r_i)$

Or, $0.999 = 1 - (1 - r_i)^n$

Or, $0.999 = 1 - (1 - 0.8)^n = 1 - 0.2^n$

Or, $0.2^n = 1 - 0.999 = 0.001$

Or, $n \ln 0.2 = \ln 0.001$

Or, $n = (\ln 0.001 / \ln 0.2) = \text{Ceiling}(4.2920) = 5$

Therefore, to achieve a reliability of 99.9%, we need **5 servers** connected in parallel.

Designing Reliable E-Commerce Systems

Calculating the overall reliability of the 3-tier E-Commerce System

- Web-server cluster perceived reliability: 99.999% (0.99999) Reliability
- Application-server cluster perceived reliability: 99.99% (0.9999) Reliability
- Database-server cluster perceived reliability: 99.9% (0.999) Reliability

Step 1: Determine how the clusters are connected

Clusters are in series.

Step 2: Apply the appropriate formula to calculating the reliability

Reliability of the system R_S when R_S is in series$= \Pi^{n}_{i=1}(r_i)$

Or, $R_S = 0.99999 \times 0.9999 \times 0.999 = 0.99889$

Therefore, the overall reliability of the E-Commerce System is 99.889%

Secure Identification in E-Commerce Applications

Applying a Zero-Knowledge Proof (ZKP) Protocol to determine the legitimacy of a prover's identity

Step 1: Choose methodology as the Zero-Knowledge-Proof (ZKP) Protocol

Method: Schnorr Identification

Step 2: Trusted Third Party ("Bank") provides shared system parameters

Generate primes p and q such that $\gcd(q, p - 1) > 1$

$p = 48731$

$q = 443$

$\alpha = 6$, a generator mod(p)

$\beta = \alpha^{(p-1)/q} \bmod(p) = 6^{110} \bmod(48731) = 11444$

System Parameters $= (p, q, \beta) = (48731, 443, 11444)$

Trusted Third Party ("Bank") sends System Parameters to both the Prover ("Alice") and Verifier ("Bob")

Step 3: Prover ("Alice") choosing a Private Key and computes other variables for the Verifier ("Bob")

$a = 357$ (Private Key)

$v = \beta^{-a} \bmod(p) = 11444^{-357} \bmod(48731) = (11444^{-1})^{357} \bmod(48731)$

$= (11444^{-1} \bmod(48731))^{357} \bmod(48731) = (29420)^{357} \bmod(48731) = 7355$

$r = 274$ (For computation of the next variable x)

$x = \beta^{r} \bmod(p) = 11444^{274} \bmod(48731) = 37123$

$(x, v) = (37123, 7355)$

Prover ("Alice") sends x to the Verifier ("Bob") to use to confirm her identity with his computations (Also sends v and to use in said computations).

Step 4: Verifier ("Bob") send a random challenge to the Prover ("Alice"), her answer to the challenge will be used in the computations to confirm her identity.

$e = 129$ (The Challenge)

$y = a \times e + r \bmod(q) = 357 \times 129 + 274 \bmod(443) = 225$

$(y) = (255)$

Prover ("Alice") sends y to Verifier ("Bob") to use to in computations to confirm her identity.

Step 5: Verifier ("Bob") computes the variable to check the Prover's ("Alice") identity

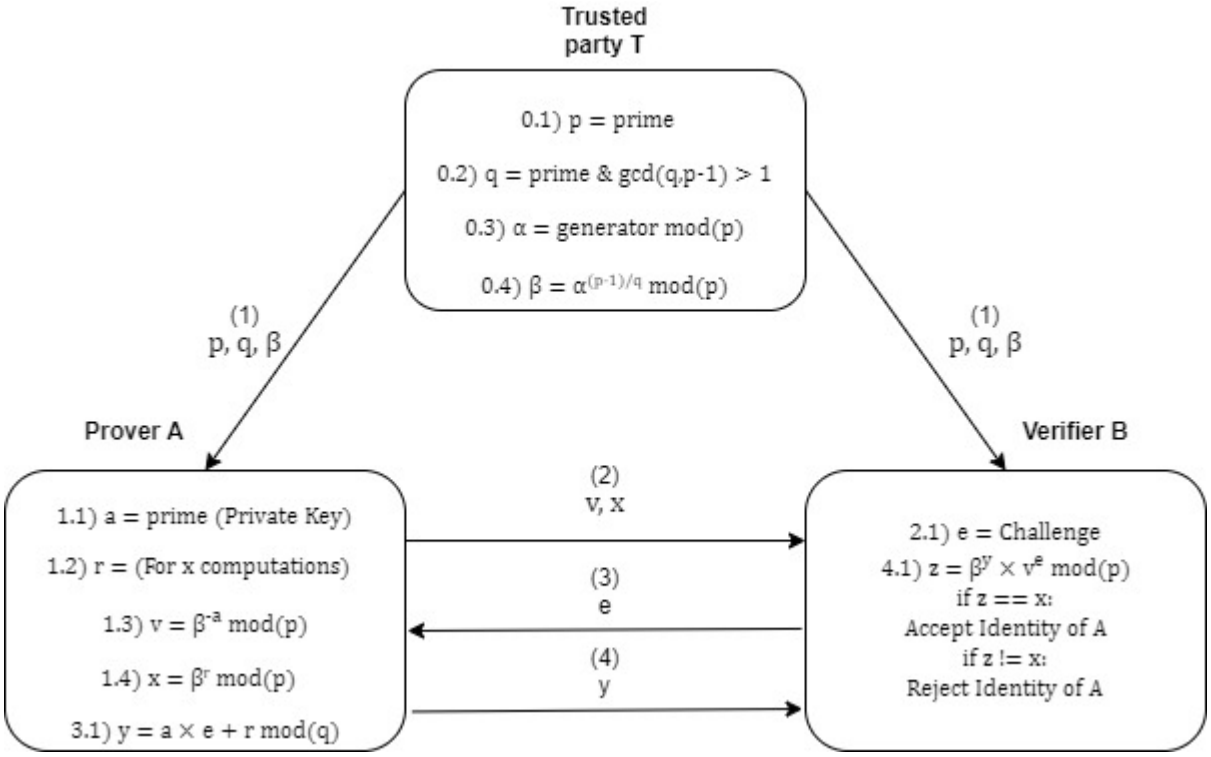
$z = \beta^y \times v^e \bmod (p) = 11444^{255} \times 7355^{129} \bmod(48731) = 37123$

$z = x$

Since Verifier's ("Bob") calculated z variable matches the Prover's ("Alice") x variable, Verifier ("Bob") accepts the identity of the Prover ("Alice").

IDENTITY VERIFED

Sequence Diagram to illustrate Schnorr Identification



Digital Cash

Digital currencies are currencies that only exist in electronic form *(1 Digital Currency, n.d.)* . The balance is stored in a distributed database on the Internet *(2 Digital currency, n.d.)* . Cryptocurrencies are a form of digital currency which uses encryption techniques to make the decentralised network secure. These decentralised networks are based on blockchain technology which is a distributed ledger that enables data storage globally across thousands of servers *(3 Mearian, 2019)* .

Digital currencies vs cryptocurrencies

Cryptocurrency is a kind of digital currency. All cryptocurrencies are digital currencies, however, all digital currencies are not cryptocurrencies *(1 Digital Currency, n.d.)* . For example, the digital yuan, which is supplied and monitored by China's central bank, People's Bank of China *(6 Kharpal, 2020)* .

Digital currencies are centralised, which means that a single authority monitors all the transactions in the network. It can also suspend or cancel transactions if anyone reports anything suspicious. Cryptocurrencies are decentralised, which means each node is independent. Users of digital currencies need to be identified using photo ID or any valid document issued by the government, whereas users of cryptocurrencies can remain anonymous *(7 Tar, 2017)* .

(4 Dienel, n.d.)

In terms of transactions, digital currencies use code from financial institutions to verify the authenticity of the transactions. In case of cryptocurrencies, for example in Bitcoin, miners on the blockchain verify Bitcoin transactions (*9 How bitcoin transactions work | Get Started | Bitcoin.com, n.d.*) . A trusted third party monitors the transactions of digital currencies and the transactions of cryptocurrencies are monitored by ledgers which are available to the public. Cryptocurrency transactions have a very low transaction cost while there is a cost associated with digital currency transactions *(8 Bakar, 2017)* .

Cryptocurrencies are not transparent, i.e. all transactions that take place are accessible to the people, but digital currencies are. Any information about transactions is kept confidential *(10 Digital Currency Vs. Cryptocurrency – What is the Difference? n.d.)* .

A brief introduction to DigiCash and how it worked

DigiCash is a digital currency created by David Chaum, an American cryptographer and computer scientist, in 1989. It was designed to make payments, more importantly micropayments, over the internet, secure. With DigiCash, users could make transactions without revealing their personal information. DigiCash users had to withdraw money from their bank account and convert the money into Cyberbucks (issued by DigiCash) which would then be stored in the hard drive of the user's computer. Payment requests used to be sent and Cyberbucks could be transferred from the payer to the payee after a request was approved. The payer was able to covert Cyberducks into USD which would be deposited to his account. The banks could verify transactions if the users placed the transactions in a "special digital envelope" *(5 Hussey, 2019)* .

Why DigiCash failed

DigiCash transformed the payments industry with its 'Blind Signature' technology. However, the company went bankrupt in 1998 and was sold off to eCash Technologies which was one of the competitors of DigiCash. DigiCash struggled to increase its user base to an adequate size to sustain its operations *(11 Frankenfield, 2018)* . The prime reason of DigiCash's failure was rather strategic. David Chaum did not prove to be a good manager. He could not entrust any employee with a specific task and was always trying to control what his employees were doing. Consequently, research was going at a slow pace. Chaum's controlling nature forced some of his employees with half-done work to resign *(12 How DigiCash Blew Everything, 1999)* .

Some banks had requested DigiCash to allow its operations without the users having to create

accounts. However, this proposal was rejected by this high-tech company. This meant, in order to use the services of DigiCash, users had to have accounts with banks that supported DigiCash. As a result, the company lost a lot of potential customers *(13 A couple of reasons for DigiCash's failure, n.d.)* . A global deficiency of adoption also led the company to bankruptcy. While merchants criticised the system of having a lack of users, users criticised the system of having a lack of merchants which Chaum described as the "chicken-and-egg problem" *(5 Hussey, 2019)* .

Another hindrance to DigiCash's long term success was its failure to make partnership deals with large financial institutions. Citibank had made a lot of negotiations with DigiCash in hope of integration, however, it moved on to other projects *(11 Frankenfield, 2018)* .

Re-designing DigiCash

The failure of DigiCash proved how important business strategies are irrespective of how advanced a product is in terms of technology. Analysing the reasons due to which DigiCash failed, I would take some strategic measures to ensure the success of the company. Firstly, it would be essential to let the employees acknowledge that they are on the same side and they are all trying to achieve the same goal - the company's success and profitability. As some banks had requested DigiCash to allow account less users, I would accept that proposal which would help DigiCash to attract more customers which would eventually lead to more cash flow into the company. Securing partnership deals with major financial institutions would be a high priority as that will result in DigiCash enjoying higher revenue.

For the development team, I would ask the developers to follow the Scrum methodology. In such an agile environment, the scrum master would prioritise the tasks based on them importance during sprint planning and developers work simultaneously. With scrum methodology, changes due to requirement alterations are easy to implement and because of the frequent stand-up meetings, it is always possible to keep track of the progress of each member *(14 Why Does Scrum Work? 6 Reasons Why & Key Benefits of Scrum | Agilest®, n.d.)* . Testing is also started early in the development stage which saves a lot of time and cost. The scrum master will help the team stay focused besides helping the development team resolve any issues or conflicts within the team. This would decrease the friction within the internal employees of DigiCash which is vital for its long-term success.

DigiCash was a centralised network that comes with some limitations. Scalability is one of the major limitations of a centralised network. Even with an increase in hardware and software potential, the performance would not increase significantly which would cause the ratio of benefit to cost to fall. Moreover, in times of heavy traffic, bottlenecks could result *(15 Comparison - Centralized, Decentralized and Distributed Systems - GeeksforGeeks, 2019)* . To overcome these limitations, I would make DigiCash a decentralised network, however, adopt the benefits of a centralised system. According to Forbes, approximately two billion people do not have access to banks and approximately five billion people use mobiles. The banking system can be integrated into the mobile network to make DigiCash accessible to a much higher number of users. This way, the users will also be able to benefit from the security, transparency, and decentralisation network *(7 Tar, 2017)* .

Other features would include making the system to be accessed by its users even when they are offline. This means a user can remain anonymous when engaging in transactions. Moreover, banks would not have to update the database for every transaction and coins can be reused. I would also consider developing an app with QR code that users need to scan. This will make making payments simpler and less time consuming. The user will only be required to connect his bank account with the app. This also makes DigiCash portable *(16 Abrar, 2014)* .

(16 Abrar, 2014)

The simpler operations of DigiCash can be made for users, the more users it will attract. This will help increase DigiCash's profits which can be further invested for more research to

include more features and enable users to have a seamless experience.

Cryptocurrencies like old-style digital currencies

The most popular cryptocurrency, Bitcoin, invented in 2008 by a programmer or a group of programmers who use the name Satoshi Nakamoto, uses the same Bit Gold's (2005) proof-of-work system for the bitcoin mining process *(17 Reiff, 2019)* . Proof-of-work is a consensus mechanism that verifies a transaction in the blockchain network *(18 Tar, 2018)* .

Advantages of cryptocurrencies

The rise of this technology has created a new line of professions which denotes that there will be more jobs created in the tech industry. The blockchain technology helps reduce the bureaucracy within institutions which saves a lot of money and time. For example, companies can track shipments through the blockchain ledger to ensure that their products are not tampered with without having to check them manually. Apart from secure transactions, cryptocurrencies can annihilate costs paid to merchant services. Furthermore, blockchain can be used in the healthcare industry to maintain security and confidentiality of medical information. An Ethereum based blockchain technology was used by MedRec, an MIT-backed enterprise, to transfer family medical records from one generation to another, maintaining full confidentiality. *(23 The Good Side of Crypto: The Potential, Positive, Global Effects of Cryptocurrency - eToro, 2018)*

Why cryptocurrencies are not widely accepted

According to a Forbes article dated November 2018, the most popular cryptocurrency, Bitcoin lost 80% value in 2017. There are many reasons as to why cryptocurrencies are not yet widely accepted despite their hype. Below are some of these reasons.

Cryptocurrencies do not generate any revenue for their owners, which means they do not have any price to earnings ratio *(20 Adkisson, 2018)* . Price to earnings ratio is used to determine whether a company is overvalued or undervalued *(19 Price–earnings ratio, n.d.)* . Moreover, cryptocurrencies are very volatile. Since there is no central authority to inflate or restrict the supply of money, there is no system to halt the global price manipulation, i.e. it is not stable. This has led to many merchants not adopting it since their value can deflate within minutes *(20 Adkisson, 2018)* . If a group of merchants decide to leave the system, its value will decrease significantly which will harm users who have invested a large amount in the system. There is also no buyer protection. For example, if someone is buying any goods using Bitcoins and the seller does not send the products to the buyer; the buyer cannot do anything about it as the transaction is irreversible. Cryptocurrencies can get lost in case there is a hard drive failure, or the data or wallet file is corrupted *(21 Onies, Daniele and Olayinka, n.d.)* . The infrastructure of cryptocurrencies is not user friendly, meaning that it is adopted by only a tech-savvy audience *(22 Vilner, 2018)* .

The Cryptocurrency paradox says "Crypto has value based on its usage to buy things. because of that value, most owners of crypto do not want to use it to buy things; therefore crypto is not widely used to buy things and thus has no value other than related to relatively minimal usage". This implies that cryptocurrencies will not be successful until investors actively use their crypto instead of holding on to them and wait till its value rises *(20 Adkisson, 2018)* .

Conclusion

Cryptocurrencies still have a long way to go before they can be efficient and widely accepted. The concept of blockchain technology, how it works and why it is secure, along with its detrimental effects, should be well understood by the general population before its mass adoption. However, as this growing technology is gradually being used in different sectors of our daily lives such as medicine, transportation, and food, this "exotic" technology will be better understood by people over time and soon enough, people could start using cryptocurrencies *(22 Vilner, 2018)* .