Micropayments: A New Solution

Dept. of CIS - Senior Design 2009-2010

Noah Ready-Campbell
anoah@seas.upenn.edu
Nate Conrad
conradj@seas.upenn.edu
Zachary Ives
zives@cis.upenn.edu
Jonathan Smith
jms@cis.upenn.edu

ABSTRACT

Today, almost all web services rely on advertisements for revenue. Though this model has been successful, it can be intrusive to the user experience, and therefore is ill-suited to many services. One solution that has been put forth is micropayments, i.e. small scale payments typically valued at less than one dollar. Such payments could be for items such as digital content and premium web services. Though micropayments offer much theoretical promise, all attempts to-date have failed. However, due to recent social and technological developments, we believe that the situation is different now. Using an approach with capacity to function in both the short term and the long term, we plan to create a viable method for micropayments to finally take root on the Internet. Such a creation would allow entirely new markets to emerge.

The solution we propose to fill this void is BitNickel™, a micropayments system designed specifically to address the failures of past attempts and to provide a platform for future scalability. The overriding design principle of the system is its ease-of-use, both for merchants and users. To that end, it employs a simple, light-weight interface that requires minimal user input and navigation, which we believe is the key to capitalizing on this opportunity. It also uses several different strategies to minimize the costs associated with each transaction, thereby enabling the profitable implementation of the system.

1. INTRODUCTION

Since the first years of the World Wide Web, businesses have tried to make a profit from it. Given the initial lack of trusted payment systems, digital cash became one of the hottest topics in the Internet sector in the mid 1990s. Dozens of papers were published on the subject, and companies were started to solve the perceived difficulties, including Digicash, Trivnet, and CyberCash. Most of these attempts failed or faded away as users became familiar with and trustful of simple online credit card processing. Despite inherent shortcomings, including insecurity and unwieldiness, credit cards were good enough for most transactions, and their incumbent advantage simply proved too great to overcome.

Though most of these attempts were directed at simply replacing the credit card for standard ecommerce purchases, one subgroup was aimed at a slightly different need: micropayments. Micropayments have been variously identified as being less than one cent all the way up to less than five dollars, which is the accepted definition today [4]. Credit card fees are generally too high to make such transactions profitable for the merchant, so a fundamentally different solution was needed. However, again, most of these early solutions were met with failure. Typically they required that users sign up and use a new currency, and user adoption was a major hurdle to growth. Indeed, these initial failures have prompted some to argue that micropayments are simply unnecessary in the digital economy [15].

1.1 A resurgence

However, there are signs that this view is misguided. Cell phone-based payments, most of them being micropayments, have taken off in many overseas markets; in South Korea, for example, they account for over 70% of online payments [2]. Within the U.S. resides Apple's iPhone App Store [13], where users pay small amounts for simple iPhone applications via their iTunes account. Apple recently announced that more than 2 billion apps have been downloaded, with the market worth up to $500 million annually [16]. We believe that the resounding success of such markets indicates that micropayments deserve a second look. Just as YouTube [3] and Facebook [27] only triumphed after a slew of earlier attempts, there is the potential for a modern micropayments solution to finally take hold. The remainder of this paper is organized as follows. In section 2, we discuss work done in the field related to our project, particularly pertaining to a company with a very similar idea. We emphasize their mistakes and how our approach differs. In section 3, we give a high-level overview of our system. Sections 4 and 5 delve into more technical detail about each of the main segments of our project. Following that, section 6 discusses the general results of the project in terms of areas of success and areas for improvement. Section 7 covers possibilities for future work, and finally, section 8 draws a variety of conclusions on the feasibility of our system and infers various possibilities.

1.2 A micropayments solution

Figure 1: BitNickel™ logo

The BitNickel™ solution presented in this paper lever-
ages one of the key insights that can be gleaned from past attempts, namely that social and legal hurdles, rather than technical challenges, are the main barriers to acceptance of a new payment system. More specifically, this paper proposes that customer adoption is a more important factor for success more than merchant or financial institution adoption. This insight is what enabled PayPal[9] to triumph and has allowed Danal Co [2], the largest of the Korean phone payment companies, to flourish, despite the higher fees it charges merchants and the logistical troubles its had with telecom operators. Naturally, in order to capitalize on this realization, a robust technical foundation is necessary, and the remainder of the paper is devoted to its discussion.

1.3 Micropayments applications

At first glance, this idea may not seem to have many uses. The Internet has survived without them thus far, one may think; why do we need them at all? First, consider an internet service that currently generates revenue using ads. Take Pandora [20], an online radio service, for example: Pandora periodically interjects short commercials between songs. With our product, users could opt to pay only a few cents to skip ads for a month. Even more, they could perhaps pay to unlock extra functionality, such as selecting songs explicitly. This sort of application also easily fits with online video streaming, such as on Hulu.com [14], or any major media site that currently injects ads into the product.

On another front, consider Facebook’s gifts: users can purchase special virtual gifts to send to other users. But how many people actually pay a dollar to send a virtual image? Though there is no public release with data on the matter, the rare use of the feature becomes apparent after a few minutes of checking profiles. Part of that may be due to the price, but another part of it is almost certainly the hassle of entering payment information.

Other applications include charging for downloads of both desktop and mobile applications. As illustrated by Figure 2, users already show a strong preference for micropayment-scale transactions in the iPhone app store, with the preponderance being priced between zero and two dollars. Analysts attribute this fact to the integrated purchasing experience, and even name this feature as a central reason for the platform’s success [19].

Finally, potential applications include fees for website registrations, charitable donations, and other such small digital transactions. The list could go on, but this at least demonstrates the broad potential usage of a viable micropayment system.

2. RELATED WORK

2.1 Security and trust

Initially, security and trust issues were viewed as perhaps the largest barrier to the adoption of online payment systems. The works of McKnight et al. [5] and Egger [8] consider the fundamental necessity of trust within a market and specifically examines consumers’ different psychological propensities to trust. They advance models and techniques for extending such trust to the ecommerce setting, thereby removing barriers to trade. Other papers focus on the technical basis for such trust, for example appropriate cryptographic techniques, such as those discussed by Chaum et al. [6]. However, much of this research is somewhat outdated. Consumers have become accustomed to the idea of the Internet as a transaction medium, and the use of it as such is extremely prevalent [4]. It can be expected that this comfort will only increase, making consumers more receptive to a new payment system than they might have been earlier.

2.2 Transaction efficiency

A second major focus of earlier research has been transaction efficiency, which is more directly targeted at micropayments. Clearly, transactions must cost less to process than they are worth to the engaging parties otherwise the merchant will simply lose money on every sale. Yang et al. [26] consider several different peer-to-peer techniques designed for minimizing resource utilization. They acknowledge that such techniques are less robust than some alternatives, but also note that individual errors or cases of fraud are unimportant on the micropayment scale, and that large-scale fraud trends can still be identified and stopped. This is similar to the fraud approach used by credit card companies. Rivest [21] uses a probabilistic model to aggregate micropayments into larger transactions, thereby lowering the average processing cost. These discussions are more applicable to today’s issues, but are still somewhat off the mark. Rivest’s model is necessarily difficult to implement in practice, when there are refunds and processing errors to consider. The costs Yang et al. consider apply most to the sub-cent transactions, which are impractical anyway, as discussed in the next paragraph.

2.3 Psychological transaction costs

The question of psychological transaction costs, a concept referring to the mental burden placed on the user to complete a given transaction, was only considered relatively late in the development of micropayment systems. Lesk [15] notes that many of the early micropayment systems aimed at sub-cent transactions, e.g. for a single page of an article, added additional psychological transaction costs which outweighed the value of the transaction itself. It is simply too much hassle to evaluate whether it is worth reading the next page of an article at every turn. However, for larger, more targeted micropayments, this argument has been shown empirically false. Nascent markets, such as those found in

![Figure 2: Number of iPhone applications by price](image)
South Korea and the App Store, both discussed earlier, prove the willingness of consumers to engage in transactions at this scale. Additional examples in the domain of massively multiplayer online role-playing games (MMORPGs) and other games further demonstrate this point.

2.4 A similar attempt

There is one previously attempted micropayment approach which we consider very promising, and on which we base the solution presented in this paper. The attempt was made by Trivnet [17], an Israeli company which has since moved into the mobile payments space in Europe and Asia. Trivnet’s product was launched in 1998 and was a back-end payments server it licensed to internet service providers (ISPs). After set-up, the server allowed the ISP’s customers to make credit card-less payments to participating websites, with the payments appearing on the customer’s monthly Internet bill.

From a technical perspective, this product is very similar to that outlined in the next section (though it lacks the auxiliary credit card-based service). However, from a business perspective, Trivnet made several inadvisable business decisions. First, it positioned the product as a convenience feature to help service providers draw new Internet service customers rather than sharing profits with the providers, which is largely wishful thinking, given the public’s view of Internet service as a utility. Trivnet’s model contrasts with the model outlined in this paper, in which a per-transaction fee is given to the ISP, with such revenue contributing materially to the ISP’s top line. Second, since the product was licensed, it left the burden of business development on the shoulders of the ISPs. Trivnet simply did not do enough to develop relationships with merchants, which significantly curtailed the value of its offering. Trivnet’s final failing was out of its control: it was too early. The Internet has evolved substantially over the last eleven years, and many of today’s most popular services, such as Wikipedia [25], could never have existed then.

3. BITNICKEL™ OVERVIEW

We propose BitNickel™, a system specifically targeted at micropayments, particularly focused on ease-of-use. The biggest step towards ease-of-use, in our minds, is the ability to complete the entire transaction without changing web pages. Evidence for this convenience as the centerpiece of a good web interface can be found easily by looking at the results when it is absent. According to a recent study, over 60% of shopping carts are abandoned [11], while another study estimated it as between 60% and 70% [7]. Why are so many abandoned? Because navigating through pages and pages of payment, billing, and shipping information is slow and chunky, particularly on older computers and internet connections. Another major problem is that users are often required to register before making a purchase or because the process is too long and confusing [24]. Users want to enter as little information as possible on as few pages as possible in as short a time as possible. Given these conclusions, through an entirely client-side JavaScript implementation, our system allows the user to do exactly that.

Our system is initiated by a button; clicking on the button creates an overlay interface, and the entire transaction takes place there. The overlay features two payment methods: payment by ISP and payment by credit card. Payment by ISP consists of appending the charge on to the user’s monthly Internet bill. Essentially, as long as the customer’s ISP has technical support for our product, the user must only enter the last four digits of the billing person’s social security number for identity verification purposes. After checking the identity with the ISP, the system then transmits the charge to the ISP for handling and confirms the purchase to the product server. Credit card is included to allow users with ISPs that have not added our functionality to use our system as well. The form for such payments is fairly standard, but one key feature is that the credit cards can be saved, not only for that particular webpage, but for all websites that use our overlay. Furthermore, rather than perform the credit charge instantly, payments are buffered. This is done because the average micropayment is so small that hardly any revenue would be generated given the subtraction of the credit card company’s fee. Instead, credit card billing is on a monthly basis.

4. FRONT-END IMPLEMENTATION

The overlay, implemented entirely in JavaScript, features two tabs. The user can select either the “Credit Card” or “ISP” tab. Merchants, upon creation of their custom buttons, actually have the capability of disabling either of the tabs. While it’s unlikely any would want to disable the ISP tab, the credit card tab may be disabled because credit card companies take a relatively high cut of the payment. Such a decision would unlikely take place unless a large population of ISP’s supported ISP payment.

4.1 Payment by credit

The credit card tab, illustrated in Figure 4, loads previously-saved credit cards, allowing the user to quickly choose one or to enter another. Saved cards can be protected, as there is a password field users can set to be required for use. Credit cards, when saved, are serialized to strings, encrypted, and stored in cookies. This storage allows them to persist between uses and across websites, so that a user must only enter his credit card once on any website to be able to quickly use it again, anytime and anywhere. The encryption of the serialized string prevents attackers from extracting credit card data from the cookies.

The overlay uses the Luhn Algorithm [9] in order to provide a best-effort credit card validation measure. The algo-
The Luhn Algorithm is a simple checksum that was developed by IBM and published in US patent 2,950,048 "Computer for Verifying Numbers" in 1960. It is designed to ensure that no data entry errors or random spoofing of credit card numbers is possible. Credit card companies choose issued numbers in such a way that they pass the checksum.

The actual implemented system uses a two-step credit card verification process before a charge is accepted. First, the card number is checked with the Luhn Algorithm, and then it is checked against an actual credit card processing system. The two-step process is desirable because such credit card validations typically are time and resource intensive, and may even cost a fee. The Luhn Algorithm, in contrast, provides a free, lightweight verification step to help reduce the number of full-scale validations needed.

### 4.2 Payment by ISP

The ISP tab, shown in Figure 5, is far simpler in appearance. The first components to notice are the statement containing the client’s domain name, Internet Protocol (IP) address, and ISP. The other main feature is the input box for the last four digits of the social security number.

In a process shown in Figure 6, the overlay automatically performs a reverse-DNS lookup using the user’s IP address to determine the user’s ISP. The user then must enter the last four digits of the social security number of the person responsible for the internet bill. This required input is to prevent charges from accumulating on public or insecure networks, since knowledge of the person’s social security number indicates either a close relationship or identity theft; the latter case would have far more serious implications than a handful of erroneous micropayments.

Interestingly, the overlay must be implemented very carefully in order to avoid destroying the DOM of the site in which it is embedded. Specifically, when using the JavaScript `document.write()` method to add new scripts, the opening and closing tags "<script>" and "</script>" must be broken up into separate strings for writing, "<scr"="ipt>" and "</sc"r="ipt>". This methodology departs from the official HTML 4 specification, but is required due to the most common browser parsing libraries. As a matter of expediency, the browsers parse the tags too early, in order to achieve a small speed advantage. Clearly, however, this can dramatically revise the intended layout of the page, and result in a variety of issues[22].

### 4.3 Phishing protection

One of the major security vulnerabilities associated with a system such as BitNickel is that users interact with the system via a widget embedded on third-party sites. This opens up a potential avenue for malicious attacks known as phishing, where a malicious entity poses as a trusted participant in the transaction in order to extract information from a victim. Specifically, an attacker could duplicate the appearance of the BitNickel JavaScript button and overlay, put it on an ecommerce website, and use it to extract credit card information or social security digits from unsuspecting users.

This vulnerability is common on the Internet, so there is also a common solution, typically based on a public-key cryptography. In a common implementation using the RSA encryption scheme, a public key is sent from the server to the client. For phishing to work, the entity must supply a public key for which it knows the corresponding private key. To prevent this, trusted companies such as VeriSign[18] use cryptography to sign and approve the true public key from the server. This guarantee cannot be tampered with, so an entity attempting to use phishing to deceive a user would be unable to provide a valid VeriSign certificate. This functionality is demonstrated in the flow chart of Figure 7.
4.4 Input sanitization

Finally, for both security and stability purposes, the overlay intensely sanitizes and validates all inputs. Symbols that could not be part of any valid input block form submission. Symbols that could be valid are escaped with a backslash to prevent them from interfering with the database. This combination of sanitization and validation prevent SQL injection attacks, wherein an attacker uses malformed inputs to interfere with or hijack SQL code.

This sanitization forms an important component of the overall security architecture of the BitNickel™. In a recent security report published by Symantec [28], cross-site scripting (XSS) was listed as the most prevalent security vulnerability of all known attacks. XSS is typically enabled by allowing unsanitized input from users, which is then presented to other users within the context of the original site, and with the same-origin privileges. Consistent and rigorous sanitization of input prevents such attacks, and thus cuts off an important attack vector.

4.5 Front-end design

Code modularity is an important feature of the front-end design. The website, shown in Figure 8, makes heavy use of Cascading Style Sheets (CSS) as well as JavaScript in order to create an appealing user experience—a critical component of success in the competitive online billing market. Each logical component of the CSS and JavaScript is broken into its own server-side file, which is then linked to the appropriate PHP-rendered HyperText markup language (HTML) by the user’s browser. This design minimizes redundant file transfers and also caters to browsers’ caching systems, simultaneously achieving user experience goals and minimizing the strain on the web server. Moreover, as demonstrated in the server-side domain in the following sections, the corresponding reduction of the code base improves the robustness of the code.

4.6 Future interface development

Though the system does not currently support actual charging of credit cards and payment via ISP, the remaining technical steps to do so are trivial. The only remaining technically complicated task is to shore up security. A good first step would be to use Hypertext Transfer Protocol Secure (HTTPS) instead of simply HTTP. HTTP adds Transport Layer Security (TLS) and Secure Socket Layer (SSL) automatically to all communication. This would give a good basis of security which could then be substantially improved using the encryption scheme we designed over the course of this project. As illustrated in Figure 9, the server transmits a public key to the client, which then encrypts everything stored locally.

To make communication safe from replay attacks, there will also be an encrypted signature in all packets containing a timestamp, a MAC or IP address, as well as perhaps some other information as needed to make it verifiably authentic. While this system could have been implemented during this project, no custom encryption scheme can be considered safe unless it has been subjected to an intensive review by experts. Even then, there can often be vulnerabilities. Consequently, for the purposes of this project, it seemed best to simply design the scheme but leave its implementation for future work.

5. BACK-END IMPLEMENTATION

The website, as illustrated in Figure 8, has been created for important tasks such as custom button creation and account management. Such features are integral to the idea of this project. Since such features cannot exist without a website to serve them, we constructed a user-facing informational website for the project. It is dynamically served, using the recursively-named PHP: HyperText Preprocessor (PHP) server-side scripting language. Different logical divisions of
the website are distributed across different PHP files, with each being included in the necessary user-accessible scripts. This model maximizes code re-use, which makes implementation more efficient and extensible, as well as more robust, since re-implementation of similar features increases the likelihood of bugs appearing. Furthermore, this design decision will enable integration with the MySQL relational database, as the PHP/MySQL web services stack is highly optimized for efficient serving.

One current note about the website is that, while it is listed at a custom domain, www.bitnickel.com, it is hosted on a University of Pennsylvania server. This is accomplished using DNS masking; this basically shows a frame of the underlying site to any visitor of the domain. The URL bar of the browser will not reflect the actual page viewed on the website due to this setting. This problem is merely due to the current hosting situation, and would not actually affect a realization of the project, though.

5.1 Database backbone

The crucial functionality of the website relies on a MySQL database. Depicted in Figure 10 is a visualization of the database model for the future of the website. Currently, there is no account management page implemented, so the database model actually used does not support such a feature. The changes needed to allow account management are not particularly complex. To facilitate future work containing this feature, we have created a model that would allow it. The model is located in the Appendix, Figure 14.

An important security note for this database model is that only a hash of the password for each account should be stored. Storing only a hash prevents database intrusion from allowing passwords to be stolen. Even if an attacker copied all of the password data, a cryptographically hash is one-way—that is, the attacker cannot use the hash to determine the original password. Therefore, the attacker would not be able to hijack any accounts.

5.2 Button generation

In addition to a demo of the button on the website, there is an implemented interface, depicted in Figure 11, for generating custom buttons. The current inputs are company name, domain, whether to accept credit cards, whether to accept ISP payments, and the amount to charge. In an actual business application of this project, inputs for an account name and password would also be necessary.

Clicking “create” then displays a custom JavaScript reference to be used to link the button. There are also directions for how to add it to a website, as shown in Figure 12. The script linked provides all of the overlay’s functionality to the HTML object implanted in the merchant’s site; it is basically a reference to a PHP file on our server that accepts a hash of a button ID as a parameter. Including only a hash rather than the button’s ID itself is a simple security precaution to keep information that could be used in a database attack from becoming available. Finally, the copy button is a simple flash trick to copy the text to clipboard for user convenience.

5.3 Future website development

The main component of future work will be an account management section. This is what will allow businesses to use our product. Using the accounts they can already create, they can gain access to features such as the ability to view their pending revenue. The main needs would be to view current balances, check charges on each button, and to create new buttons on the same account.
To implement the account management section, the first step to be taken is to replace the current database model with the one found in the Appendix labeled Figure 14. From there, the steps to supporting the aforementioned functions are fairly simple. Most of what remains are basic SQL functions to aggregate the relevant data. Creating a webpage to display that data would also be fairly trivial.

In addition to the account management section, there are plenty of areas for improvement to make the website more professional and useable. Such features could be added given enough time, but they fall beyond the scope of this technical project.

6. USER EXPERIENCE

6.1 Performance

For the most part, performance in terms of computation time is not a concern in this project. The only extent to which it does matter is that the user interface should not be noticeably slow. At this point, with the majority of the JavaScript to power the client written and tested, we can say that the computation time is currently negligible, and all client operations are essentially instantaneous. A mild delay could arise when encryption is added, but it would certainly not be enough of an issue to make the overlay uncomfortably slow. The amount of delay obviously increases with the complexity of the encryption and the size of the key, but it is also decreases with hardware advances. Really, clients with fast hardware should notice little delay, while those with slow hardware would hardly notice the time compared to the delay of other tasks.

6.2 Client interface

In terms of the interface's user friendliness, the two concerns are time required for use and simplicity. Though these are both fairly hard to determine, we have conducted a small trial with this product consisting of showing it to several users and asking them to rate it. Overall, the remarks were entirely positive about both amount of time required and ease-of-understanding. The persistence of credit cards via the encrypted saving feature allows repeat credit card usage to be quite fast. Similarly, since the user only needs to enter four digits, payment via ISP is also quick. Similarly, the ratings on ease-of-use reflect the success of features such as cookie-based default tab selection and the credit card persistence.

7. FUTURE WORK

7.1 Credit card-based functionality

The credit card processing wing of the project needs further development in several areas. First and foremost, the encryption of sensitive user data must be strengthened and refined. Likely this will involve encrypted cookies, as well as encrypted HTTPS transfers. This security will be based on an RSA public-key encryption scheme where all keys are stored on the server for security. Such a design would completely protect the client-side from any vulnerability, since nothing could be compromised without the private-key stored on the server. Secondly, code must be added to actually communicate with credit card companies for test debits and routine charges. All that would be required would be code to transmit the card data to verify that the data describes a valid card and code to actually charge the final payment. The business aspects such as what fee to pay the
credit card companies and what fee to charge as profit are left outside of the scope of this project.

### 7.2 IP address-based functionality

The execution of ISP-based charges has been left to the future, given the inherent business-facing difficulties in arranging partnerships and obtaining access to test systems. Nevertheless, leveraging the same steps outlined in Figure 6, we have laid the groundwork to make such an advance technically easy. In short, all that remains to be done is to add a component to the ISP’s software that receives a secure transmission of an IP address and four final digits of a social security number, verify that against billing information, and then add a charge to the financial database. The business aspects such as what fee to pay to the ISP and what fee to charge as profit are left outside of the scope of this project.

### 7.3 Website functionality

The main missing feature of the website is the account management section. Much, if not all, of the functionality needed from the database to support such a module already exists in the back-end. The key points of the idea are the ability for a merchant to log in, to view all existing buttons and their balances, to edit these buttons, and to manage their method of extracting money from the buttons. Clearly, the merchant needs to be able to change settings, but the more subtle need is to maintain information about customers for marketing purposes. Though marketing is an often trivialized field, marketers actually base decisions on raw data. The more raw data available, the better targeting can be achieved. Therefore, would be invaluable to merchants using this service. While there are privacy concerns involved with the availability of this data, such issues can be handled by privacy policies and agreements. This component, however, was deemed to be more of a business feature and less of a technical feature necessary for the demonstration of our central idea. It was therefore triaged to be left for future work.

### 7.4 Back-end functionality

To serve the true functionality of credit card charges, a script must be created to automatically crawl through the database periodically to check card balances and usage dates to perform the charges if the balance is over a certain point or if the card has not been used for a certain amount of time. In a functional business, there would also be a need for financial record-keeping of the ISP charges, which is currently not supported. The general idea of this project is that this would occur on a monthly basis, but the timeframe would be easily changed in the event that business decisions determine another choice to be optimal. Also, as mentioned earlier, a small portion of the included database model was optimized out when the decision was made to exclude the account management section from this project. To then implement the account management section in future work, various tables and fields must be added back in to the back-end. The work needed is fairly trivial given the database model, so this should be no concern to any future developers. Functions for extracting the data in useable form to support basic account management functions or even marketing research would also be relatively simple to create.

### 7.5 Security

This will be by far the largest component of future work. As outlined earlier in the paper, a security scheme must be implemented to maintain the safety of the credit card data stored locally in cookies on clients’ computers. This system will most likely be a public-key scheme wherein the private key is stored on and never leaves the server, and there exists a unique key pair for each client. Then the public key could encrypt data for the cookies, and it could not be easily decrypted since that would require extracting the private key from the server, assuming the actual method of encryption cannot be broken. This proposed system would require further changes to the database backbone.

The IP-based payment, conversely, already enjoys substantial security due to the nature of Transmission Control Protocol (TCP) communication. This protocol relies on the establishment of a socket between two known hosts, and, in order for the communication to occur, the addresses of both hosts must be known. This poses a significant architectural obstacle to any would-be attacker. For example, in Figure 13, an attempted attack is demonstrated, with the attacker’s socket communications illustrated in dotted arrows, and the user’s communications in solid arrows.

![Figure 13: TCP Architectural Security](image-url)

Also required is an encryption system to protect the contents of transmissions from the client to the server. The same public key as is used for the credit card storage could be used. What is necessary is a signature for the contents of each message that would prevent tampering with the values or replay attacks. This signature would most likely include a hash of all of the arguments, the client’s IP-address, and a timestamp at a bare minimum. The timestamp and IP-address would ensure that the message is only valid from that client at that specific time, thereby preventing replay attacks. The hash of the arguments prevents any arguments from being changed, such as the button performing the charge. A system of this sort cannot be easily implemented in a secure manner. A bare minimum to attempt to create a secure system should include an intensive review by a panel of security experts. Every facet of the system must be examined.

The positive note about security in this project is that flaws will not be instantly financially devastating. Each exploitation can only cause a few dollars of damage at most,
since this is a micropayment system. Malicious entities, then, must repeat their attack many times to do much damage. This convenience allows for a secondary model of security, in which transactions are monitored to try to detect patterns that might indicate the system is being exploited. By inspecting the overall flow of transactions, a well-designed algorithm may be able to discover attacks and aid implementers in fixing security flaws before much damage has been done.

8. CONCLUSIONS

The success achieved in the implementation of this project puts forth the indisputable idea that a viable micropayments system is within the grasp of today’s businesses, even those with rather limited resources. Our two-person team managed to create a functional manifestation of this design in what technology firms would consider to be a fairly small amount of time.

While there are a variety of vulnerabilities that threaten this idea, the project has shown that they can be overcome. As long as the field of cryptography advances and continues to show that it can create methods of public key encryption that are safe from man-in-the-middle or general cracking vulnerabilities, this project will be able to function in a manner that does not render insecure the client’s credit card information or social security number. We have shown that JavaScript can be used to not only create a comfortable and operable product to manage micropayments, but that with the current state of cryptography, that it can be a safe way to do business.

The remaining challenges which lie outside the scope of this project but must be overcome to realize our vision can be enumerated as follows.

First, this idea may fall under the reach of U.S. Patent US 2002/0156696 A1. This patent makes a claim to micropayments systems operating over a network where involving operation by a service provider, including an ISP [23]. Patent law and interpretation falls outside of the areas of expertise of those working on this project, so such a judgment is left to others in the field. Let it suffice to say that we are not wholly convinced that it is covered, since our system doesn’t involve sole operation by an ISP. Furthermore, even if the system is patented, its creation and operation could still be profitable, as was the case for Google [1] in its youth.

Second, a business must make the effort to broker deals with ISPs for their participation and merchants for their usage. This step is truly crucial. It’s almost something of a chicken-and-egg problem to actually pioneer this project; it’s hard to get ISPs to add support without a customer base, and it’s hard to build a customer base without the main convenience of this project. Such a problem raises the possibility that only an ISP or a web merchant that could provide enough exposure to potential customers to entice ISPs could succeed.

Third, most, if not all, of the features suggested in the future work section must actually be implemented. As discussed in each of the subcategories, all of the components other than security are relatively simple on their own. The ideas described in the security section, however, are the most crucial, the most complex, and the most costly to realize. We believe all of the required work to be manageable, but it must be done well.

Though these issues exist, the key note is that they are all feasible. There are no unsolved technical issues with the ideas raised by this paper. Furthermore, as time passes and business settings change, it is more than probable that a company with the ability to execute this project finds it fiscally viable and realizes our idea.

9. REFERENCES

APPENDIX

A. FUTURE DATABASE MODEL

Figure 14 is the database model to be implemented in future work to support an account management page. Though it looks drastically different from the model included earlier, there are only two new tables and one new field in an old table. The added tables are the Account Table and the Account to Button Map. Also added was the Account ID field in the Button Table. Basically, this new model simply allows for a series of buttons to be grouped together under an authentication method. In an actual business implementation, the Account Table would also have various financial fields for the extraction of button balances.

B. LUHN ALGORITHM

Below is a pseudocode listing for the Luhn Algorithm credit card verification checksum. There is some variation in the details of the implementation for varying lengths of credit card numbers, but we implemented an algorithm very similar to the described below.

```python
def Luhn_Algorithm_Check(credit_no):
    sum = 0
    count = 0
    for d in credit_no:
        temp = int(d)
        if count % 2 == 0:
            temp *= 2
            if temp > 9:
                temp -= 9
        total += temp
    return total % 10 == 0
```
Figure 14: Database Model for Account Management