הערה: To do: add more on computational methods esp. MicroMint. Emphasize efficient methods that provide non-repudiation. Add details, analysis for interoperability.

Chapter 12: Micropayments

In this chapter, we discuss payment systems with low operational costs, supporting low value transactions (micropayments). We first explain the typical scenarios, and then identify, discuss and rank the cost factors. We elaborate on the three most significant cost factors: handling and refunding disputes and chargebacks; customer acquiring and support costs; and computational costs (processing, storage and communication). While most research so far focused on the computational costs, these are often almost negligible. The two other cost factors are more critical. In particular, strong security measures, including digital signatures or equivalent means for non-repudiation, are essential to reduce the dispute handling and chargeback costs.
The hardest challenge is to reduce customer acquiring and support costs. First, there is the problem of creating a sufficiently large market, with many customers, merchants and transactions, to justify the initial investment of all parties (considering the low value of transactions). This requires interoperability among competing Payment Service Providers, which is difficult yet technically feasible, as we show. Second, there is a conflict between the need for customer-controlled `wallet` for non-repudiation, and the difficulty and high costs of convincing customers to install local wallet software. The solution may be secure protocols that allow the use of multiple wallets, including wallet software and/or services provided by third parties (e.g. portals).

Open data networks, such as the Internet and the wireless data networks, allow low-cost delivery of content (information) and services to a huge population (market). The production costs of content and services are often small and largely independent of the number of customers. Therefore, producers of content and services, provided to many customers, often want to charge very small amounts – if the payment system allows it (with reasonable overhead). Payment by credit cards, which is the common method for on-line consumer purchasing, involves substantial minimal fee per transaction, e.g. 20 cents, and therefore is not applicable for charging smaller amounts. This provides one definition of the micropayments, as charging amounts smaller (or close to) the minimal credit card transaction fees (of about 20 cents). There are other difficulties in using credit cards for low value transactions, namely substantial delay and user involvement, and the potential for disputes resulting in refunds, chargebacks, and substantial handling costs.

This creates a difficulty for many existing and potential applications and services on the Internet[1], which need a source of income to cover their costs and generate profits, but the amount they can charge (for one use) is too low to justify a credit card transaction. Currently, most of the deployed services and applications are funded only by advertising or by charging substantial amount in advance for multiple purchases (e.g. subscriptions). A direct payment mechanism could be an important alternative or complementary source of funding, especially to facilitate smaller vendors and applications where advertising cannot be used (e.g. due to lack of appropriate display, and in particular when services are consumed by automated agents

[1] We only mention the Internet but the discussion applies to most open networks.

without any advertising potential). This motivates the development and introduction of *micropayment* schemes and systems.

In this chapter, we focus on providing micropayment services with acceptable (low) transaction cost. This is the basic requirement from a micropayment system, namely that it can be used to charge sufficiently low amounts, in particular below credit card minimal fees (of about 20 cents). In many applications, there is also a minimal amount to be charged with the micropayment system. In particular, when considering payments which involve a manual decision element (by a person), it seems that a minimal amount of about one cent may be sufficient. The reason is that one cent is probably close to the cost of the decision process itself, and smaller value items should probably not require specific user decision and action (otherwise, a lower denomination coin would have been introduced). When considering payments by a software agent of the user, e.g. to pay for the actual communication services, there may be room for payments of amounts even significantly smaller than one cent.

In Section 12.2 below, we analyze the different cost factors for online payments, and in the following sections we elaborate on different mechanisms used to reduce each of the significant cost factors. In the rest of this Section we provide an overview of micropayment systems.

12.1 Overview of Micropayment Systems

There are different motivations for developing new payment mechanisms. We focus on Micropayments mechanisms, operated by one or more Payment Service Provider (PSP), and allowing merchants to charge small amounts from customers. There have been many different definitions, goals and proposals for Micropayment mechanisms. Our focus is on the most common interpretation[2], namely many payments of small amounts (micropayments) from customers to merchants, over open data networks such as the Internet, by using one or more Payment Service Providers (PSPs). A PSP maintains a long-term relationship with customers and merchants, receiving payments of aggregated (large) amount from customers, and passing aggregated payments to the merchants, as illustrated in Figure 1. This model assumes that consumer relationships with merchants are sporadic rather than long-term, and a major role of the PSP is to provide facilities for efficient and secure transactions by using its relationships with the parties.

[2] Some of the alternative interpretations for micropayments, which we will not discuss, include low-value offline payments (using a device rather than coins), anonymous payments (digital cash), and systems where the merchant charges a large amount once but allows the customer to use it incrementally over many small purchases (merchant acting also as a micropayment service provider).

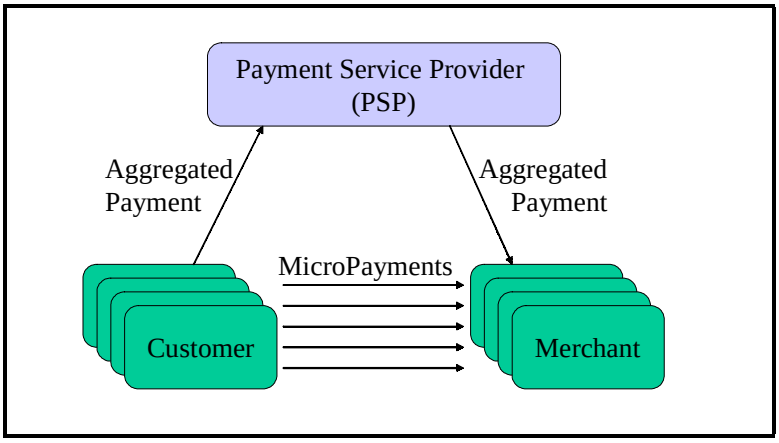

Figure 1: Micropayments via a single PSP

Figure 1 shows the payment relationships between the parties: sporadic micropayments from consumers to merchants, and long-term, usually periodic, payments of aggregated amounts from customers to PSP and from PSP to merchants. This does not describe the flow of messages for a micropayment transaction; we will describe different protocols, with different message flows. Payment protocols include mechanisms for *payment approval* by the customer, where the customer agrees to pay, as well as *payment authorization* by the PSP, where the PSP indicates that there are funds to cover the payment.

Payment approval and payment authorization may be integrated or separated. Separation of the payment approval process from the payment authorization process is appropriate, in particular, in scenarios where the PSP is (or controls) a gateway between the customer and merchant, as illustrated in Figure 2. The customer approves the payment to the PSP, and then the PSP sends an authorized Payment Order (PO) to the merchant. This scenario is applicable whenever the PSP is also providing the communication services to the consumer or is in close alliance with the communication providers. In particular, this scenario is appropriate when the PSP is also the consumer's ISP or a mobile communication provider, or when the ISP or mobile provider are cooperating with the PSP to improve the user interface for payments. In this case, the payment authorization is naturally always online, and involves only the PSP and the merchant. Furthermore, payment authorization can be completely independent from the payment approval process between the consumer and the PSP, and certainly from the login process (if any) between the consumer and her computer or device. In particular, in this scenario we can take advantage of existing security mechanisms between the consumer and his ISP or mobile gateway, to validate that the consumer approved the payment. For example, a mobile gateway usually can identify the handset, e.g. using a shared key, and the handset may identify its user, e.g. using PIN, voice recognition or any other identification technology.

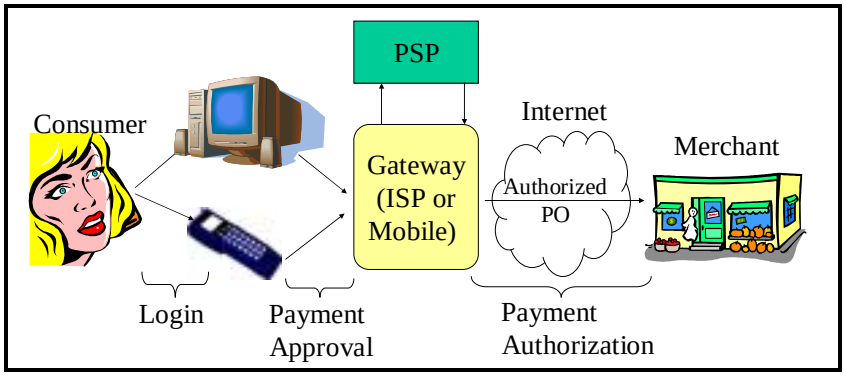

Figure 2: Payments via the PSP

In other scenarios, the PSP is not `on the path` between the consumer and merchant, and therefore either consumer or merchant should contact it to request authorization for payment when required. This is typical e.g. for web browsing, when the PSP is not the ISP (or in alliance with the ISP). In most micropayment systems, the consumer is contacting the PSP to approve the payment and to request the PSP to authorize the Payment Order (PO). For technical reasons, namely allowing the PSP to operate as an efficient server application, the PSP sends the authorized PO as a response to the consumer, who forwards it to the merchant. The merchant will later (offline) deposit the Payment Order, often in a batch process with many other payment orders, to receive the aggregated payment. Figure 3 presents a high level illustration of the online payment process in this scenario (payment invoked by the consumer), as implemented by most currently-deployed micropayment systems, e.g. by Qpass™, iPin™, and TrivNet™. We will later also discuss systems where the merchant is requesting the authorization from the PSP, or where there is no online payment authorization.

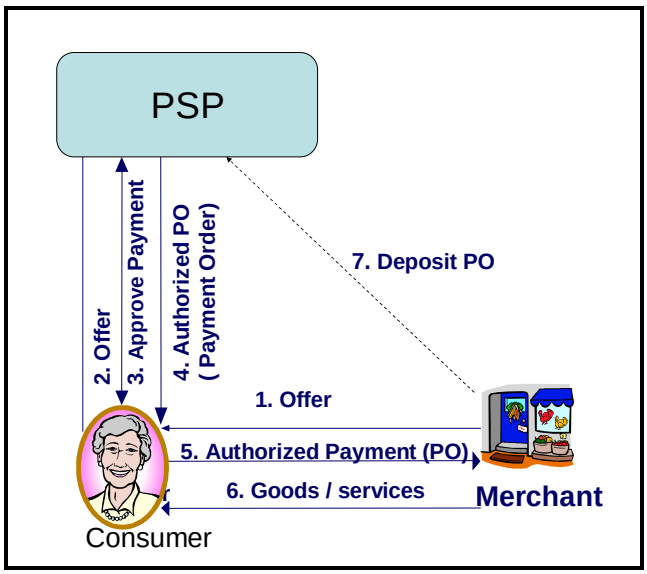

Figure 3: Online Payments Invoked by Consumer

So far, we discussed only a single PSP providing service to both customer and merchant. The single PSP solution is simple and efficient. However, currently there is no dominant single PSP for micropayments. Indeed, there is a substantial number of competing PSPs for micropayments, and we can expect more PSPs to emerge as the demand for micropayments grows and the market matures. In fact, the expected financial returns from micropayment system may not be high enough to justify a sufficient effort by a single PSP, or even a small number of PSPs, to gain market dominance (in contrast to the 2-3 major credit card brands). We therefore expect that there would be multiple PSPs offering micropayment services.

It is unrealistic to expect all customers and all merchants to have accounts with multiple PSPs. Instead, we expect that micropayment systems will need to support interoperability among multiple PSPs, each with its own customers and merchants, and with aggregated payments and long-term relationships between the PSPs, allowing purchases of customers of one PSP from merchants of the other PSP. We illustrate a simple architecture with two PSPs in Figure 4.

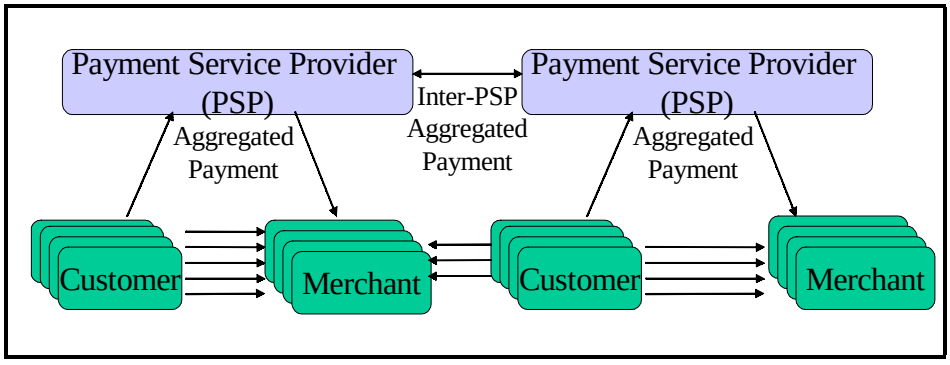

Figure 4: Micropayments via two interoperating PSPs

12.2 Cost Factors for On-line Payments

In order to allow economical charging of small amounts, we need to consider the different cost factors affecting the merchant directly, and indirectly – by affecting the Payment Service Provider. Indeed, from the merchant perspective, it may be enough to use a PSP without the substantial minimal-fee requirement of existing credit cards - but these costs are not pure profit, and the PSP needs to have a viable business case. In order to design a mechanism with substantially reduced costs (cf. to credit cards), we need to consider - and minimize – at least the most significant cost factors.

In the following sections, we consider three major categories of costs:

- **Disputes, Chargebacks and their processing cost:** many payment systems, and even certain laws, allow the customer to dispute charges, or otherwise not to pay, usually under certain circumstances. In some or all of these cases, the PSPs may *reverse* the transaction, requiring the merchant to return the funds (*chargeback*). In particular, payment orders received electronically, without a signed authorization, are often reversible. Indeed, disputes and subsequent chargebacks are substantially more common for Internet transactions than for face-to-face transactions, authorized with the signature of the customer. The costs here include the actual refund amount, as well as substantial processing cost (for the PSPs and the merchant) and possible penalty payments (by the merchant). The processing costs of credit card payment service providers are estimated at about 50$, with penalties payments for merchants with frequent disputes of about 100$ [SRW01]. This is probably the most critical cost factor for micropayments and much of the work on micropayment systems is targeted at reducing the expenses associated with it; see Section 12.3 below.

- **Customer acquiring and support costs:** the costs of encouraging customers to deploy the new service (open an account, install a wallet, etc.), and later assisting customers. These expenses may be substantial, esp. compared to the small value of the transaction. The main mechanism to reduce theses costs is to use simple procedures and user interface. In particular, it is highly desirable to offer easy to use, `click and pay` user interface for micropayments. On the other hand, to minimize installation and

support costs, customer should be able to use standard software tools (e.g. browser) rather than installing customer software (wallet) on the consumer's machine. Finally, PSPs should be interoperable, namely customer of one PSP should be able to buy from merchant of another PSP, as in Figure 4, so that multiple PSPs share the customer acquiring and support costs. See more details in Section 12.4 below.

- **Equipment, processing and communication costs:** these are the costs of the necessary hardware, software and communication for processing the payments, by customer, merchant and PSP. These costs are a function of the processing and communication requirements of the payment protocol, including the dependency on online involvement of the PSP, requiring high availability. Indeed, most of the research on micropayments, and several of the deployed systems, focus on minimizing the processing and/or communication costs. In particular, many efforts have focused on reducing the processing costs by avoiding public key operations. Another area that received a lot of attention is reduction in communication requirements, and in particular, allowing offline or semi-offline payments (in particular, `stored value` offline payments, where the merchant and browser are in direct connection but disconnected from the PSP). See more details in Section 5 below.

There are several additional cost factors, which are less significant or easier to deal with, such as:

- **Bookkeeping and auditing costs:** many payment systems have substantial bookkeeping and auditing mechanisms and costs. It is tempting to suggest that these costs can be eliminated by simply not logging and auditing micropayments, or keeping only very partial and temporal logs. However, accurate logging and auditing is often required by law, and may also be necessary to provide non-repudiation for efficient dispute resolution and to detect fraud. Bookkeeping and auditing costs may be reduced by secure automated *record aggregation* mechanisms, e.g. [HM02], whereby the customer signs a single document which is archived instead of multiple separate documents (similarly to the presentment of a monthly statement by utilities). Record aggregation may reduce also help to protect the privacy of the customer, by not keeping track of individual transactions for long. Other approaches try to protect the privacy of the buyer even further, by preventing the PSP from identifying payments of a particular customer, using one of the many anonymous (digital) cash protocols, e.g. [C92]. Some, e.g. [B99], believe that anonymous payments would also be less expensive, by avoiding bookkeeping (almost) entirely and preventing disputes and chargebacks.

- **Point-of-sale integration costs:** these are the costs for a merchant to set up merchandise for sale, and of publishing information and services. These costs can be reduced by simple and automated tools for the merchants; for small merchants, and for initial phases, hosting services by the PSP may also be desirable. However, these are one-time expenses and should usually be rather insignificant in the long run.

- **Credit risk:** when the customer is charged for the aggregated payments only after the purchases are made, the customer may refuse to pay her PSP. Often, PSPs eliminate this risk by requiring funds to be deposited in advance. This problem also appears when interoperating

between multiple PSPs, where one PSP ends up owing the other PSP; in this case, pre-payment is rarely a solution, since usually both PSPs may end up owing the other PSP. This becomes a risk management issue, with associated costs of estimating and containing the risk.

12.3 Disputes and Chargebacks

As noted above, disputes and chargebacks and in particular their processing costs, are of the most significant expense factors for online credit card purchasing. In credit card purchasing, and to some extent in any remote purchasing such as through the Internet, there are laws protecting consumer's right to dispute and reverse transactions that were not approved by the consumer, and often to some extent also transactions that were not properly fulfilled. This creates a difficult challenge to the designers of micropayment systems. Clearly, it is difficult to provide a sustainable business for transactions under a dollar, when the dispute resolution process itself costs around 50$, and with a substantial fraction of the transactions being disputed. In general, disputes and chargebacks for electronic payments fall into four main categories:

- **Disputes on whether Payment was Approved by Consumer**: Consumer claims that she did not *approve* the payment.
- **Unauthorized Overspending Chargebacks:** PSP claims that it did not *authorize* the payment, and that there are not sufficient funds in the customer's account to cover it (in systems where the merchant should receive authorization from the PSP for each payment).
- **Disputes on Delivery and/or Quality**: Consumer claims that the merchant did not *deliver properly* the goods or services as ordered.
- **Chargebacks due to Consumer Default**: PSP claims that the consumer *defaulted,* and did not provide necessary funds.

We now discuss each of these categories for disputes and chargebacks, beginning with the last two, which we believe should simply be disallowed for micropayment systems. But first let us consider an example.

12.3.1 Example: First Virtual Payment System

A simple example is the First Virtual payment system, illustrated in Figure 5, which was of the earliest payment systems proposed [N95]. One of First Victual's goals was to offer lower fees for small charges (compared to credit cards) and avoidance of chargebacks. First Victual's system used two (non-cryptographic) security mechanisms for each purchase: a secret `first virtual account number` (FV#) and e-mail approval. The consumer sends the (secret) FV# to the merchant, who forwards it to the First Virtual Net server (on the Internet). The First Virtual Net server sends the e-mail approval request to the consumer, who approves (or declines) in a message back to the First Virtual Net server. Only after the consumer approves the payment, the First Virtual Net server credits the merchant's account with First Virtual and informs the merchant that the payment is Ok. A credit card transaction is performed (immediately or

periodically) to collect the funds from the customer's credit card account, whose details were kept in a separate First Virtual `Credit Card Server`.

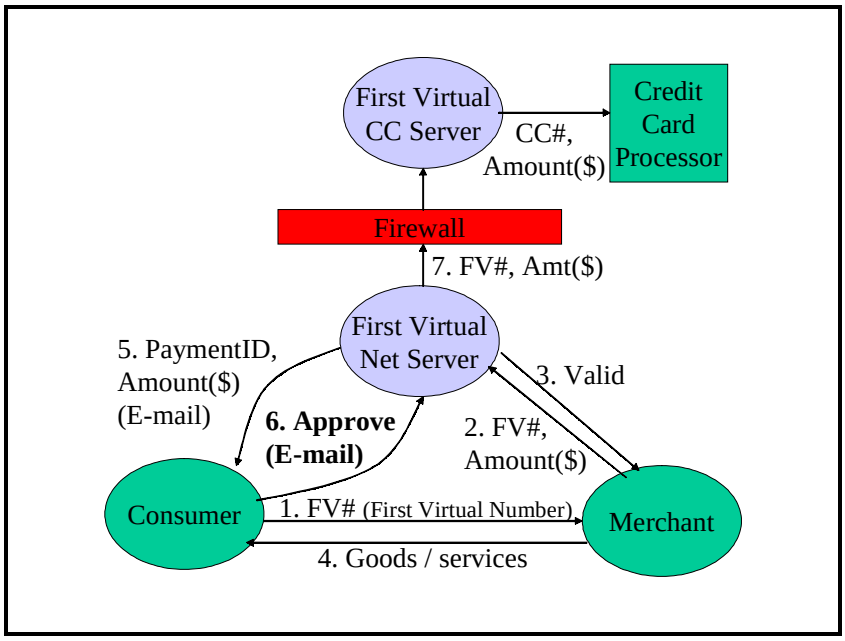

Figure 5: First Virtual's Centralized PSP based on e-Mail confimation.

The payment approval process of First Virtual is subject to attacks, since standard Internet e-mail is not secure; this makes the entire authorization process insecure. It is particularly easy to inject e-mail messages with incorrect source identification; First Virtual protect against this attack by including a random payment transaction identifier in the approval request e-mail message (PaymentID). However, the attacker may be able to intercept the e-mail message in transit to the consumer, and then send the approval response with the correct PaymentID. To protect against this, First Virtual suggested that consumers might use available secure e-mail software to digitally sign their payment approval responses.

The use of e-mail software also allows First Virtual to offer their customers a certain level of offline purchasing. Namely, the customer may approach a merchant and provide the First Virtual number, without contact to the First Virtual server. However, the merchant does not know if this is a valid payment, until receiving the payment authorization. First Virtual expectation was that for small amounts (micropayments), many merchants will agree to take the risk and provide services without waiting for the authorization.

12.3.2 Consumer Default and Disputes on Delivery/Quality

We believe that micropayment systems should not allow disputes on product delivery or quality, as well as chargebacks due to consumer default. The basic reason is that there may not be a long-term relationship between the buyer and the merchant. Therefore, the merchant can

hardly manage such risks, e.g. the risk of the consumer defaulting on its payment obligations to its PSP.

In particular, consider disputes about whether the bought product or service was in fact delivered, or about the quality of the product or service. This problem can exist for many face to face purchases as well as for remote purchases, however it is more common and acute in remote purchasing, especially on the Internet, where there are many potential merchants, and reputation is harder to establish. Some proposed and deployed systems attempt to avoid disputes on product delivery, by using a trusted third party as an `escrow agent` that ensures delivery of product (to buyer) and of payment (to merchant). Most of the deployed systems, e.g. [E02, L01], focus on escrow of high-value physical products (or legal title, e.g. to real estate), and in particular involve minimal fees of several to dozens of dollars. Proposals were also made for escrow for digital content (for efficiency, the actual escrow is often of a key to decrypt the content, see [CTS95]), which may seem applicable to micropayments. However, it is impossible to completely automate the resolution of disputes regarding quality of (most) products and services; and in fact even disputes on delivery are not automatically resolvable, in the sense that merchants may deliver product or service of no value (but this can usually only be detected with manual intervention). Therefore, the third party will require manual intervention for every dispute, making the process rather expensive, and inappropriate for micropayments. Therefore, in micropayment systems, consumers should resolve any disputes on the delivery or quality of the service or product directly with the merchant. On the other hand, when purchasing via programmable devices, consumer are well positioned to check reputation of merchants and to reward good service and merchandise or punish merchants providing bad service or products, using standard communication facilities (forums, mailing-lists, newsgroups, web site, etc.), reputation and evaluation services, e.g. queried using PICS, and secure reputation and trust management systems. We conclude that it is impossible to use low cost, technical, automated mechanisms to resolve or prevent dispute regarding the delivery of the goods or services, and therefore micropayment service providers may be better off prohibiting this kind of disputes.

By disallowing disputes related to proper delivery/ quality and chargebacks due to consumer default, micropayment systems can focus on disputes related to payment approval (by the customer) and payment authorization (by the PSP).

12.3.3 Unauthorized Overspending Chargeback

In some payment systems, a customer may overspend, i.e. approve more payments than the funds available in her account. In this case, the PSP may wish to refuse to pay (or chargeback) the merchant. Some micropayment systems allow this, claiming that merchants can accept this risk, since the amounts are small, and especially for selling information (with negligible cost for the merchant for each extra transaction).

However, since merchants do not have a long-term relationship with consumers, they often require secure *payment authorization* from the PSP, such that payments properly authorized by the PSP cannot be reversed. Micropayment schemes often focus on reducing the overhead required for payment authorization, especially on the PSP, by adopting different strategies.

The overhead for payment authorization has two major elements: communication and computation.

First, consider the computational overhead, and in particular, whether the PSP must perform computationally-intensive operations such as public key signature for each payment. Public key signatures are the main technique for achieving non-repudiation. For large amounts, and when the merchant does not trust the PSP, the merchant may require non-repudiation of the payment authorization from the PSP, to make sure that the PSP is committed to transfer the funds for the payment. In particular, in multi-PSP scenarios as illustrated in Figure 4, the merchant may require non-repudiation from the customer's PSP. In Section 12.5.2, we discuss techniques for achieving non-repudiation with reduced computational requirements, compared to digitally signing each payment authorization.

When the transaction amounts are small relative to the value of the PSP-merchant relationships, then the merchant may not demand non-repudiation for every (micropayment) transaction, and agree to received only secure payment authorization from the PSP (not signed). When the aggregated amount of authorized (but not signed) payments exceeds some merchant-specified threshold, the merchant may require the PSP signature authorizing the total, aggregated amount (providing non-repudiation). By not signing (and validating) every micropayment authorization, the PSP (and merchant) may save some computations.

Consider now the communication required for the PSP to authorize payments. Several micropayment schemes support *offline payment authorization*, i.e. transactions where the client communicates with the merchant, without involving the customer's PSP to authorize the payment. Offline transactions can be used where communication with the PSP during the purchase process is impractical, e.g. for payment using direct communication between consumer and merchant, without requiring connectivity and communication with the PSP. Usually, the PSP will require the merchant to return funds in case of double spending; but sometimes PSPs may assume limited liability, usually when the consumer is using a tamper-resistant hardware that authorizes the payments on behalf of the PSP (and keeps track of the available funds in the consumer's account). The PSP may pre-authorize payments up to a predefined amount for a particular merchant, but rarely useful (since it is hard to predict purchases) and indeed rarely used.

Therefore, whenever communication with the PSP during the purchase process is feasible, micropayment schemes use it to perform *online payment authorization.* Sometimes, as in Figure 2, all communication between the consumer and the merchant flows through the PSP (or a gateway associated with it), in which case the online payment authorization does not add substantial overhead. In other cases, the customer or the merchant must contact the PSP to request payment authorization. When possible, it is preferable for the customer to request the (online) payment authorization, so that the merchant can respond to the purchase request from the customer immediately, after local validation, without having to keep the connection with the customer open while requesting authorization from the PSP. Some micropayment schemes try to minimize the online authorization communication, e.g. by requesting authorization for only some payments (randomly or based on purchase amount and/or other parameters); see Section 12.5.1 below.

12.3.4 Disputes on whether Payment was Approved by Consumer

We now focus on the most important (and common) type of disputes and chargebacks, where the customer claims she did not approve the payment transaction. To understand this problem better, consider the common mechanism of performing credit card transactions over the Internet (or phone), by sending the credit card details to the merchant. Usually, the credit card details are encrypted on transit using SSL/TLS [R00, RFC2246], as illustrated in Figure 6. Since payment approval is only by inclusion of the credit card number in the transaction, an attacker with access, at any time, to the card number may invoke an unapproved transaction. As a result, there are many disputes in Internet transactions, much larger than their proportion of all credit card transactions. The dispute rate in Internet transactions was so high that it caused losses to credit card issuers, and in the first years of Internet commerce it was growing rapidly (for example, see practical experience in [SRW01]). The rate of disputes substantially reduced with the introduction of substantial penalties to merchants with high dispute rate[3], which may indicate that many disputes were due to merchants that charge differently from agreed with or understood by the customer (merchant fraud or aggressive sale techniques). Many other disputes are due to unauthorized purchases by a third party which somehow got hold of the customer's details, by exposure in Internet or non-Internet transaction, or otherwise. In addition, there are disputes made by customers denying payments they actually approved, knowing that there is no way to distinguish between them and unapproved transactions, namely that there is no non-repudiation (this is considered consumer fraud).

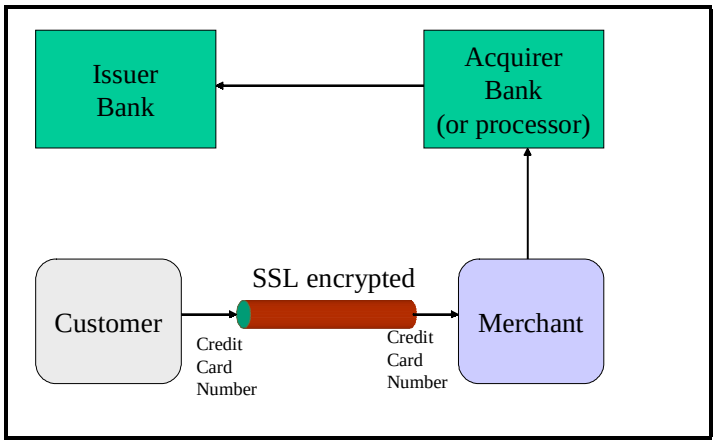

Figure 6: SSL Credit Card Payments

Due to the high cost of dispute resolution and charge-backs, there have been several proposals, so far none too successful, to improve the security of the consumer approval process for credit card payments over the Internet. This resulted in the iKP protocol [BGH*00], which evolved to the SET credit card payment standard [SET, L98]; and later in other proposals such as 3D Secure and Secure Payment Application (SPA) [L98]. Most of these proposals attempt to emulate `face to face` transactions, where the customer physically signs a payment order (and

[3] Typical penalties for disputes: merchants with over 2.5% disputes among their transactions, pay 100$ per dispute [SRW01].

presents a physical card with the same signature), often by using digital signatures. The goal is to reduce the number of disputes by preventing unauthorized use and preferably ensuring non-repudiation.

Reducing the number, and associated cost, of disputes resulting from claims of payments which were not approved by the consumer, is even more critical for micropayment systems, where the average transaction amount is several orders of magnitude below the reported cost of handling a dispute in the credit card business (which is around 50$, see e.g. [SRW01]). In the rest of this section, we discuss three major approaches to reducing the number of disputes and chargebacks due to consumer denial of approving the payments, and the associated costs.

The first approach is to perform a **secure payment approval** process, confirming to the PSP that the consumer approves the transaction. This would prevent an attacker from initiating transactions without consumer approval, which the consumer would later dispute. It also allows the PSP to detect when a consumer is disputing a properly approved transaction, although the PSP may not be able to `prove` to a third party that the transaction was properly approved, as long as non-repudiation is not provided. Possibly, the agreement between the consumer and the PSP may indicate that the consumer is willing to accept the records kept by the PSP of transactions approved by the consumer, and not to dispute them.

A side benefit of secure payment approval is to provide better security and convenience to the consumers. Indeed, with current credit card system, consumers can always reverse a transaction they did not approve, yet many consumers are still concerned about them and some even avoid internet shopping as a result. The concerns are mainly fear of not realizing an unauthorized transaction happened, the overhead to the consumer of reporting an unauthorized transaction, and the potential damage to the credibility of the consumer with the issuer; some consumer are also not sufficiently aware of their right to dispute transactions that they did not approve. These concern may be more significant for micropayment services, which by normally provide less value to the consumer.

The second approach takes another step and provides not only secure payment approval, but also **non-repudiation** of the payment approval. Namely, whenever the PSP authorizes a transaction, it will retain a proof allowing it to convince a third party that the consumer indeed properly authorized the transaction. The proof will usually be in the form of a payment order digitally signed by the consumer. When complemented with an appropriate agreement with the consumer, accepting the digital signature or other proof as sufficient evidence to the consumer's approval of the transaction, this may be sufficient to prevent disputes. The agreement may also require the consumer's signature only when the aggregated amount of approved payments exceeds some pre-defined threshold, thereby amortizing the computational overhead over several micropayments.

The third approach takes a somewhat radical view, namely, that ensuring consumer agreement to certain proof of transaction, even including digital signatures or similar means for technical non-repudiation, may not be sufficient to prevent disputes. The reasoning is that the consumer may still claim that there may have been a compromise of the signing equipment and/or apply statutory consumer-protection rights that would prevail over any agreement in which the consumer agrees to accept certain evidence by the PSP to the fact that the consumer approved

the transaction, e.g. a digital signature. Therefore, in this approach, the micropayment system will operate in such a way that the PSP will not maintain track of individual payment transactions and would therefore be *unable to reverse transactions*. We call such transactions **irreversible transactions.** This payment instrument is often referred to as **`bearer certificates`**, since the payment is done by passing a message (`token`, `bearer certificate`) to the merchant, who deposits it at the PSP, but the message does not indicate the identity of the consumer. Designs that try to ensure irreversible transactions sometimes use *anonymous payment (`digital cash`)* mechanisms, such as blinded signatures.

We discuss each of these approaches in more details in the following subsections.

12.3.4.1 Secure Payment Approval

Secure payment approval is relatively simple between two parties with long-term relationship, such as the customer and her PSP. In this case, the payment approval is by an authenticated message from the customer to the PSP. Often, end-to-end authentication of the payment approval message from customer to PSP is possible (typically, when the customer communicates directly with her PSP, as in Figure 2 or Figure 3). Such end-to-end authentication of the payment approval requires only standard, pre-installed software components. The payment approval process, when invoked by the consumer, as illustrated in Figure 3, using only standard browser, has the following steps:

1. The merchant presents the offer of goods or services to the consumer. In the usual browsing scenario, the offer is in a regular or special hypertext link, sometimes called a *per-fee-link,* invoked by the consumer to pay, and encoding the payment details necessary to create the payment order. Inside the per-fee-link and possibly also outside it, the merchant provides the *offer description,* which is text and/or graphics describing the offer and presented to the consumer. We discuss some standardization efforts for the offer and per-fee-link later on.

2. Typically, the browser displays the offer description. The customer invokes the payment process, typically by pressing the per-fee-link. This results in passing the per-fee-link parameters to the PSP, providing it with the payment offer details.

3. At this point, in most cases, the consumer was already presented with the payment details in the description sent with the per-fee-link. However, a malicious merchant could have provided description that differs from the payment details sent to the PSP. Therefore, the PSP has to validate that the consumer actually approves the payment details. This is usually done by sending a payment approval request, e.g. as a web page. The user approves (or declines) this request. The PSP should verify that the approval came from the consumer. This verification is usually done using one of the following techniques:

    a. The consumer types user name or identification number, and a password. This solution is most simple, and in particular requires only standard browser facilities and no dedicated `wallet` software for the payment process. On the other hand, this solution requires manual typing by the consumer for each transaction. Some browsers and utilities may reduce the amount of typing involved by auto-

completion form filling features, often at the price of exposing the password to malicious programs or users accessing the consumer's computer. The password can be encrypted on transit to protect from eavesdropper, by using the standard browser facilities, namely encrypting the communication using SSL/TLS [R00, RFC2246].

b. Most browsers also support a *secure cookies mechanism* that may be used for (easier) authentication and confirmation of the approval of the payment. Cookies are (small) files which web servers can send to browsers, and browsers normally store them on the local machine, and send them to web servers when requesting a web page with address matching an address template in the cookie. Therefore, the PSP can easily request the browser of each consumer to store a cookie identifying that consumer; the browser will send this cookie only to this PSP. To prevent an attacker from creating unauthorized cookies, the cookie contains a secret, used much like a password. However, the PSP does not need to store the password of each cookie; it suffices to store one *master-cookie key* and have each cookie contain a message authentication code (MAC) computed using this key over the consumer ID and other details in the cookie, $MAC_{master\text{-}cookie}(consumer\text{-}ID,\ldots)$. To prevent an attacker from learning the cookie by eavesdropping the communication, the cookie can also be encrypted on transit using SSL/TLS. Note that cookies are normally exposed to malicious programs or users accessing the consumer's computer.

c. The payment approval, with details, may also be authenticated by sending it with a cryptographic authenticator. Such cryptographic authenticator may be a Message Authentication Code (MAC) using a secret key shared between the consumer and the PSP, or a digital signature using the private key of the consumer and validated using the corresponding public key (known to the PSP). Unfortunately, most existing browsers and operating systems do not offer the necessary message authentication or signature facilities built-in; therefore this requires the consumer to install additional software such as a payment wallet.

4. Once the PSP validated that the consumer properly approved the payment, then the PSP issues an authorized *Payment Order (PO)* sent to the consumer. The payment order is often authorized by being signed by the PSP or authenticated using a key shared between the PSP and the merchant. If the payment order as sent to the consumer may be `stolen` by an attacker and used to obtain goods and/or services to the attacker, this communication may be encrypted e.g. using SSL.

5. The consumer's browser usually processes automatically the authorized payment order, by sending it as a request to the merchant. When a dedicated payment wallet is used by the consumer, it may modify the payment order before sending it to the merchant, e.g. adding the consumer's signature or otherwise `validating` the payment order.

6. The payment order should be validated by the merchant, checking it was properly authorized by the PSP. This may be done using a MAC key shared between the PSP and the merchant, or by the PSP's public key signature. Once the payment order was validated, the merchant should provide the ordered goods or services to the consumer

7. In an offline, `batch` process, the merchant deposits the payment orders and receives the funds

Alternatively, and in particular when the customer does not communicate directly with the PSP, but only via the merchant, then the customer can authenticate the payment approval by appending a message authenticator to it. The authenticator may be Message Authentication Code (MAC), using a secret key shared between the PSP and the customer, or a digital signature, using the customer's private signature key and validated using the customer's public key. This may require dedicated payment software in the customer's computer (*local wallet*); we discuss this issue in Section 12.4.

12.3.4.2 Non-repudiation for Payment Approval

The solutions discussed so far provide different levels of security for the payment approval from the consumer. However, they do not completely prevent fraud; hackers may guess, steal or otherwise expose passwords and keys, e-mail may be misrouted, hackers may expose cookies in transit and other users of the same computer may expose keys and passwords. In particular, all of the mechanisms above are vulnerable to a software virus in the consumer's computer or device (although many existing mobile devices may not be susceptible to viruses, due to limited functionality). Furthermore, the process depends completely on the trustworthiness of the PSP; a corrupted PSP (possibly hacked by an employee or third party) could claim that the consumer authorized transactions incorrectly. The corrupted PSP can also compute any authenticators and authentication keys. Definitely, therefore, this mechanism does not ensure non-repudiation. Hence, consumers may still dispute their transactions, claiming unauthorized use, rightfully or possibly to avoid payment (without justification). (In addition, of course, customer may dispute a transaction claiming dissatisfaction with the service or merchandise, as discussed earlier; but we focus on disputes claiming unauthorized use.)

Some micropayment systems, e.g. First Virtual ([N95], or see Section 12.3.1 above), solved the remaining disputes problem simply: they automatically refunded each dispute, and in this case did not pass the funds to the merchant. In fact, to completely protect First Virtual, they simply did not pay the merchants until the dispute period expired; thereby making additional profit from the float as well as avoiding the dependency on the merchant to actually pay them back. Considering the credit card experience showing the disputes are largely due to problematic merchants and merchant practices, there is some justification to this policy. Yet, it is clearly open to abuse by consumers, and raises a substantial business risk for merchants. Indeed, one my suspect that if a payment product adopting such a fully automated refund will become widely popular, then (automated?) cheating may become commonplace. We therefore focus on systems which do not automatically accept all disputes. As noted above, we discuss only disputes claiming unauthorized transactions.

Many micropayment systems take the opposite approach, and simply forbid any disputes claiming unauthorized use (and therefore, usually, any form of dispute with the PSP). Often this is done simply by requiring the customer to agree to accept the record of transactions kept by the PSP. However, in many cases this mechanism may be unacceptable and possibly even illegal, especially when consumers are not fully protected against unauthorized charges. It appears that in order to completely disallow disputes, it is highly desirable that the PSP is able

to prove to a third party that the consumer actually authorized each payment. This requires *non-repudiation.*

We can achieve non-repudiation by using digital signatures such as RSA to sign the payment approval from the consumer computer or device to the PSP. The customer must also agree in advance that he is responsible for payment orders digitally signed using his private key (possibly with some restrictions, e.g. maximal amount); the PSP should confirm that this agreement is legally binding in the relevant jurisdiction. The private signing key will be installed by the customer in his signing software, service and/or device.

There is, unfortunately, one serious pragmatic problem with digitally signing payment orders by the customer's computer or device: digital signing is *not* a standard, easy-to-use feature of widely deployed[4] operating systems, browsers, or mobile devices. Compare this with the solutions in the previous subsection, which do not offer non-repudiation, but on the other hand do not require any new software in the consumer's computer or mobile device, or any complex operation for a consumer wishing to use them.

An obvious solution is to require the consumer to install digital signing software. Indeed, digital signing functionality is often one of the main roles of a *wallet* utility, installed on the consumer's machine. Wallets can perform other useful functions such as payment management and logging, but their deployment is difficult and expensive. We discuss this important issue, and some possible solutions, in Section 12.4 below.

Another concern is that digital signing technology is relatively computationally intensive. A large amount of work on micropayment systems is focused on providing non-repudiation while minimizing or avoiding completely the use of public key operations; see Section 12.5.2 below.

12.3.4.3 Irreversible transactions and `bearer certificates`

We now discuss the third approach for preventing disputes, which takes a somewhat radical view, namely: design the micropayment system in such way that it will be technically infeasible to reverse payments, rather than relying on consumer's agreement that properly authorized (or digitally signed) payments cannot be disputed. In this approach, the micropayment system operates in such a way that the PSP will not maintain track of individual payment transactions and would therefore be *unable to reverse transactions*. This kind of payment order is often referred to as **`bearer certificates`**, since the payment is done in by the customer passing a message to the merchant, who deposits it at the PSP, but this `bearer certificate` payment order message does not indicate the identity of the consumer. More specifically, a payment involves two separate phases:

1. Customer `buys` bearer-certificates from PSP (payment approval or withdrawal phase)
2. Customer pays merchant by providing the bearer certificate (payment authorization phase)

[4] Digital signing functionality is available in some versions of the Netscape browser, e.g. 4.04, as well as in some mobile devices.

The PSP does not keep records of the identity of the customer who received each bearer certificate. In some proposals, the bearer certificates are provided, during withdrawal, in a `blinded` manner, which does not allow the PSP to identify which bearer certificate was sent at which withdrawal; for more details on such `blinded withdrawal` and digital cash, see Chapter ….. In other systems, weaker anonymity is used, and the PSP simply does not maintain the records linking from the bearer certificate to a particular customer.

Bearer certificate systems should ensure that an attacker cannot `steal` the bearer certificate on transit from the PSP to the customer, from the consumer to the merchant or from the merchant to the PSP. When the consumer is using standard, available browser software, and the bearer certificate is forwarded to the merchant exactly as sent from the PSP, the communication may be protected by using browser provided encryption (usually SSL/TLS). In some cases, this may be avoided, by requesting a bearer certificate specific to the requirements of this particular consumer; but this may conflict with the requirement that bearer certificates cannot be linked to a particular consumer and purchase (to make it impossible to dispute transactions).

When the consumer uses dedicated wallet software, then the wallet may `activate` the bearer certificate it receives from the PSP; a bearer certificate which the wallet did not activate is not considered valid. Therefore, the attacker will not gain anything from a copy of the bearer certificate on transit from the PSP to the consumer. This activation may be as simple as attaching a random number *x* to the bearer certificate, where on payment approval the consumer provided the PSP with the result of a cryptographic, one-way hash function *h(x)* applied to *x,* and the bearer certificate is linked to *h(x),* e.g. by including the PSP's signature on *h(x).*

In any case, using `bearer certificate` it should be impossible to link back from the payment order to the identity of the customer or to the withdrawal transaction in which the customer bought the bearer certificate. Therefore, the PSP simply has no way to reverse a payment done using the bearer certificate. Therefore, disputes are technically impossible. Proponents of this approach [H98a, B99] argue that bearer certificates are the only way to avoid disputes, since consumer-protection laws may overrule any limitations on disputes included in the agreement between the consumer and the PSP. On the other hand, using a bearer-certificate, the consumer does not receive from the payment system a receipt of having paid; this could be a substantial disadvantage for some applications.

Even if bearer certificates are used, it seems that for the goal of avoiding disputes, it may be acceptable for the PSP to know the linkage between the customer and the bearer certificate at the time of the withdrawal. This seems acceptable, as long as the PSP always erases this information later and does not provide a proof of it to the consumer. Clearly, such a solution is much simpler than the techniques used for digital cash.

12.4 Customer Acquiring and Support Costs

Businesses are often spending large amounts to acquire new customers and to retain and support existing customers. Micropayments services have very small revenues per transaction and per customer; therefore, it is especially critical to minimize the average, amortized costs of customer acquiring and support efforts. In the following subsections, we discuss three

techniques for minimizing costs and maximizing the number of payment transactions, together ensuring low amortized costs per transaction:

1. Enable simple to use, intuitive `click and pay` mechanism for payment approval. This have the dual impact of encouraging usage (more payments) and reduced support and acquiring costs (easier to use – less questions, easier to `sell`).

2. Support customers without requiring installation of a PSP provided and supported local wallet application. This can reduce the substantial costs of providing wallet applications and supporting them (for multiple platforms), as well as of convincing customers to install the local wallets.

3. Allow interoperability among the PSPs, namely, a customer of one PSP can pay a merchant of another PSP (as in Figure 4 above).

12.4.1 `Click and Pay` using `Per Fee Links`

The goal of micropayment systems is to enable low-value transactions. As such, it is important that the user interaction will be as natural, convenient and quick as possible – ideally, `click and pay`. In fact, if the customer is not willing to pay much for a product or service, she may also not be willing to waste a lot of time and energy in the payment process. An easy, convenient and fast `click and pay` process also reduces costs for customer acquiring and support. The `click and pay` payment process becomes a natural extension of the familiar web surfing interface: to buy information or service, the user just clicks on *per-fee link*, much like clicking on a normal hyperlink.

We must ensure that the user pays only intentionally, i.e. the seller cannot trick the user into pressing a per-fee link without the user being willing to pay the price charged. For example, the IBM Micro Payments system [HY97, H98] provides per-fee-links that appear very similar to regular hyperlinks, by adding cues for payment. Specifically, when the cursor is over the per-fee link, the shape of the cursor changes to either a dollar sign (if the amount is over a user set threshold) or a cent sign (if the amount is under the threshold). Furthermore, the exact price is indicated in message/status area of the browser (where it normally writes the URL of the hyperlink); and the customer can specify a maximal amount for `click and pay`, such that over this a pop-up box will require confirmation.

There are multiple ways for implementing per-fee links. The Web consortium developed a proposal [M00] for per-fee-link syntax (this proposal was suspended since there were only few implementations). The proposal supported several ways for specifying and displaying per-fee-links, as <Embed>, <Applet> or <Object> elements; all of these require extensions to the standard browsers to display the per-fee-link, such as a plug-in, ActiveX control, or an applet. This requires installation of `wallet` software on the customer's machine.

It is sometimes also possible to provide per-fee link using only standard browser, and display the price simply as added text to the link. Consider Figure 2, where the communication between the customer and the merchant flows through the PSP or a gateway associated with the PSP. In such scenarios, the PSP may modify the per-fee-link on its way from the merchant

to the customer, and indicate the purchase details information (in particular the price) by adding the description of the payment details as textual and / or graphical hypertext link as part of the hypertext content sent to the consumer. For example, when using HTML:

```
<A HREF="uri">click here to receive the song [5cents charge]</A>
```

The price information (`[5cents charge]`) was inserted or validated by the payment gateway (so that the displayed amount will be identical to the charged amount). The consumer indicates agreement by simply selecting (clicking on) this link. The transformation of the page sent from the merchant is especially natural in mobile scenarios, where the mobile gateway often creates the encoding (HTML or otherwise) appropriate to the consumer's device display capabilities.

Some caution is necessary, however, when using a regular hypertext link with the price added to the textual message as shown above, to prevent a merchant from invoking payment from a seemingly free (or less expensive) link. Namely, the PSP must accept as approval only requests which result from the consumer following the link it modified; the risk is that the merchant will insert in a page sent to the consumer a link invoking the payment procedure in the PSP but different (seemingly free) text.

To avoid this threat, the PSP may filter the hypertext to remove any fraudulent per-fee-links inserted by the merchant. Another technique to prevent such fraud is to customize the per-fee links, e.g. by encoding in the hypertext link (the uri field in HREF="uri") an authenticator such as MAC_k*(price, description, clientID, time, requestID)*. The PSP (only) knows the key *k* used in calculating the authenticator. The *price* and *description* fields ensure that the purchase conforms to what the customer approved. The *clientID*, *time* and *requestID* fields prevent the merchant from copying the authenticator field from a previous request (stateless server use *time,* and servers who can remember states use *requestID*).

In addition, the gateway should validate that the page, as sent to the consumer's computer or device, does not contain script that may modify the page presented to the consumer (e.g. changing the description presented to the consumer). See [HN98] for techniques to validate that a web page is not modified by a script contained in it.

12.4.2 Local Wallets and Server Wallets

Acquiring customers, and helping them when they encounter difficulties, is difficult and expensive. It is difficult to convince customers to sign up with the system, open an account, and do any additional operations such as installation of a wallet (if necessary). Acquiring customers may therefore require substantial investment in the form of advertising and incentives, and even then, it usually takes substantial time to convince a small fraction of the potential customers to try the system.

When customers use the system, the costs of answering customer inquiries, which could be technical, financial or administrative, can be significant. Customers are usually expecting financial service providers, including micropayment PSPs, to provide highly available and free

customer support. The average cost per customer call is substantial and far exceeds the typical cost of micropayment transactions, not to mention the fees. These costs may exceed the thin profit margin per transaction, especially of a micropayment service provider.

Customer acquiring and support costs depend significantly on whether the customer has to install and use dedicated *local wallet* software to authorize micropayments, or whether the customer can use pre-installed, general-purpose mechanisms such as the browser and/or e-mail utilities, with the payment functionality provided by a *wallet server* in the network. Dedicated, local wallet software, running on the customer's computer or device, can provide better user interface and functionality, e.g. provide transaction log. Most importantly, dedicated wallet software can perform public-key digital signatures for payment orders, providing non-repudiation of the fact that that customer authorized the payment, and possibly avoiding dispute resolution and chargebacks.

However, it is important to realize that a malicious local user, or program, can often bypass local wallet software. In particular, when running insecure operating systems such as Windows™, any malicious program (e.g. a virus) or any other user of the computer will be able to perform unauthorized signatures. Therefore, even a local, dedicated wallet cannot completely prevent unapproved payments.

Furthermore, it is substantially harder to motivate customers to download and install dedicated wallet software, compared to opening an account using purely pre-installed, general purpose mechanisms (e.g. browser). Furthermore, dedicated, locally installed wallet software requires substantially more support costs, e.g. to resolve installation issues, and to support different operating systems. As a result, wallet software results in high customer acquiring and support costs, and a lower adoption rate. Another problem for local wallets is that consumers may use multiple devices and computers, e.g. at home, office and while mobile, resulting in coordination problems between multiple wallet installations of the same consumer. Indeed, as shown in [K01], there have been only failures so far in efforts to introduce wallet software, for micropayment as well as for credit card and other payments, and essentially all existing deployments support a wallet server.

However, a server wallet operated by the PSP has some significant drawbacks as well, in particular in forcing the consumer to completely trust the PSP, which may necessitate dispute resolution with its associated high costs. In order to support micropayments, without allowing disputes, the PSP may find it necessary to offer the customer control over the payment authorization, possibly as an option.

12.4.3 The future: multiple and third-party wallets

We expect that in the future, PSPs will offer their customers to either use the PSP's server wallet, in which case they agree to trust it completely, or to use local-wallet or third-party operated server wallet that provides digitally-signed payment orders to the PSP, in which case non-repudiation is achieved by the public key signature. The PSP would offer local wallet software that signs the payment orders. Alternatively, the PSP may allow the customer to use third-party provided wallets to sign the payment orders, where the third-party wallets may be local or server-based. The third parties will offer the wallets as a productivity aid and often as

part of a larger money-management or personal information management software or service, e.g. as part of the services offered by a portal. If some of the operations are especially sensitive, then the PSP may even require approval by more than one of the wallets of the same customer, providing additional protection against unauthorized use (but this additional complexity is probably not necessary for micropayments). By offering customers the option of using local wallets and third-party operated wallets, the PSP should be able to avoid dispute resolution process entirely, and therefore maintain low operational costs.

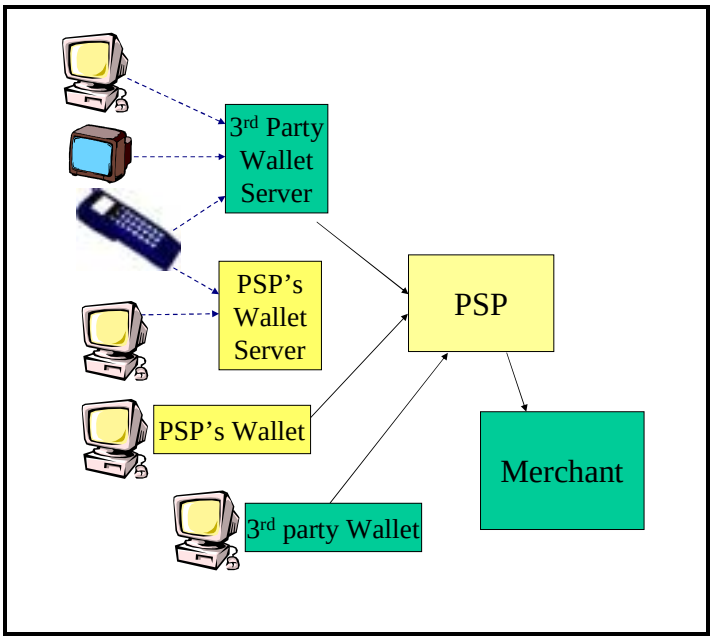

Figure 7: Multiple Wallets, Devices and Computers for the Same Account

In Figure 7 we show such an architecture, where the PSP allows the customer to use the PSP's wallet server, a local wallet provided by the PSP, or a third party provided wallet server or local wallet. The customer may authorize payments from her own computer or another computer, or from any of a variety of devices (different wallet servers may support different devices). This architecture requires protocols for managing an account and authorizing transactions from multiple wallets (software agents of the consumer), ensuring that the entire log of transactions is available to one or more wallets as needed. See such protocols, also supporting record aggregation (see below), in [HM02].

To invoke a wallet server, the merchant usually includes a link to it in the web page, with text describing the offer. It is possible for a single Per-fee link to invoke one of multiple wallet servers as needed; namely, the merchant includes a link to a special `PSP directory` site. The

PSP directory can try to detect automatically the identity of the customer's PSP, e.g. using a cookie stored in the browser and sent automatically to the PSP directory; see Figure 8.

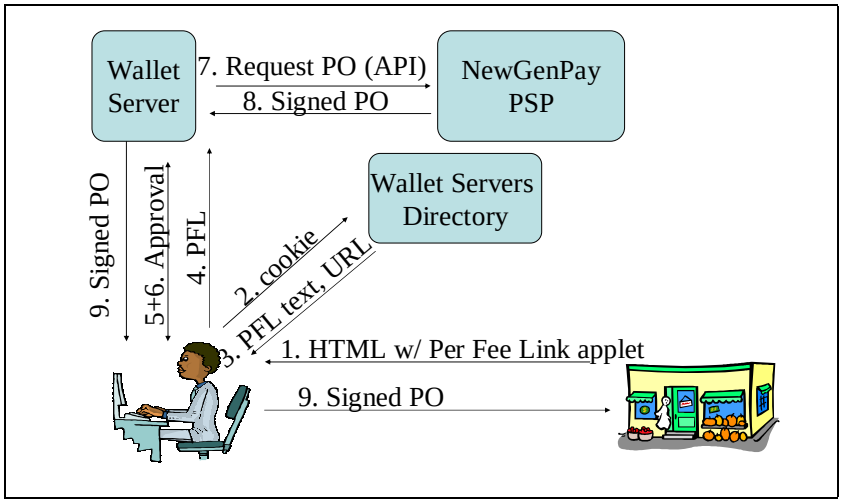

Figure 8: Automated Wallet Server Selection

12.4.4 Building Critical Mass and Acceptability by Interoperability among PSPs

For a micropayment system to be profitable, it is critical to have a large number of transactions, customers and merchants. This is required both to reduce the amortized cost of fixed expenses (such as software development), and to make it easier to acquire customers and merchants (since customers and merchants prefer systems where they can transact with most merchants and customers, respectively).

Many micropayment service providers (PSPs) approach this problem by simply planning (or hoping) to become the dominant micropayment PSP, providing services to most customers and merchants. However, in an open, competitive market, it will probably require a very large investment in order to become the dominant PSP, and there is a substantial risk of failure to meet this goal.

An alternative approach is for multiple PSPs to cooperate, allowing customers of each PSP to pay merchants of each of the PSPs. Cooperation between competitors or *coopetition* is common in different areas of the modern, global economy. Indeed, there are several large networks connecting competitors and allowing them to interoperate, in particular in payments, e.g. the credit card networks, clearing networks between banks, ATM networks, and others. It is therefore reasonable to expect similar global network will handle micropayments.

It is difficult, and expensive, to establish trust among competitors, and even more among many competitors. However, trust is required for allowing an interoperable payment network, where a merchant can receive a payment authorized by different PSPs. Existing, deployed payment networks handle this problem in one of two ways:

- Centralized solution – the payment network is `owned` by a single entity, usually called a *brand*. The brand sets up rules of operation for all the participating PSPs, including possibly technical means such as communication protocols, and audits their operation. The recipient of payment trusts only the brand (and possibly their PSP). The major credit card systems, e.g. Visa and MasterCard, operate in this way. This solution could be appropriate for micropayments as well, if a very small number (say under five) of dominant micropayments brand would become dominant. However, substantial investment is necessary to establish a dominant brand. Such investment may not be economical, for establishing a brand for micropayments, where the expected fees are low.

- Offline clearing solution – the other approach minimizes any assumptions about global trust or organization. This is the approach normally used to deposit checks from remote, unknown banks. Namely, funds are available to the depositor (e.g. merchant) only after the depositor's bank received the funds from the payer's bank. This solution has two problems which seem to make it unsuitable for electronic payments (and in particular micropayments):

  The merchant must wait very long to know if the payment is valid.

  A corrupted intermediary bank (PSP) may deposit the check, receive the funds, but keep the funds to itself while claiming to the depositor that the payment was rejected. With physical checks, the merchant usually receives back the check in case of failure, to prevent this attack (and for other remedies).

We conclude that to provide interoperability among PSPs, it is desirable to ensure security without requiring global trust in all PSPs, as such global trust will require central management and ownership (which may not exist).

12.4.4.1 Open, Decentralized Payment Network

We now describe the design of open, decentralized payment network, as presented in [HY97, H98, HSZ00]. Such network can provide interoperability among many PSPs without requiring global trust in each PSP. The following two principles are the basis for this design:

- **Minimal trust requirements – only between PSP and its account owners:** All account-based payment systems require trust between the party maintaining the account (e.g. bank or PSP) and the account owner (customer, merchant or another PSP). This trust allows the account owner to deposit money and received payment orders in the account, trusting that the PSP will properly credit the account and that the funds will be available to the account owner and used only according to the account owner instructions. Similarly, the PSP may trust the account holder to provide funds to cover any debt due to credit payments from the account. Such direct trust between the PSP and the owners of the accounts kept by the PSP is unavoidable; on the other hand, it is relatively easy to manage as it involves only two parties with long-term relationship. However, no other trust relationships should be required. By avoiding any `global` trust requirements we make it easier and cheaper to manage risks and avoid fraud, thereby reducing costs.

- **Automated dispute resolution between PSP and account owners:** by specifying the exact terms for any transaction related to the account, we can completely automate the resolution of any disagreement between them.

The main mechanisms for achieving these goals are two digitally signed messages: the *Payment Order (PO)* and the *Payment Routing Table (PRT)*. Consider the simple scenario with two PSPs, A and B, illustrated in Figure 9. In this scenario, the customer C accepts an offer (flow number 3) from the merchant M, and pays for it by sending to the merchant a Payment Order signed by PSP_A (flow number 6). The merchant can immediately validate the payment order. The merchant can then deposit the Payment Order, as in flows 7 and 8. Deposit can be immediate or delayed; the merchant (and intermediate PSP B) may some time for possible batching with other deposits for efficiency. The immediate, local validation, without online communication by the merchant, is possible using information that the merchant received from PSP_B in the Payment Routing Table (PRT) signed by PSP_B, in an offline process before the purchase, in flow 2.

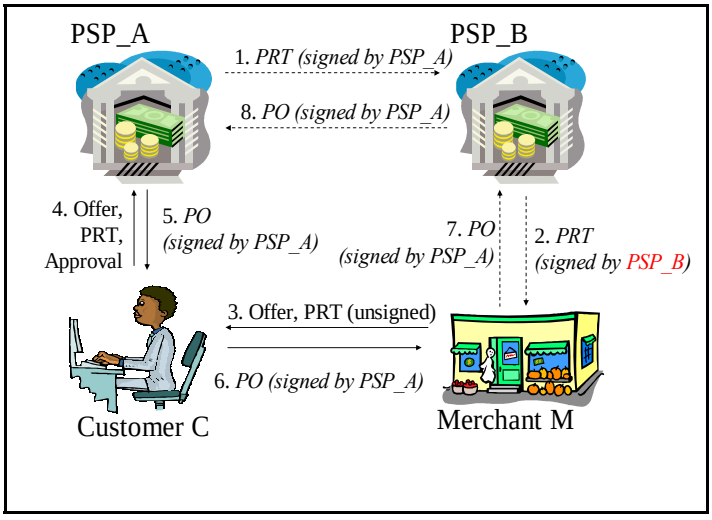

Figure 9: Interoperability between Two PSPs

The Payment Routing Table (PRT) is sent in advance, e.g. daily, from each PSP to all of the entities keeping an account with this PSP and which may receive payment orders for deposit. In Figure 9, PSP_A sends (e.g. daily) PRT to PSP_B, and PSP_B sends PRT to the merchant. For a more complex scenario, consider Figure 10, where we show five PSPs. Normally, each PSP will send a PRT message periodically to all of its merchants and to all of the PSPs keeping an account with it (namely to every entity that may deposit a payment in this PSP). For efficiency, a single PRT message may contain payment routing entries for multiple issuing PSPs (i.e. for payment orders issued by different PSPs).

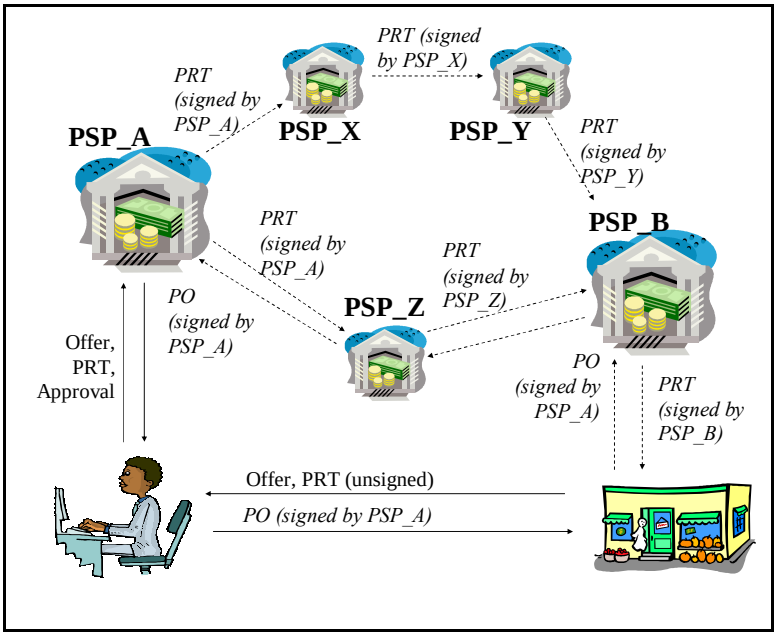

Figure 10: Interoperable PSP Network (Example)

The PRT contains all the information necessary to allow the merchant to process a payment order signed by PSP_A. The PSP sending the PRT will always sign it with its private key (e.g. PSP_B). Payment Service Providers construct their outgoing PRT messages, based on the incoming messages, and on local policies (e.g. for limiting the damage if a particular PSP does not keep its commitments, or for maximizing revenues via adjustment of fees). The PRT, signed by B, will include the following details for every PSP from which PSP_B agrees to receive payment orders:

1. The identity of the account holder (Merchant M).
2. The public key of the PSP issuing the payment order (e.g. PSP_A)
3. The maximal total amount of payments from this PSP allowed (until the next PRT)
4. The fees applied to deposits of payment orders from this PSP (e.g. maximum between 5 cents and 2%). The merchant receives the amount in the payment order minus these fees (no additional, `hidden` fees).
5. Minimal deposit time – deposits will be honored only if made before this time.
6. Acceptable proof of transmission – identifies what is a sufficient proof of transmission of a message and of the time of transmission. This proof allows automated resolution of disputes on whether payments were deposited in time or expired. Typically, the proof will be either the transmission log of the PSP, or a signed receipt from one or more trusted third parties store-and-forward servers. The store-and-forward allow the

depositor (merchant or PSP) and the PSP to resolve disputes due to failure of communication between them; as long as the total pending payments amounts are not very large, the PSP log files may be acceptable.

7. The path of PSPs through which the payment orders will be deposited; in the scenario of Figure 9, this will only include the identity of PSP_B. In the scenario of Figure 10, this may include the path {PSP_B, PSP_Y, PSP_X} or the path {PSP_B, PSP_Z}.

8. The validity period during which the PSP issuing this PRT (e.g. PSP_B) is committed to the terms specified in it. Later deposits may not be honored, unless covered by a new PRT.

9. A unique identifier for this PRT (e.g. counter).

10. Few other technical details may be added to ensure that there are no disagreements, see [HSZ00].

The Payment Order (PO) contains the precise details of the payment, including:

1. The amount to be paid (and currency).

2. The path of PSP's through which this PO is to be deposited.

3. The expiration time (the issuer will not pay for the PO after this time – it will become void).

4. The identifier of the PRT that this payment is applied to.

5. The issuing time of this PSP and a serial number of PO issued at this time (for detecting replays).

6. Possibly, additional conditions on the validity of this PO, such as a hash $h(x)$ sent by the customer, such that the payment order is valid only if the customer attaches to it the pre-image x.

The merchant also sends a (simplified) version of the PRT to the customer, and the customer may send it to her PSP (PSP_A). This allows the customer to choose the best route (and possibly currency) to use in the payment order.

12.5 Computational costs (processing, storage and communication)

In this section, we discuss schemes to reduce the cost of the equipment necessary for the micropayment system and its computational costs. In particular, we discuss the cost due to the processing and communication requirements. Computational and communication requirements are reasonably well defined, and minimizing them is an obvious goal, easily measured, and similar to problems of minimizing communication and processing complexities in many other algorithms. Therefore, it is not surprising that much (or most) of the academic research on micropayments have focused on minimization of processing and communication costs.

However, while these techniques are of algorithmic interest, we caution that the actual impact on cost and performance may be insignificant in many cases, as processing and communication costs are often a negligible compared to the other expenses (and delays) in the system.

Most of the efforts in reducing complexities and requirements focused on one of the following areas:

1. *Avoiding or reducing the use of public key cryptography* (digital signatures), since it is relatively computationally intensive.

2. *Allowing offline (or semi-offline) payments*, where the PSP is not involved during the process of payment (or most payments, respectively). An important form of offline payments are *stored-value payments.*

We now discuss offline (and semi-offline) payments, and then techniques to avoid or reduce the use of public key cryptographic operations.

12.5.1 Reducing communication costs – offline (and semi-offline) payments

As discussed in Section 12.3.3 above, most micropayment systems require secure authorization of every transaction by the PSP, to assure the merchant that the payment is final and that the PSP will not refuse to pay, claiming that the customer spent more than the maximal amount allowed (overspent). Usually, this requires the PSP to authorize each payment online. In many cases, the online authorization is easy and inexpensive, as the PSP controls a gateway on the communication path between the consumer and the merchant, as in Figure 2. However, in other cases, the PSP is not on the communication path; in order to involve it, the merchant must contact it (as in Figure 5) or the customer must contact it (as in Figure 3). This requires additional appropriate communication capabilities and capacities by the PSP and merchant or customer. In particular:

- The PSP must have sufficient capacity to handle the maximal number of concurrent payment authorization requests expected at peak time.

- Either merchant or buyer (or both) must have communication capabilities to the PSP online (during payment process). This is usually the case when paying using communication devices such as mobile phones or computers connected to the Net. In fact, in many applications, it is feasible and cheap to add communication capabilities to either the payer or payee (e.g. add a GSM module to a vending machine). However, there are scenarios where both payer and payee are disconnected and where it is not reasonable to require one of them to add communication capabilities, e.g. payments between two personal organizers.

In this section, we look at mechanisms for avoiding or reducing the online authorization requirements from the PSP, thereby eliminating or reducing the costs of the communication (messages) and of the necessary communication capabilities.

We say that micropayment schemes are *offline* if the PSP is *never* involved during the payment process and *semi-offline* if the PSP is involved only in few payment transactions. Both offline and semi-offline payment schemes may reduce the load on the PSP, and in particular avoid bottlenecks at peak hours. Offline payments may further allow applications where payments are between a (low-cost, mobile) device carried by the customer and a (low-cost, fixed or mobile) point-of-sale device, without any network connectivity to the PSP.

The main approaches to providing offline and semi-offline payments are:

- **No authorization:** a trivial way to avoid online authorization is not to require any authorization by the PSP at all. The risk of overspending is borne by the PSP or the merchant, depending on the agreement between them. However, this requires the PSP or merchant to cover the cost of overspending customers, which may be an unacceptable risk and expense.

- **Random or threshold authorization:** to reduce the number of authorization requests from the PSP, while limiting the risk to the merchant of a payment being cancelled due to overspending, the merchant can decide on whether online authorization by the PSP is required for each transaction. When using randomized or threshold authorization, either the merchant or the consumer wallet can request the authorization; requesting by the merchant is more direct as the merchant decides on the need for online authorization, but requesting indirectly by the consumer's wallet allows a simple, efficient client/server design for the merchant. In early versions of IBM Micro Payments [H98, HY97] we used threshold authorization, where the merchant requests payment authorization from the PSP when reaching a threshold amount (of that particular transaction or of all transactions pending authorization by that buyer). However, we later changed to online authorization, since we found that the its overhead is negligible, that merchant servers may spend considerable resources to keep track of total purchasing per customer, and that merchants are alarmed and confused by the possibility of unapproved payments being cancelled later. We also considered using random authorization, as proposed (independently) in Agora [GS96], where the merchant requests authorization randomly and the PSP identifies and blacklists any overspending customers. However, maintaining and distributing the blacklists may become a bottleneck and open the system to denial-of-service attacks, while on the other hand the overspending is still possible (until detected and until blacklist is updated); this, combined with the simplicity of opening multiple micropayments accounts, makes this solution impractical.

- **Pre-authorization:** the PSP may authorize the customer payments up to a pre-defined limit to a *specific* merchant. Overspending is then limited to multiple payments to that specific merchant, which the merchant can easily detect (e.g. using sequence number). When the PSP pre-authorizes payments (up to some limit, for a given merchant), it actually delegates it authority to the customer, who provides the final payment authorization directly to the merchant or point-of-sale (e.g. vending machine). The customer may digitally sign each payment authorization, or use one of the techniques described below to authorize the payments with reduced computational requirements.

Pre-authorization is a semi-offline technique, as the customer needs to request pre-authorization for each specific merchant.

- **Stored Value Payments:** taking one-step further than pre-authorization, the PSP can avoid online authorization completely by delegating its authority to authorize payments to a *tamper-resistant module trusted by the PSP,* e.g. a smartcard provided by the PSP. The module keeps track of the spending by the consumer, and authorizes payments only as long as the customer does not overspend. In a sense, the funds (value) which the customer can legitimately spend are stored in the device; hence, the name *stored value*. Stored value solutions depend on the temper-resistance of the module, to prevent duplication of money. Tamper-resistant modules seem to require hardware (there are some efforts to create tamper-proof software, by *obfuscation*, but recent negative results seem to indicate that this is difficult or impossible). This introduces significant installation costs. Even tamper-resistant hardware is often subject to attacks; therefore, stored-value protocols should limit the damage due to exposure of the keys of a limited number of modules. To limit the damage, most stored-value systems use a different key for each module and blacklist over-spending modules (but this requires identification of over-spending modules and informing all merchants, a non-trivial undertaking). Some authors claim that stored-value payment devices have the advantage of a total limit to the value lost if the card/device is stolen and abused. However, such a limit is easy to achieve with an account-based solution, simply by setting a limit to the amounts that the customer can spend using a given key.

12.5.2 Reducing Computational Complexity due to Public Key Operations

As motivated in Section 12.3.4.2, non-repudiation of payment approval is highly desirable, as it can help to reduce disputes and detect consumer fraud. The main cryptographic tool for achieving non-repudiation is public key digital signature, typically using the [RSA] or [DSA] algorithms. However, most public-key cryptographic mechanisms, and in particular digital signatures, are computationally intensive operations, compared to hash functions and shared-key cryptographic mechanisms; the ratios in the processing times depend, of course, on specific functions and implementations, but ratios of 100 and even substantially more are quite common. Much of the research on micropayment systems focused on reducing this computational burden, by designing micropayment protocols and systems that avoid the use of public key cryptography, use only a very small number of public key operations, or use more special, efficient public key cryptographic mechanisms. In this section, we review some of these techniques.

We comment, however, that while the goal of avoiding or minimizing the use of (computationally intensive) public key operations is natural and interesting, the actual cost of their processing time may be negligible compared to other costs and overheads in a practical micropayment system. The computation of an [RSA] digital signature, on typical desktop machines, takes only very few milliseconds; validation usually takes even less (when a small public exponent is used). The [DSA] algorithm is about as efficient (but with computation faster than validation). Hardware accelerators can further substantially reduce the overhead. Therefore, it seems that for most realistic micropayment applications, computation and validation of a digital signature on the payment order is not a significant cost factor. We

therefore believe that techniques to avoid or reduce the use of public key operations and in particular of digital signatures and their validation, should only be used in special circumstances, and only when they do not result in more substantial increase in other expenses. For example, a situation where it may be important to avoid digital signatures by the buyer is when the buyer is using inexpensive mobile devices for payments, such as a key-chain gadget or smartcard.

The main techniques for avoiding or minimizing the computational burden due to public key cryptographic operations are:

- Use authentication mechanisms that do not provide non-repudiation, such as (shared-key) Message Authentication Code (MAC). This is appropriate between two parties with long-term relationship, such as customer and her PSP or merchant and his PSP, and as long as the total amounts are not too high. However, this is not recommended between two parties with sporadic, ad-hoc relationship, such as customer and merchant, or when the total amounts become larger than the value of the relationship. The possible savings in computation time may become smaller than the added risks and operational costs due to dispute resolution and customer support. Proposals for micropayment systems using MAC instead of digital signatures include NetBill [CTS95] and MilliCent [M95, GM*95].

- Use public key signature algorithm that is substantially more efficient than [RSA] or [DSA]. There were several proposals of significantly more efficient public key signature schemes, e.g. [S93]. However, none of these schemes has yet gained sufficient adoption, and the amount of cryptanalysis efforts to break them were so far limited, therefore we do not recommend to use them for sensitive and high-value signatures or where it may lead to (expensive to process) disputes.

- Use an online/offline signature scheme as proposed in [EGM96]. With these schemes, the payment is signed (online) using a one-time (or limited-use) public key digital signature scheme, which is substantially more efficient than regular, unlimited use public key signature schemes such as [RSA] and [DSA]. The public key of the one-time scheme is signed in advanced (offline), using a regular digital signature scheme. This reduces the amount of computations required online (during the payment process); by using an appropriate limited-use scheme, it may also be possible to reduce the average computational load. These schemes are easily adopted for semi-offline payments where the PSP pre-authorizes the one-time signature scheme for a particular merchant and a specific maximal amount, and the consumer applies the one-time signature to authorize a specific amount (up to the maximal).

- Use one-way hash functions, which are much more efficient than public key signatures. These techniques fall into the following two categories:

  Hash chains and *Hash trees:* this technique uses digital signatures for non-repudiation, but only once for multiple purchases between the same customer and merchant. Often, all purchases must be of the same amount. The customer or PSP digitally signs one (pre-)authorized payment order for the merchant,

which the customer use to authorize multiple purchases. We can therefore use this technique for semi-offline pre-authorized payments, where the PSP pre-authorize the payment and the customer provides the final authorization. The (pre-)authorized, signed payment order includes a value y, which is the result of repeatedly applying l times a one-way hash function h to randomly chosen seed x. Namely, $y=h^{(l)}(x)$. The signature also includes a monetary value per pre-image, say c. To pay ic (e.g. i cents, when c is declared to equal one cent), the customer sends the authorized payment order together with a value x_i such that $y=h^{(i)}(x_i)$, namely y is the result of applying i times the one-way hash function h to the value x_i. As long as $i \leq l$ it is very easy for the customer to compute this since he knows x. Therefore, repeated payments of the same amounts to the same merchant require only few computations from consumer and seller. This scheme is useful when the consumer buys repeatedly from the same merchant (and usually for the same amounts). Proposals based on hash chains include PayWord [RS96], micro-iKP [HSW96] and others. Some variants use the natural extension of the hash-chain idea into a *hash-tree*, e.g. [JM96], for improved performance and flexibility.

MicroMint [RS96]: This scheme is unique in requiring the PSP to perform a `hard` cryptographic operation, i.e. an operation which is assumed to required huge computational resources, but is easy to verify. This is justified by performing many such operations together, which is substantially more efficient *per operation* than performing only a single `hard` operation or relatively few operations, as can be expected of an attacker. Rivest and Shamir [RS96] suggest as `hard` operation to find a *k-way collision* for a collision-resistant hash functions, namely values $(x_1, x_2, .. x_k)$ such that, $h(x_1)=h(x_2)= .. =h(x_k)=y$; it is easy to see that indeed if searching for collisions by exhaustive search, the overhead *per each k-way collision* is much smaller if a large number of collisions is collected together. The scheme has several variants, including identifying the buyer and possibly even the seller (e.g. by producing only strings where specific bits are the buyer / seller identity), non-repudiation (e.g. by signing a common prefix to all produced coins), and others.

- A simple technique that we can use when the consumer and merchant have long-term relationship is to digitally sign only when amounts exceed a certain threshold. As long as the total purchases remain under the threshold, the customer only authenticates the purchase approval using a key shared with the merchant.

- Probabilistic payments: this is another hybrid technique, where the merchant receives one signed payment order for a substantial maximal amount, but with additional messages defining the actual amount paid, thereby allowing the same pre-authorized payment to be used for many micropayments. However, in this case, the micropayments are not done by gradually increasing the value (as with hash chains), but by gradually raising the *probability* of payment of the maximal (total) amount. Therefore, for each micropayment the customer increases the expected value that the merchant will receive – but the amount that the merchant actually receives is always either zero or the maximal amount. See such techniques in [R97, LO98]. However, notice that this

behavior may be hard to explain to customers, which may lead to substantial customer service and education expenses. Therefore, this technique may be more appropriate for payments between organizations or payments by software agents.

Exercises

1. A company wants to offer payment services using mobile phones, for paying small amounts to vending machines (instead of paying with coins). It is critical that the additions to the vending machines are minimal and in particular not to require it to communicate for each payment. Consumers will open an account and deposit in it funds used for each payment. Propose an appropriate protocol between the vending machine, consumer and the company's payment server.
2. Consider the Payment Routing protocol for a network containing several PSPs as in the following Figure.
    1. Present an XML version for the PRT message. Include every security-related field, and explain the function of each field.
    2. Present an XML version for the PO message. Does the PO message have to contain the identities of the PSPs thru which the PO must be deposited? Explain.
3. Design (and justify) a game for two players and a bank. Both players keep accounts in the bank, which they use to pay 1$ to each other (the loser pays the winner). The game is `war module 3`, defined as follows:
    1. Each player *i* selects a number x_i from *0* to *2*
    2. Player *i* wins over player *k* if x_i is greater than x_k by *1 (mod 3)*
    3. If there is a match, no player wins
    4. Assumptions and requirements:
        1. Bank must be able to know, and prove if necessary, that all payments were done according to player's instructions.
        2. The game may be played multiple times and with multiple partners. Winners in each match must be paid, losers must be debited.

    If you use hash functions, specify which type of hash function is necessary and explain why.

References

[B99] Adam Back, "bearer = anonymous = freedom to contract", message posted at the cryptography mailing list, Feb. 1999, available online at http://www.privacy.nb.ca/cryptography/archives/cryptography/html/1999-02/0108.html.

[BGH*00] M. Bellare, J.A. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, E.Van Herrenweghen and M. Waidner. "Design, Implementation and Deployment of the iKP Secure Electronic Payment System". In Journal on Selected Areas in Communication, special issue on Network Security, April 2000, Vol. 18, No. 4, pp. 611-627. Earlier version "iKP -- a Family of

Secure Electronic Payment Protocols" published in Proceedings of the First USENIX Workshop on Electronic Commerce, New York, July 1995.

[C92] Achieving Electronic Privacy, D. Chaum, Scientific American, August 1992, pp. 96-101

[CTS95] NetBill Security and Transaction Protocol, Benjamin Cox, J. D. Tygar and Marvin Sirbu, in Proceedings of the First USENIX Workshop on Electronic Commerce, 1995.

[DSA] National Institute of Standards and Technology, "*Digital signature standard (DSS*)," Federal Information Processing Standards Publication FIPS PUB 186, U.S. Department of Commerce, May 1994.

[E02] Escrow.com, Escrow Payments – Process Overview, available online at http://www.escrow.com/solutions/escrow/process.asp, 2002.

[EGM96] S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signature. Journal of Cryptology, 9:35--67, 1996.

[GM*95] Steve Glassman, Mark Manasse, Martin Abadi, Paul Gauthier, and Patrick Sobalvarro. The Millicent Protocol for Inexpensive Electronic Commerce. In World Wide Web Journal, Fourth International World Wide Web Conference Proceedings, pages 603-618. O'Reilly, December 1995.

[GS96] The Agora Electronic Commerce Protocol, Eran Gabber and Abraham Silberschatz, presented at Second Usenix Conference on Electronic Commerce, Oakland, November 18-21, 1996.

[H95] Micro Payment Transfer Protocol (MPTP), Phillip M. Hallam-Baker, W3C Working Draft WD-mptp-951122 (22-Nov-95).

[H98] Safeguarding Digital Library Contents - Charging for Online Content, Amir Herzberg, Digital Library Magazine, January 1998, ISSN 1082-9873.

[H98a] Robert A. Hettinga, A Market Model for Digital Bearer Instrument Underwriting, manuscript, September, 1998. Available online at http://www.philodox.com/modelpaper.html.

[HM02] Amir Herzberg and Itsik Mantin, Secure Transactions with Multiple Agents, Manuscript, 2002.

[HN98] Amir Herzberg and Dalit Naor, "Surf'N'Sign: Client signatures on Web documents", IBM Systems Journal, Vol. 37 No. 1, pp. 61-71, 1998.

[HSW96] **Micro-Payments based on iKP**, Ralf Hauser, Michael Steiner and Michael Waidner, IBM Research, 12 February 1996, Research Report 2791 (\# 89269),

[HSZ00] Amir Herzberg, Eldad Shai and Ilan Zisser, `Decentralized Electronic certified payment order`, US patent application, filed July 2000.

[HY97] Amir Herzberg and Hilik Yochai, Mini-Pay : Charging per Click on the Web, Sixth WWW conf, Santa Clara, April 97.

[JM96] C. Jutla, M. Yung, "*Paytree: amortized signature for flexible micropayments*", In 2nd USENIX Workshop on electronic commerce, 1996, pp. 213--221.

[K01] Knud Böhle, "Access is King: About the Bright Future of Server-based E-payment Systems", ePSO Newsletter, No. 6, March 2001, available online at http://epso.jrc.es/newsletter.

[L98] L. Loeb. Secure Electronic Transactions: Introduction and Technical Reference. Artech House, 1998.

[L01] Oon Tik Lee, "Trust and Confidence with Escrow Payment Service to DRIVE Internet/eCommerce transactions", CommerceNet Singapore (CNSG) eSecurity and ePayment Seminar, October 2001, available online at http://www.cnsg.com.sg/archive/eSecurity%20Stratech%20011017.pdf.

[LO98] Richard J. Lipton, Rafail Ostrovsky, Micro-Payments via Efficient Coin-Flipping , Appeared in Proceedings of Second Financial Cryptography Conference, February 1998. Lecture Notes in Computer Science LNCS volume 1465, pp. 72-82.

[M95] The Millicent Protocols for Electronic Commerce, Mark S. Manasse, in First Usenix Workshop on Electronic Commerce, New York, July 11-12, 1995.

[M00] Common Markup for micropayment per-fee-links, Thierry Michel (Editor), W3C Working Draft, September 2000, online at http://www.w3.org/TR/Micropayment-Markup/.

[PO95] Scaleable, Secure Cash Payment for WWW Resources with the PayMe Protocol Set, Michael Peirce and Donal O'Mahony, Fourth WWW Conference, Boston, December 1995.

[R97] Ronald L. Rivest. "Electronic Lottery Tickets as Micropayments ", in Financial Cryptography: FC `97, Proceedings, R. Hirschfeld (ed.), Springer-Verlag, LNCS vol. 1318, pp. 307--314, 1998. http://citeseer.nj.nec.com/rivest98electronic.html

[R00] Eric Rescorla. SSL and TLS: Designing and Building Secure Systems. Addison-Wesley, 2000.

[RFC2246] T. Dierks, C. Allen, The TLS Protocol: Version 1.0, Network Working Group, Internet Engineering Task Force (IETF). Available online e.g. at http://www.ietf.org/rfc/rfc2246.txt.

[RS96] PayWord and MicroMint--Two Simple Micropayment Schemes by R.L. Rivest and A. Shamir, presented at RSA Security conference, 1996.

[RSA] PKCS #1 - RSA Cryptography Standard, RSA Laboratories, available online at http://www.rsasecurity.com/rsalabs/pkcs/pkcs-1/.

[S93] *Efficient signature schemes based on birational permutations*, Adi Shamir. In Douglas R. Stinson, editor, Proceedings of CRYPTO'93, number 773 in LNCS, pages 1-12, 1993. Available online from http://citeseer.nj.nec.com/shamir93efficient.html.

[S01] Oliver Steeley, Guaranteed Transactions: the Quest for the `Holy Grail`, ePSO Newsletter, No. 10, Nov. 2001, available online at http://epso.jrc.es/newsletter.

[SET] Secure Electronic Transactions, MasterCard and VISA, http://www.setco.org/set.html.

[SRW01] Tilo Schurer, interviewed by Ulrich Riehm and Arnd Weber, `Largest German Credit Card Issuer on Massive Reduction of Charge Backs`, ePSO Newsletter, No. 10, Nov. 2001, available online at http://epso.jrc.es/newsletter.

[TL96] Chrg-http: A Tool for Micropayments on the World Wide Web, Lei Tang and Steven Low, presented at Sixth Usenix Security Symposium, San Jose, July 22-25, 1996, pp. 123-129.