Lecture Notes for Advanced Web Security 2017
Part 1 — Electronic Payments

Martin Hell

1 Introduction

Electronic or digital payment systems can be defined as a way to pay for something without transferring physical items, such as bills or metallic coins. With that definition, electronic payments are not new, but have been around at least since 1871 when Western Union introduced money transfers using telegraphs.

There is today a plethora of ways of paying without using physical bills and coins. Covering all protocols, platforms and hardware solutions would require a course in itself. Instead, a few different aspects will be covered, highlighting the possibilities and the limitations of some different protocols.

2 Credit Card Payments

The first credit card was the Diner's Club card invented in 1950. Since then, the cards have developed and changed in several aspects. A history of credit cards and how they have evolved in response to different attacks and forgeries can be found in [1, Ch. 10]. Only very few aspects will be covered here.

The typical credit or debit card payment is done face to face with the merchant, but the number of payments done on the Internet is increasing every year. In a card transaction, there are five parties involved.

- Cardholder.
- Merchant.
- Cardholder's bank that has issued the card, also called issuer.
- Merchant's bank, also called acquiring bank or acquirer.
- The network.

2.1 Cards and Networks

There are a huge number of different cards with different conditions. The exact conditions for a card is an agreement between the issuing bank and the customer.

Still, the cards can be divided into two distinct types, debit and credit cards. A debit card (called "bankkort" in Swedish) is connected to an account in a bank and the account is debited at the time of purchase. In a credit card purchase, the merchant will get the money from the bank and the customer will later get a bill from the issuer. For credit cards, the customer pays off the bill with some amount each month. An alternative is that the customer must pay the full bill each month (called "betalkort" in Swedish). These two variants of credit cards can be combined in some way, where the customer decides how much to pay each month and the rest is postponed to the next monthly bill, often with a rather high interest. Of course, other variants are also possible as banks are not known to pass on a profitable business opportunity.

In order to facilitate the payments between buyers, merchants and banks, a network is needed. There are many types of credit and debit cards, but only a few networks. The largest worldwide networks are Visa, MasterCard, and American Express. Cards connected to Visa and MasterCard are typically provided by banks which in turn have agreements with the network companies as a membership. Visa calls its network VisaNet while MasterCard's network is called Banknet.

2.2 Processing a Transaction

The processing of a transaction is divided into two steps. The first stage is the *authorization* step and the second stage is *clearing and settlement*. No money is exchanged in the first step. The purpose is instead that the issuer authorizes the cardholder to make the purchase using the card. For a debit card the issuer checks that there is enough money in the account and for credit cards the issuer checks that the purchase limit is not exceeded. The authorization can be summarized as follows.

1. The cardholder presents the card to the merchant.
2. The merchant swipes the card in a terminal and sends the purchase amount together with the card information to the acquirer which forwards it to the network.
3. The network maps the card number to an issuing bank and forwards the authorization request to the issuer.
4. The issuer decides to either approve or decline the transaction and sends this message back to the network.
5. The network forwards the request to the acquirer which in turn forwards it to the merchant. The merchant gets a receipt of the authorized transaction.

If authorization is successful the purchase can be made. In the clearing and settlement stage, the merchant gets paid. This is typically done in one batch with several authorizations at once. The steps are given below.

6. The receipt of the transaction is sent to the acquirer, who credits the merchant's account with the purchase sum minus a transaction fee.

7. The acquirer sends the transaction to the network which pays the acquirer for the transaction.

8. The transaction is sent to the issuer which pays the network.

9. The cardholder pays the issuer, either immediately, at the end of the month or according to an agreed upon payment plan.

Transactions can be divided into two large classes, called Card Present Transaction (CP) and Card-Not-Present Transaction (CNP). The difference, as suggested by the names, is that in the first case, the customer is able to present the card physically to a merchant, e.g., in a store, or some device operated by the merchant (e.g., gas station terminals). In the second case, the merchant cannot physically examine the card, which is the case for Internet based transactions.

2.3 Card-Present Transactions - EMV cards

This section will briefly cover CP-transactions using IC cards (or chip cards). Payments using these cards follow the EMV standard, which is a global standard developed by Europay, MasterCard and Visa, hence the name. The standard is very involved and covers several different alternatives in order to be interoperable in many types of environments. For a detailed overview, the reader is referred to the specification documents [7].

An EMV transaction consists of inserting the card in a terminal. The terminal typically has a PIN pad attached to it, but it is not necessary (parking payment terminals is an example of a terminal that typically does not have a PIN pad). The terminal communicates with the chip on the card, and possibly also with the network. The transaction involves several steps, but the most important can be summarizes as

- Data authentication
- Cardholder verification
- Transaction authorization

All steps above can be done either online or offline. Online in this case means that the check is made through online communication with the issuer, while offline means that the chip on the card is responsible for the check.

2.3.1 Data Authentication

The goal of the data, or card, authentication step is to ensure that the card is valid, or actually that the data stored on the chip is genuine. This data is e.g., the card number, expiry date and the CVM list used for cardholder verification (see below). Offline data authentication can be realized in three different ways. *Static data authentication* (SDA) is the simplest method. The data is authenticated by the chip providing a digital signature on the data. The data is signed by the issuer and does not change. Provided with the signature is a certificate binding the issuer to the public verification key. The terminal verifies the signature and checks the certificate. If both are valid, the data is assumed valid. SDA has the drawback that replay attacks are possible since the authentication data send to the terminal never changes. A more secure alternative is given by *Dynamic data authentication* (DDA) in which the card has its own asymmetric key pair. For each authentication, the terminal generates a random number which is included in the signed data. Thus, the signature is different every time which makes replay attacks impossible. The third option is Combined DDA/generate application cryptogram (CDA) which combines DDA and an application cryptogram. CDA will tie the data authentication step with the transaction authorization step. Online data authentication is performed using the ARQC and ARPC messages described in the transaction authorization section.

2.3.2 Cardholder Verification

The cardholder verification step is used to verify that it is the true owner of the card that is making the purchase. There are several cardholder verification methods (CVMs), including *offline PIN*, *online PIN*, *signature* and *no CVM required*. The card stores a CVM list which consists of the supported CVMs and the requirement for each CVM. They are listed in order of preference and the terminal chooses the first in the list that matches the requirement. In online PIN CVM, the PIN is verified by the issuer while in the offline case the PIN is verified by the chip. No CVM required can be used e.g., for low value payments.

2.3.3 Transaction Authorization

In the transaction authorization step, the payment is authorized. This is most often done online according to the steps in Section 2.2, but it is also possible to do this step offline if the chip allows it. Offline authorization can be done in environments where there is no online connection to the issuer or when the transaction is of low value. In that case, the card generates a Transaction certificate (TC) cryptogram that can be used later in the settlement phase. For online authorization an Authorization Request Cryptogram (ARQC) is received from the card, which is sent to the issuer through the network. The response sent by the issuer is called Authorization Response Cryptogram (ARPC). Data in these cryptograms are symmetrically encrypted using a key shared by the issuer and the card. The ARQC message will allow the issuer to authenticate the card and the ARPC message will allow the card to authenticate the issuer.

2.3.4 Online vs. Offline

The need for offline cards are much less today than before, especially in countries with well developed communication infrastructure. Best practice suggests that all checks should be carried out online. However, online and offline are not mutually exclusive. They should be seen as functionality provided by EMV, not as alternatives that depend on online communication. It is common to use offline authentication, but still go online and do the ARQC and ARPC message exchanges. Similarly, offline PIN verification is common in some countries, but these transactions still go online. A well known attack on offline PIN verification was published in 2010 [9]. A man-in-the-middle attack was performed between a genuine card and a terminal. It was shown that it was possible to tell the terminal that the PIN was correctly entered while at the same time tell the card that a signature was used, so the PIN did not have to be verified. Offline authorization, however, is not performed if the transaction goes online.

2.4 Card-Not-Present Transactions

In an Internet card payment the merchant can not verify that the cardholder actually is in possession of the card. The merchant must process the payment without the cardholder or the card being present. These transactions are thus called card-not-present transactions. These transactions also include orders made over the phone and mail, often referred to as a MOTO (Mail Order/Telephone Order) transaction. Since signature and PIN authorization is not possible, other means are needed in order to verify that it is the real owner of the card making the purchase, or at least that the customer is in physical possession of the card. This is often done in one of the following two ways.

- The customer supplies the card security code, CVC2/CVV2/CID, written on the card. This code is never to be used in card-present transactions and thus it is assumed that only the holder of the card has access to this code. The code is a MAC based on the account number, version number and expiry date. Usage of this was initiated by Visa, but MasterCard followed soon.

- The customer supplies the billing address of the card. This is not written on the card, but is supposedly known by the true owner of the card. A merchant can also choose to only ship the purchased item to this address.

These checks do not protect against all types of attacks and forgeries, but it provides a rather easy way of avoiding a large class of them. Looking at the economical aspects, it is clear that there is more money to be made and saved by allowing card payments than what is being lost due to frauds. Using the information provided by the cardholder, the procedure is the same as in card-present transactions.

When purchasing something over the Internet, the merchant (or at least the one receiving the card details) will have enough information to make further purchases with the card. A company that in some way handles card data must follow the PCI DSS standard (Payment Card Industry Data Security Standard). This is a standard jointly developed by Visa and MasterCard. Often, at least in Sweden, a merchant uses another party, called PSP (Payment Service Provider), as an intermediate when communicating with the bank during a purchase. In that case, only the PSP, and not the merchant, must be PCI DSS compliant. This simplifies the situation for both merchants and banks since the banks only have to communicate securely with a few known PSPs. Of course, it also adds one more party to those that want to make money on Internet payments. For simplicity, the PSP is included in the "merchant" in these notes.

3 Increasing Security in Web Based Card Payments

Card-not-present transactions are increasing due to the increasing number of purchases done on the web. This section will present two significant efforts to increase the security in CNP transactions over the Internet, namely SET and 3D Secure.

3.1 Secure Electronic Transaction (SET)

To increase the security of Internet payments, the Secure Electronic Transaction (SET) protocol was proposed by SETco in 1997. SETco consisted of several large companies and was led by Visa and MasterCard. SET provided application level confidentiality and integrity protection of data. It also offered one more important security feature, namely that the Merchant does not get access to the card information. Thus, it significantly limited the risk of card informations ending up in the wrong hands.

There are three parties participating in the protocol.

- Cardholder. As before this is the party making the online purchase.
- Merchant. This is the party selling an item through a web store.
- Payment gateway. This party will take care of the actual transaction, contacting the issuer's bank for payment authorizations.

In SET, the information necessary to complete a purchase was divided into two parts, the *order information* (OI) and the *payment information* (PI). The order information consisted only of information needed by the Merchant to complete the purchase while the payment information only included information, e.g., card details, that was needed by the Payment Gateway.

All details are prepared by the customer and the OI and PI are separated using a *dual signature*. The dual signature is constructed by first hashing (creating message digests of) OI and PI, resulting in OIMD and PIMD. Then these two digests were concatenated, hashed again, and then signed by the customer.

In order to participate in a SET transaction, all involved parties must have an asymmetric key pair with a corresponding certificate, signed by a CA. The CA for customer and merchants could e.g., the issuer and the acquirer respectively, with certificates signed by e.g., Visa or MasterCard.

The steps in a transaction can be summarized as follows.

- In an **initiate request** message, the Cardholder requests the Merchant's and the Payment Gateway's certificates from the Merchant.
- The Merchant returns the requested certificates together with a Transaction ID in an **initiate response** message. Certificates are verified, OI and PI are prepared and the dual signature is created. The PI is symmetrically encrypted and the key is encrypted using the Payment Gateway's public key.
- The Cardholder sends its own certificate, dual signature, encrypted PI, PIMD and OI to the merchant in a **purchase request** message.
- The merchant verifies the dual signature using PIMD and OI. The merchant hashes the OI and sends an authorization request to the Payment Gateway. This message includes the encrypted PI, dual signature, OIMD, Transaction ID and the Cardholder's and Merchant's certificates. This is signed by the Merchant and symmetrically encrypted. The encryption key is encrypted with the Payment Gateway's public key.
- The Payment Gateway contacts the issuer through the network and requests payment authorization in the same way as if SET is not used. If successful, an **authorization response** is returned to the Merchant, symmetrically encrypted, with key encrypted with the Merchant's public key.
- The protocol is ended with **capture request** and **capture response** messages between Merchant and Gateway, finalizing the actual payment and the Merchant getting the money deposited in his bank account.

Though SET introduced a much higher level of security, providing confidentiality, authentication, integrity and non-repudiation on message level, and in particular by not giving the credit card information to the Merchant, it was never really embraced by neither merchants nor customers. There were several reasons for this. One important reason was that it introduced security at the expense of convenience. Instead of just submitting card information through an HTML form, customers now needed to get a certificate and be enrolled in a PKI. In addition, dedicated software had to be installed on the customer's computer which would create interoperability problems between different software implementations. It also introduced additional costs for merchants in order to deploy the extra functionality in their existing systems.

3.2 3D Secure

Frauds related to card-not-present transactions increased by more than 100% between 2004 and 2008 in the UK[1] [2]. One solution to increase the security in these types of transactions is denoted 3D Secure and was proposed by Visa.

[1]I have not been able to find any figures for Sweden, but such information would be much appreciated.

Similar to many other credit card payment technologies the deployment has been rather slow. One reason is that both the bank, the merchant and the customers need to adapt and embrace the solution before it can work. The "3D" in the name refers to the three domains that are active in the protocol, the issuing domain, the acquiring domain, and the interoperability domain.

3D Secure is a framework that allows an issuer to authenticate a cardholder, in addition to the controls made by the merchant. The issuing bank can authenticate the cardholder using any method it finds appropriate, and send the result of the authentication to the merchant. This has the important effect that the merchant will not be liable for fraud resulting from unauthorized use of cards. The responsibility is instead shifted to the issuing bank. Even if the cardholder or issuer is not part of the program, it is enough that the merchant supports 3D Secure in order to avoid liability.

3D Secure can be thought of as a PIN code, i.e., the cardholder presents a secret that is known to only the issuing bank and the cardholder. It can be an actual PIN in the simplest case, but it can also be a more complex password or some multi-factor authentication possibly involving actually proving that the cardholder has physical access to the card. The complexity of the authentication is often a tradeoff between security and user friendliness.

3D Secure has been implemented by the networks under different names. Visa calls it *Verified by Visa*, MasterCard calls it *MasterCard SecureCode* and American Express calls it *American Express SafeKey.* The transaction flows given in this section are those specified by Verified by Visa, abbreviated VbV. For more details, the reader is referred to [18].

3.2.1 Enrollment

Before a purchase can use VbV, the cardholder must activate the card. In the activation process, the issuer verifies that the cardholder is the actual owner of the card. Moreover, the cardholder can choose a Personal Assurance Message, which is a message that will always be displayed to the cardholder in each purchase. This allows the cardholder to authenticate the issuer at the time of purchase, i.e., to verify that it is sending the authentication messages to the bank and not to e.g., a phishing site.

The enrollment should be done in a separate registration process. Typically, the cardholder logs into the issuer (bank), and activates the card. It can also be done in a face-to-face meeting with the issuer or in some other way that allows the issuer to authenticate the cardholder.

To simplify the enrollment, a method known as Activation During Shopping (ADS) is also specified. Then, the enrollment is done as part of the first purchase made using VbV. The cardholder answers a series of security questions in order to authenticate and then also chooses a password and the Personal Assurance Message.

3.2.2 Making a Purchase

When the cardholder is authenticated, several different services are involved. Inside the network (the interoperability domain) there are different servers to handle different functions. All these are grouped into the network in this overview. On the issuer's side the most important part is the Access Control Server (ACS). This server authenticates the cardholder and provides digitally signed messages. A bank can outsource this functionality to a third party. On the merchant's side the most important part is the Merchant Server Plug-In (MPI). This is a software module that, e.g., handles the communication with the network and verifies the ACS's digital signatures.

When a purchase is made, there are two main phases in the 3D Secure protocol.

1. Verify enrollment
2. Cardholder authentication

In the first step, the MPI checks whether the card is enrolled in the VbV program. The second step is the actual authentication of the cardholder performed by the issuer.

Assuming that the card has been enrolled, the information flow in the verify enrollment phase is summarized below.

1. The cardholder submits the credit card details to the merchant, similar to what is done in an Internet purchase without 3D Secure.
2. In a verification request (VEReq), the MPI sends the card information to the network, which checks if the card is enrolled. The network also authenticates the merchant and forwards the request to the ACS.
3. Assuming that the cardholder is enrolled and can use VbV, the ACS sends a verification response (VERes) to the network with the URL to use for the Payer Authentication Request (PAReq). The network forwards the VERes to the MPI.

If the card is enrolled, the cardholder authentication phase follows.

1. The MPI sends the PAReq to the ACS via the cardholder's device, e.g., web browser. The location of the ACS was received in the verify enrollment phase. The ACS sends an authentication request page to the browser via the network. This is typically opened in an iFrame in the browser.
2. The cardholder enters the password or authenticates in some other way to the issuer's ACS. If the authentication is successful, the ACS creates a Payer Authentication Response (PARes) which includes the result of the authentication. This response is digitally signed by the issuer.
3. The PARes is sent to the cardholder's browser via the network. The cardholder forwards the PARes to the MPI, which verifies the signature.

4. The network saves the response so that future disputes can be handled.

If authentication is successful, the merchant can continue with the purchase as usual, by sending authorization requests to the acquirer.

3.2.3 Drawbacks

In [10] 3D Secure is heavily criticized and several problems are pointed out. Some of them are related to the protocol itself, but some are more related to the implementation used by issuers.

The fact that the issuer shows up in a pop-up window or in an iFrame on the merchant's webpage presents a problem in itself. It is not at all obvious for people not familiar with the protocol to understand that they are now communicating with the issuer, and not the merchant. In fact, when people see a pop-up window and are asked to enter their bank passwords, intuition says that this is a phishing attempt, i.e., some attacker tries to steal your bank credentials. Moreover, when looking at the domain it might not be the issuer domain that you are actually communicating with, but instead a third-party domain acting on behalf of the issuer through outsourcing. In the current specification, pop-ups are not allowed and iFrames must be used.

The main reason to use ADS is to simplify the enrollment process and to increase the adoption rate. Thus, this is solely a business choice made to attract more people to the product. As a consequence, security is compromised (which in turn can backfire and people will not use it for that reason). First, in the middle of a purchase people are less likely to choose good passwords since their focus is on the purchase. Second, as pointed out above, sending sensitive information to a webpage that is difficult to authenticate is something people have been told over and over again not to do.

4 Untraceable E-cash

The main property of untraceable e-cash is that it is not possible to link withdrawals and deposits. With credit card transactions, the bank knows the identity of both the buyer and the merchant. Untraceable e-cash is in that sense more similar to physical bills. It is possible to withdraw, e.g., 100 SEK from an ATM and buy something. When the merchant deposits the bill the bank has no way of tracing that it was actually Alice's bill and that she has made a purchase. There are serial numbers on the bills so in theory it would be possible to trace them, but it is not practical for the bank to do so. Even if it did, it would have no way of knowing how many times the bill was used, and which merchants and people used it, between a withdrawal and a deposit. Still, the e-cash is an electronic currency so the bank must have some control over it. First, it should not be possible to create electronic coins without involvement of the bank. This is rather easy using digital signatures. Second, and most importantly, it should not be possible to copy or double spend a coin. With bills, this is accomplished by making them very difficult to counterfeit, adding e.g., embedded strips.

Electronic cash was first introduced in [6]. Since then, several e-cash systems have been proposed. Still, most build upon the original ideas in [6]. Subsequent improvements make some things a little bit differently, improving some aspects of the security and the efficiency of the protocol. These notes will focus on the main ideas that basically all e-cash systems build upon, using the scheme in [6]. This scheme was used in the DigiCash electronic payment system introduced in the 90's. It received a large amount of publicity, but never managed to become widespread enough for a real breakthrough. Bankruptcy was declared in 1998.

There are three parties involved in the process, namely a *bank*, a *payer* and a *merchant*. In the following, Alice will be the payer and Bob will be the merchant. The bank issues the electronic cash to Alice and debits her bank account. Then Alice transfers the e-cash to Bob who in turn deposits the money back into the bank. This is very similar to credit and debit card transactions but the important difference is the anonymity introduced in the protocol. When the bank receives the e-cash from a merchant, it can not trace this e-cash back to the payer. This is achieved using *blind signatures*.

4.1 Blind Signatures

A blind signature is a signature where the signer signs something but he does not know what he signs. This can quite easily be achieved using the multiplicative property of RSA. Let n be the public modulus, d be the private signing key and e be the public verification key. Recall that an RSA signature on x is computed as $s = h(x)^d \bmod n$ and is verified by finding $h(x)' = s'^e \bmod n$ and checking that $h(x) = h(x)'$. In the blind signature, Alice wants the bank to sign an electronic coin, but she does not want it to be traceable back to Alice once Bob deposits the same coin. This can be accomplished by letting the bank sign $r^e \cdot h(x)$, where r is a random value. Then, Alice will get the signature $s = r \cdot h(x)^d$. By multiplying this with the inverse of r, Alice has a valid signature on x, i.e., $h(x)^d$. Since r is randomly chosen, the bank will get no information about the value of $h(x)$. Still, $h(x)^d$ is a valid signature on x.

In the following, the public verification key will be fixed to $e = 3$. Thus, d is the inverse of 3 in the ring modulo $\phi(n)$, where $\phi(\cdot)$ is Euler's phi function. The inverse is simply written as $d = 1/3$ (however, note that it is actually an integer).

4.2 Protocol

Based on the blind signature scheme above, a first simple scheme for untraceable e-cash can be formulated as follows:

1. Alice generates two random numbers x (the coin) and r (used for blinding), computes $B = r^3 h(x) \bmod n$, and sends B to the bank.

2. The bank signs B resulting in the signature $rh(x)^{1/3}$. This value is returned to Alice. At the same time, Alice's account is debited with one currency unit. The *withdrawal* of a coin has been completed.

3. Alice uses the inverse of r to compute $h(x)^{1/3}$.

4. Alice gives the pair $(x, h(x)^{1/3})$ to Bob. Bob can verify that this is a valid coin by verifying the signature.

5. Bob contacts the bank to make sure that x has not been deposited before, and then deposits x by sending $(x, h(x)^{1/3})$ to the bank. The bank credits Bob's account with one currency unit.

One coin has now been transferred from Alice's account to Bob's account. In the last step the bank can verify that it has actually signed x some time before, so it is a valid coin. However, since the bank has not seen $h(x)$ before (it was blinded), it can not trace it to a particular instance of a withdrawal action. The last step of the protocol is crucial in order to verify that Alice has not used this coin before with some other merchant. However, it is not very practical to contact the bank for every single transaction. Moreover, if it has been sent, the bank has no way of knowing who double spent the coin since it can not relate it to an individual. What is needed is a way for Alice to *stay anonymous*, and at the same time provide the following:

- Bob does not have to contact the bank for every transaction in order to verify that the coin has not already been spent.

- If and only if Alice double spends the coin, the bank can find the identity of Alice.

By solving the second problem, the first is automatically solved. By making sure that Alice is identified when she double spends a coin, Bob does not have to check if it is spent or not. A fraud will reveal Alice anyway and actions can be taken to reimburse Bob for his possible loss. The process of withdrawing one coin from the bank is given below.

1. Alice chooses $2k$ quadruples of random numbers (mod n) a_i, c_i, d_i, r_i. These will result in one coin. The coin is associated with an identifier, ID, which links Alice to this coin and this specific transaction. Alice computes $2k$ values
$$B_i = r_i^3 f(x_i, y_i) \bmod n$$
where
$$x_i = h(a_i, c_i) \qquad \text{and} \qquad y_i = h(a_i \oplus ID, d_i).$$
Note that $h(\cdot, \cdot)$ is a hash function with two inputs, but it has the same properties as normal hash functions. All $2k$ B_i values are sent to the bank.

2. The bank randomly picks k indices out of the $2k$ that were sent, and sends these indices to Alice. Denote this set by R, i.e., $R = \{i_1, \ldots, i_k\}$ where $1 \leq i_j \leq 2k$.

3. For each $i \in R$, Alice reveals a_i, c_i, d_i, r_i to the bank, which checks that they can be used to compute $B_i, i \in R$, knowing ID.

4. Using the remaining B_i, the bank computes the blind signature

$$\prod_{i \notin R} B_i^{1/3} \bmod n$$

and sends this to Alice. Since Alice knows r_i she can extract the signature

$$S = \prod_{i \notin R} f(x_i, y_i)^{1/3},$$

which is also regarded as the serial number of the coin.

The purpose of steps 2 and 3 above is to allow the bank to verify that Alice has properly included her identifying information in all B_i. This is needed to make sure that Alice can be identified in case she double spends the coin. If this information is properly encoded into k randomly chosen B_i, then there is a high probability that it is properly encoded also in the remaining k B_i. This technique is called *cut-and-choose.*

When Alice buys something from Bob the steps are as follows.

1. Alice sends S to Bob. Recall that S is computed using k quadruples $a_i, a_i \oplus ID, c_i, d_i$.

2. Now Bob needs to verify the signature. He randomly generates a binary vector of length k, $z = (z_1, \ldots, z_k)$, and sends this to Alice.

3. Let the indices $i \notin R$, which were used to create the signature, be numbered from 1 to k. For each $j, 1 \leq j \leq k$, Alice does the following:
   If $z_j = 0$, then Alice sends $x_j, a_j \oplus ID, d_j$ to Bob.
   If $z_j = 1$, then Alice sends y_j, a_j, c_j to Bob.
   This will allow Bob to compute $f(x_j, y_j)\ \forall j$, and verify the signature. Still, Bob can not compute ID since he does not have both a_j and $a_j \oplus ID$ for any j.

4. Bob can at any time send S, z and Alice's response from the previous step to the bank. The bank verifies the signature and credits Bob's account.

Since the bank receives the vector z and Alice's response it can verify the signature. For each index j, the bank now has either a_j or $a_j \oplus ID$, but not both. Thus, the bank knows that it signed the coin sometime before but it can still not compute ID and trace the coin to Alice. The idea now is that, if Alice decides to double spend the coin, then another merchant (Charlie) will also construct a random vector z' and Alice has to perform step 3 above for that vector. If $z \neq z'$, i.e., there is an index j such that $z_j \neq z'_j$, then Alice will have revealed both a_j and $a_j \oplus ID$, one value to Bob and one value to Charlie. When Charlie deposits the coin, which he thinks is valid since the signature is ok, the bank will notice that S has been deposited before. The bank can then compute $a_j \oplus a_j \oplus ID = ID$, identify Alice, and take proper action. If the two vectors are equal, then Alice can not be identified. Instead the bank will assume that Bob and Charlie have cooperated and that they are the cheaters.

4.3 Improvements

The protocol given in the previous section was first given in [6] in 1988. Since then, many different protocols have been proposed, improving different aspects of security and efficiency. The details of many subsequent protocols require a mathematical background which is not required in this course, but most protocols follow the same basic principles as the one in [6].

One additional feature is that Alice can add a signature to the *ID*. This can be done by using the *ID* together with an index, such that the value is different for each B_i, and to digitally sign this value with Alice's private key. Then it is not possible for a malicious Bank to frame Alice when she is actually innocent. Only Alice could have produced the values that include the *ID*. In general, any information together with the *ID* that can only have been created by Alice will have the same effect.

The cut-and-choose technique allows Alice to prove (probabilistically) that she has correctly encoded the *ID* in the values B_i. Unfortunately, she has to reveal information about half of the computed values by doing so. This has later been improved by using other types of protocols, e.g., zero-knowledge proofs. This allows Alice to prove that she knows some secret without revealing any information about the secret. A very simple introduction to zero-knowledge proofs can be found in [14].

Subsequent e-cash schemes typically uses secret sharing techniques instead of plain xor in order to identify Alice in case she double spends the coins. As an example, assume that Alice's identity is a point, e.g., $f(0)$, on a straight line $f(x) = a_0 + a_1 x$. Knowing one point on the line does not reveal any information about $f(0)$ (unless the point is $f(0)$), but knowing two points on the line allows the bank to find $f(0)$ easily.

Today the most efficient e-cash scheme is probably the one denoted compact e-cash [5].

5 Micropayments

A micropayment is defined as a very small payment. There is no consensus on how small a payment has to be in order to be regarded as a micropayment. A credit card transaction comes with a fee taken by the banks. This fee is typically in the order of 2-3 SEK. For small payments, this fee is non-negligible, and the merchant might actually lose money in a transaction due to this fee. This is the motivation to study micropayments and to come up with feasible solutions that allow customers and merchants to exchange money without the banks taking most of the money in the transaction. With this background, a micropayment can be defined as a payment where the credit card transaction fees are, or would be, a *substantial* part of the total transaction. Some micropayments can be less than 1 SEK, but one can also argue that anything less than 100 SEK is a micropayment.

The e-cash solutions discussed in Section 4 does not use credit card transactions, at least not in their description. In an actual implementation, credit cards could of course be involved. Those e-cash proposals could thus also be used for micropayments. However, this is not their main motivation as they primarily focus on anonymity and to solve the problems introduced when providing the anonymity. This section will focus on techniques where the main focus is to transfer very small amounts from customers to merchants, but the reader should not draw a too hard line between the two sections.

Similar to the e-cash solutions, there have been very many proposed solutions for micropayments. The main idea that all these systems has in common is *aggregation*, i.e., to convert many micropayments into a small number of *macropayments.* A macropayment is a transaction of a large amount (larger than the micropayment) that involves the transaction fees taken by the bank. For these payments, the fee can be seen as a very small part of the cost. As a comparison, the untraceable e-cash technique by Chaum does not have any aggregation. Each transaction is handled independently from other transactions.

Following [16], three different types of aggregations will be discussed below, namely *session-level aggregation*, *aggregation by intermediation* and *universal aggregation.* It should be noted though, that there are also other ways to categorize micropayment solutions, see e.g., [13].

5.1 Session-Level Aggregation

In session-level aggregation, the user makes several micropayments to one merchant during a short period of time, e.g., one day. At the end of the day, all payments are collected and one macropayment transaction for the total sum is made. One simple, but elegant solution to accomplish this is the PayWord scheme given in [17]. In order to make sure that the correct sum is debited the amount, and not more, a hash chain is used. The idea is that it is very easy to go forwards in the chain, by computing a hash, but it is infeasible to go backwards as this would correspond to finding a pre-image of the hash. The parties involved in the process is the customer (or user), the merchant and a bank. The public and private keys are denoted PUB and PRI respectively, with subscript U, M and B for the different parties. A signature on the data $\{\cdot\}$ is given by $\{\cdot\}_{PRI}$ and it is assumed that the data in cleartext is always submitted together with the signature and that the signature is computed on a hash of the data. The process is as follows, see Figure 1.

1. The user establishes an account with the bank, and the bank issues and signs a certificate C connecting the user's public key PUB_U with the user. The certificate also contains the name B of the bank, showing that the user has an account with the bank, i.e., $C = \{B, U, PUB_U\}_{PRI_B}$.

2. When the user wants to make purchases from a new merchant, the user computes a hash chain of n hashes, starting from w_n.

$$w_i = h(w_{i+1}), \qquad i = n \dots 0$$

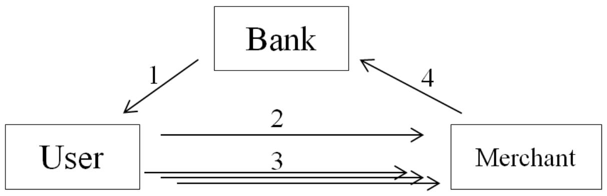

Figure 1: An overview of the PayWord protocol.

The user connects the last value in the hash chain with the merchant by computing $S = \{M, w_0, C\}_{PRI_U}$ and sends this to the merchant. The merchant checks the certificate and verifies that the user can make purchases using money in the bank B. S is called a commitment. For each new merchant that is contacted, a new commitment is computed using a new hash chain.

3. When the user buys something from the merchant he sends a PayWord $P = (w_i, i)$. For each new item, or currency unit, i is incremented, i.e., first he sends w_1, then w_2 and so on. By hashing w_i i times, the merchant can verify that the payment matches the commitment. No one else but the user can produce w_i from w_j ($j < i$). Note that the commitment S should typically be sent together with the first PayWord.

4. At the end of the day, or when enough PayWords have been sent, the merchant contacts the bank, providing the commitment S and the PayWord (w_i, i) with the largest i received from the user. The bank verifies the PayWord by hashing w_i i times and checks that it matches w_0 in the commitment. The bank credits the merchant's account with i units and debits the user's account (or sends a bill, whatever is appropriate).

This scheme assumes that the user is buying several items from the merchant during a relatively short time period. An example purchase used in [17] is that the user pays for the webpages that are accessed. Each requested webpage, e.g., a news article or an encyclopedia entry, results in a new PayWord. Other examples can be that users pay per minute of video in a video on demand, or per minute or per song in a music streaming service.

5.2 Universal Aggregation

The main problem with session-level aggregation is that each macropayment is a collection of micropayments by one user to one merchant. If users make many small payments, but to different merchants, there is no gain. It would be better if a merchant could aggregate the micropayments from different users and get paid from all using one macropayment. Similarly, a user should be able to make micropayments to many different merchants but collect all in one macropayment. This can be achieved by using *probabilistic* payments. This means that instead of making a micropayment for μ SEK, he makes a macropayment for γ SEK with probability μ/γ. With probability $1 - \mu/\gamma$, he pays nothing. As an example, let each micropayment be $\mu = 1$ SEK and the macropayments be $\gamma = 100$ SEK. Then with probability 1/100 the customer pays 100 SEK and with probabilty 99/100 he pays nothing. Thus, on average the customer will always pay 1 SEK.

One way to accomplish this is to let the merchant also produce a hash chain and tie the two chains together in the commitment. If the corresponding hash value from the merchant equals that of the customer, then a macropayent is made. This was proposed in [15] and denoted *electronic lottery tickets.* A more elegant solution was given in [8] avoiding interaction in the payment decision between the customer and the merchant. Another problem that was solved is that users may not be comfortable with the risk of being charged with several macropayments, while only purchasing a few items. This can of course happen by chance although it will always even out in the long run. The solution in [8] is to guarantee that the user never pays more than he has spent. Instead the psychological disadvantage of the variance in payments is moved to the banks. This scheme is the basis for Peppercoin micropayment scheme and can be summarized as follows.

Let F be a function that maps a binary string to a real number between 0 and 1. The selection rate, i.e., the proportion of purchases that results in a macropayment is given by $s = \mu/\gamma$. All information about a purchase is represented by T. A serial number that is incremented for each purchase is denoted by S. Then, the protocol is as follows.

1. Upon making a purchase, Alice sends $\{T, S\}_{PRI_A}$ to the merchant.
2. The purchase results in a macropayment of γ SEK if
$$F(\{\{T, S\}_{PRI_A}\}_{PRI_M}) \leq s.$$
3. If a macropayment should be made, then this value together with $\{T, S\}_{PRI_A}$ is sent by the merchant to the bank. The bank verifies the signatures and looks at the serial number. The bank keeps a record of the highest serial number that has been charged previously, S_{max}. Alice's account is debited with
$$\mu(S - S_{max})$$
SEK and the merchants account is credited with γ SEK. The recorded value of S_{max} is updated as $S_{max} \leftarrow S$.

If Alice tries to cheat by using the same serial number twice, but with different merchants, then this can be revealed if both results in a macropayment. If the fine is larger than the expected gain, then users will not attempt this fraud.

Peppercoin is based on the principles in the above scheme, though there are some differences in order to make it more practical. The peppercoin system takes the role of the bank and also charges its own fees for each transaction.

Even if the problems in session-level aggregation are solved with universal aggregation, the systems still depend on users making very frequent micropayments.

5.3 Aggregation by Intermediation

In aggregation by intermediation, a third party is used to keep track of all micropayments paid by a user and all that are received by a merchant. This third party acts like a bank for all the micropayments. When enough payments have been made/received then these are aggregated in one large payment using the regular banks. This requires one more party to be involved and that has to make money on the transactions.

This can be solved by letting users and merchants have an account with the third party. Users can deposit a macropayment sum into an account with the third party and then for each micropayment, a small amount is debited this account and credited the merchant's account. These systems do not involve much interesting technology and are left here without further investigation.

5.4 Drawbacks

In [12], it is argued that micropayments have several difficult issues to overcome in order to be widely used by customers. These issues are not technological, but rather psychological and related to economical models. For merchants, it is more profitable to sell bundles instead of selling small individual pieces. One example is Microsoft Office which is a bundle of several programs which is much cheaper than the sum of the individual program prices.

Another issue is that research has shown that customers are more willing to pay more for a flat rate than what would be the sum of the individual payments. This is a sort of insurance for the users and it gives them more control over their own economy.

6 Bitcoin

Everything discussed above relate to digitalization of already existing money. The bank handles the money electronically, but the consequences for the bank accounts are the same as if regular bills are used. The sum is debited one account and credited another. Bitcoin is different from this in that it is a completely new currency with new money being "printed" within the system. Banks and governments have no control of the system and are not even needed.

Bitcoin was proposed by Satoshi Nakamoto (the name is believed to be a pseudonym) in [11]. It is a decentralized currency system that does not require any trusted third party or bank to work. It is based on peer-to-peer technology, where all nodes always have access to all information about the system and all transactions ever made. For security, it requires that the majority of all participants (actually, computing power), is honest and follow the rules. The fundamental ideas behind Bitcoin are very simple, though there are several advanced concepts needed in order to make it work in practice. This section will give the basic functionality of Bitcoin. The original paper [11] also provides a good introduction to the different concepts. The reader is encouraged to consult that description. The protocol specification, or parts of it, can be consulted if necessary [4].

6.1 Transaction

Informally, transferring money from one user to another is done by digitally signing a "document" where the sender says that a certain amount of coins now belongs to the receiver. This is called a transaction and the money are called Bitcoins (BTC). The sender and receiver are identified by public keys, or hashes of these keys. In a transaction, Bitcoins are typically sent to a Bitcoin address, though there are other alternatives. A Bitcoin address is basically just an encoded hash of a public key. Thus, the system is anonymous since the money is owned by a public key and there is not necessarily a known relation between users and public keys. Only the holder of the corresponding private key can use the Bitcoins and transfer them to another public key. Transactions are tied together by letting the input of a transaction refer to an output of a previous transaction. In this way, it is not possible to just produce your own money and sign it over to someone else. The money can be traced back to the root from which they originally came.

Below is a (simplified) example of a transaction.

```
1  hash: 26a6230f29715cfbb19b465...90be3195b837f667b2c6a46ac6adee
2    in:
3      prev_out:
4        hash: 3e5969d6314cdf5b8...edad50c1eaea3ae7bc94cb44479c82
5        n: 1
6      scriptSig: 3045022100cd85...18cef0b2360f51aa43ca2b02218601
7                 04e744fd5b041b...a6b888082c839368e510134a3251ab
8    out:
9      value: 15.00000000,
10     scriptPubKey: fa532de64071fb72198d17fc4ebdc0210d063110
11
12     value: 0.07450000,
13     scriptPubKey: 07c017250f85b2590a31730c4908009fd83dcc62
```

Line 1 gives the hash of the transaction. This is the hash to be signed by the sender. Then follows two main sections, the input and the output sections (`in` and `out`). The input always refers to an output in a previous transaction. There can be several inputs but they must all be controlled by the same user. They often correspond to different public and private keys, but always to the same user. There can also be several outputs. Looking at the example, there is one input referring to an output in the transaction with hash `3e5969...` (line 4).

The specific output that is referred to is the second output in that transaction (line 5). Note that the index starts at zero. Then follows first the signature (line 6) and the public key of the sender (line 7). There are two outputs in the transaction. This is very common since then one output can be the money actually sent to the receiver, while the other output is the change that is sent back to the sender. Recall that the input always uses a previous output so the value on the input can not be chosen to any value. Thus there is usually some change. The case is similar to real money. If something costs 29 SEK and you have one 100 SEK bill, you have to give away the full bill (100 SEK as input) and then there is 71 left as change. The difference is that you do not get change from the receiver, you split the 100 SEK bill in two parts instead. The part that is kept is then equivalent to a 71 SEK bill. Returning to Bitcoin, by adding more outputs it is possible to transfer money to several receivers at the same time. The output corresponding to a specific receiver will be referenced by him (`hash` and `n`) when he transfers it to the next address.

Thus, the signature (line 6) in the transaction can be verified using the public key of the sender (line 7). Tracing this backwards to the transaction `3e5969...`, it can be verified that the hash of this public key corresponds to the address given in the corresponding line 13 in that transaction. When the receiver with address `fa532d` decides to use the 15 BTC received, he will construct a new transaction that refers to the first output (`n: 0`) in the transaction above and provide his real public key in the corresponding line 7 in the new transaction.

6.2 Blocks and Block Chains

Since there are no third parties or banks, the transactions can take place on any computer anywhere. Transactions are published to all nodes in the Bitcoin network so all participating computers will know about the transaction. While this provides a way to let Bitcoin money change owner it does not prevent double spending. Alice can construct a transaction that transfers money to Bob, and then later construct a new transaction that transfers the same money to Charlie. To prevent this, without involving a bank, a transaction is put into a *block*, which in turn is part of a *block chain*. The block chain is known to all participants and all blocks in the block chain represents transactions that are valid and accepted by the network nodes. In order to add a new block to the block chain, a concept known as *proof of work* is used. A simplified example showing the most important parts in a block is given below.

```
1 hash: 000000000000055f4f1...3a8a001c3307e0e6cbea474798a223e9e50,
2 prev_block: 0000000000000a4d5d7...527ceb9f8f52f86aeeef8e3b52464,
3 mrkl_root: 18353cf8f8f4bfa2ecff...923b40871a016d1793f1a946a1201,
4 nonce: 97560167,
5 tx: ...
6     ...
```

The hash (line 1) is the hash of the block. This starts with a certain number of zeros, showing that a significant amount of work has been put into computing the hash. The block is repeatedly hashed, each time incrementing the nonce (line 4). When a valid hash is found, i.e., one that starts with a certain number of zeros, the block is sent to all nodes in the network and added to the block chain. The chain is constructed by including the hash of the previous block (line 2) in the current block. Transaction that have been published since the last block was created are included in the block, making these transactions valid. When nodes receive a new block with a valid hash, they check the transactions and make sure that there is no double spending. If the block only consists of valid transactions, the block is accepted and the nodes start computing the next block.

The proof of work provided by requiring a certain number of zeros is the same idea as that used in hashcash. Actually, Bitcoin generalizes the concept and determines a specific number which the hash must be lower than. This allows a more detailed control over how long time it will take to produce a valid hash. The number is chosen such that new blocks are created every 10 minutes. The actual average over the last 14 days is used to update the number so even if many more computers enter the network and start computing hashes, the difficulty is adapted and it will still take on average 10 minutes to find a valid hash. The computer that finds a valid hash is rewarded. Currently the reward is 25 BTC, but it will be halved for each 210000 blocks generated. This is how new money is constructed in the system, and it is the only way for new money to enter the system. This also automatically creates a root transaction for the new money and it provides an incentive to provide ones computing power to the Bitcoin network. For this reason, computers that are used to compute hashes are called Bitcoin miners. Because of the reward halving, the total number of Bitcoins created will be about 21 million. The reward is called a *coinbase* transaction and has no inputs. It is defined to be the first transaction in a block and is created by the miner using an arbitrary address (corresponding to the miner's public key) as output.

Since all transactions are published to all nodes, a miner can choose to include or not to include it when trying to compute valid blocks. In order to speed up the process of including a transaction in the block chain, a *transaction fee* can be used. This fee is the difference between the sum of the input values and the output values. Since this difference has no specified receiver it can be claimed by the miner. The transaction fees are also included in the coinbase transaction and give a miner a reason to include the transaction in a block. Thus, having a transaction fee will make the transaction accepted faster. If there is no fee, the transaction will eventually also be included in a block, but it should take more time since just a subset of miners include it in the block. Transaction fees are expected to be the mining incentive once the rewards will not be used any more, i.e., when the total number of Bitcoins reach about 21 Million. This will happen around year 2140.

Several computers may compute a valid hash around the same time and broadcast it to the network. This means that the block chain will temporarily fork into two chains. This fork may continue for a while and the problem is solved by defining the chain that represent the most amount of work to be the valid chain. Thus, it is possible for a transaction to be included in a block that is later abandoned in favor of another block. For this reason, it is recommended to wait for a depth of approximately 6 blocks (one hour) before being certain that the transaction will stay valid.

An attacker can attempt to change the history of his Bitcoin transactions by e.g., modifying a transaction such that less money is sent or money is sent to another address. This will require the attacker to produce a new valid block with the modified transaction and then create valid hashes for all subsequent blocks in the block chain such that the chain fork produced will be longer than that one used by the honest nodes in the network. For this to happen, the attacker must have computational power that exceeds the total computational power of the honest nodes. Otherwise, the modified chain will never be able to catch up with the real chain, unless it is very lucky, which motivates the recommended delay of 6 blocks before accepting that a transaction can not be changed. Note that even though the attacker has a huge amount of computing power, it is not possible to make arbitrary changes in transaction. The attacker would only be able to change transaction belonging to him, i.e., transaction where his coins are used as input. Other transactions are still secure since it is assumed that the attacker does not have the private key needed to sign other transactions. He could try to change a block such that he receives the award, but then he could just use his computing power to compute new blocks, which will also give him rewards.

6.3 Problems

Bitcoin has several very appealing features in that it allows users to anonymously and securely transfer money without having to trust a third party. Even though the security is rather robust as it depends on well known cryptographic principles, there are other issues that can be seen as drawbacks. Some drawbacks are listed below.

- The fact that only 21 million BTC will ever enter the system will cause deflation. It is unclear what will be the exact consequences, but if the currency keeps increasing in value, people will not use it. Why spend it if it has more value the next day? If money is not spent, there is no transaction fee, removing the incentive to create new blocks and to participate in the network at all.

- If the private key corresponding to an output address is lost, the money can not be used as input to a new transaction. This means that the money exist in some sense but can never be used. It is known that there are already a large amount of Bitcoins that can never by used since the private key is lost.

- Since all transactions are anonymous, Bitcoins can be used as black money, paying salaries while avoiding taxes.

A more detailed overview of drawbacks and discussions on how to overcome them is given in [3].

If it turns out that many merchants start accepting Bitcoins, the currency will be considered useful and it will have a real value. In particular, those that are early adopters will be able to collect quite many Bitcoins by being miners. As more people enter the network, it will be much more difficult to collect coins from mining. It can be noted that already today, there are merchants accepting Bitcoins as payment for items.

6.4 Additional Resources

There are web pages that give an overview of all created blocks and all transactions in these blocks, giving a complete history of all transactions ever made using Bitcoin. An example of such a web page is *http://blockexplorer.com/*. Another web page of interest is *http://www.bitcoincharts.com* which provides some statistics and also the current exchange rates to other currencies.

References

[1] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems, Second Edition.* Wiley, 2008. Online chapters at *http://www.cl.cam.ac.uk/~rja14/book.html.*

[2] APACS. 2008 fraud figures announced by apacs. Available at: *http://www.ukpayments.org.uk/media_centre/press_releases/-/page/685/.*

[3] Simon Barber, Xavier Boyen, Elaine Shi, and Erzin Uzun. Bitter to better : How to make bitcoin a better currency. In *Financial Cryptography—FC 2012*, volume ???? of *Lecture Notes in Computer Science*, pages ???–??? Berlin: Springer-Verlag, 2012. Available at: *http://www.cs.stanford.edu/~xb/fc12/.*

[4] Bitcoin. Protocol Specification. Available at: *https://en.bitcoin.it/wiki/Protocol_specification.*

[5] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 302–321. Springer Berlin / Heidelberg, 2005.

[6] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 319–327. Springer Berlin / Heidelberg, 1990.

[7] EMVco. EMV specifications, 2011.

[8] S. Micali and R. L. Rivest. Micropayments revisited. In *Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology*, CT-RSA '02, pages 149–163. Springer-Verlag, 2002.

[9] S.J. Murdoch, S. Drimer, R. Anderson, and M. Bond. Chip and pin is broken. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 433–446, May 2010.

[10] Steven J. Murdoch and Ross J. Anderson. Verified by visa and mastercard securecode: Or, how not to design authentication. In Radu Sion, editor, *Financial Cryptography and Data Security, 14th International Conference, FC 2010*, volume 6052 of *Lecture Notes in Computer Science*, pages 336–342. Springer, 2010.

[11] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2009. Available at: *http://bitcoin.org/bitcoin.pdf.*

[12] A. Odlyzko. The case against micropayments. In W. Hunt and F. Somenzi, editors, *Computer Aided Verification*, volume 2742 of *Lecture Notes in Computer Science*, pages 77–83. Springer Berlin / Heidelberg, 2003.

[13] R. Párhonyi, L. Nieuwenhuis, and A. Pras. Second generation micropayment systems: lessons learned. In M. Funabashi and A. Grzech, editors, *Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government*, volume 189 of *IFIP International Federation for Information Processing*, pages 345–359. Springer Boston, 2005.

[14] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michael Quisquater, Louis Guillou, Marie Guillou, Gaid Guillou, Anna Guillou, Gwenole Guillou, and Soazig Guillou. How to explain zero-knowledge protocols to your children. In Gilles Brassard, editor, *Advances in Cryptology—CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 628–631. Springer Berlin / Heidelberg, 1990.

[15] R. Rivest. Electronic lottery tickets as micropayments. In R. Hirschfeld, editor, *Financial Cryptography*, volume 1318 of *Lecture Notes in Computer Science*, pages 307–314. Springer Berlin / Heidelberg, 1997.

[16] R. Rivest. Peppercoin micropayments. In Ari Juels, editor, *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 2–8. Springer Berlin / Heidelberg, 2004.

[17] R. Rivest and A. Shamir. Payword and micromint: Two simple micropayment schemes. In M. Lomas, editor, *Security Protocols*, volume 1189 of *Lecture Notes in Computer Science*, pages 69–87. Springer Berlin / Heidelberg, 1997.

[18] Visa. Verified by visa acquirer and merchant implementation guide, 2011. Available at: *http://usa.visa.com/download/merchants/us-vbv-acquirer-merchant-implementation-guide.pdf.*