TAPI: Transactions for Accessing Public Infrastructure

Matt Blaze[1], John Ioannidis[1], Sotiris Ioannidis[2], Angelos D. Keromytis[3], Pekka Nikander[4], and Vassilis Prevelakis[5]

[1] AT&T Labs – Research
{mab,ji}@research.att.com
[2] CIS Department, University of Pennsylvania
sotiris@dsl.cis.upenn.edu
[3] CS Department, Columbia University
angelos@cs.columbia.edu
[4] Nomadic Lab
pekka.nikander@nomadiclab.com
[5] CS Department, Drexel University
vp@drexel.edu

Abstract. This paper describes TAPI, an offline scheme intended for general Internet-based micropayments. TAPI, which extends and combines concepts from the *KeyNote Microchecks* and OTPCoins architectures, encodes risk management rules in bank-issued users' credentials which are in turn used to acquire small-valued payment tokens. The scheme has very low transaction overhead and can be tuned to use different risk strategies for different environments and clients.

Keywords: Micropayments, trust management, wireless networks, access control

1 Introduction

Traditional electronic payment systems impose a low bound on the value of each transaction due to the associated processing and clearing overhead. For small-value transactions, this overhead dominates the value of the transaction itself, making the use of such a system uneconomical. Various schemes have been proposed, aiming to reduce overheads so as to handle payments of fractions of a cent. These systems must cope with problems of scale, risk, and trust. It is important to have mechanisms that can scale to millions of transactions while maintaining acceptable levels of risk.

However, cryptographic and other computational operations have non-negligible cost. Thus, we need to minimize the crypto operations by aggregating them in larger transactions. Our observation is that we can take advantage of any locality of reference exhibited by micropayments, *i.e.,* a user paying for a service from a web site is more likely to purchase additional services from the same site. For applications where this holds, we can amortize the cost of many micropayments over a larger payment.

We present a mechanism that allows multiple partial charges against a payment authorization. By splitting a micropayment transaction into a number of partial transactions (minipayments) for smaller amounts, up to the amount of the original micropayment, we can accommodate multiple purchases within the original (single) transaction. Thus, we spread the cost of the transaction over a number of distinct purchases. A similar approach is being used by some vendors: multiple small credit card transactions are aggregated and presented as a single transaction to the credit card company. Typically, special agreements that cover liability and specify dispute handling policies need to be in place before this can be used. We built a system where dispute handling can easily be managed, *i.e.,* the merchant or the user can prove (or disprove) that a particular minipayment occurred, and thus limit the exposure to fraud.

We discuss a case study involving per-packet charging in a wireless network. In Section 2, we describe the background for our case study and arrive at the requirements for the charging scheme. We then describe the key features of our partial micropayment architecture by presenting a detailed example, namely a microcheck payment framework that is based on the KeyNote [1] trust-management system and a mechanism for making partial payments from a single microcheck. We briefly discuss our implementation in Section 3, and give an overview of related work in Section 4.

1.1 Motivation

The massive reduction in cost of wireless LAN (WiFi) base-stations has resulted in the gradual deployment of wireless LANs in public places by commercial operators, who want to charge for access. In most existing installations, the user must establish credit with the site operator (usually through a Web portal) before being allowed access to the network. As the density of WiFi coverage increases, the requirement for separate authentication with each provider becomes more onerous. Ideally, the user should be able to move between WiFi networks and access the Internet with authentication and payment done automatically. The rigidity of current payment methods (including the inability of the payment infrastructure to handle small payments) forces the network operators to charge for access in large time slots (*e.g.,* on a daily or weekly basis). The use of micropayments would allow the operator to be much more flexible. For example, the operator may wish to charge for each packet sent or received by the user.

Even if operators do not charge real money for the services offered, it is sometimes desirable to have some type of accountability, to detect infrastructure abusers (to avoid the "tragedy of the commons"). Such schemes still depend on some type of user registration and accounting; the missing part is the translation to the real-world concepts (*i.e.,* money). A micropayment scheme with "play" money can be used to provide this accountability. In order to be able to perform this type of charging, we need a system that satisfies the following requirements: First, it must be able to handle very small payments. Second, it should not require a user-initiated login procedure; instead, be able to receive payment dynamically. Third, it should not require on-line authentication with the user's credit institution in order to minimize connection overhead and accommodate situations where the user's credit institution is temporarily inaccessible.

To satisfy these requirements, we employ two different techniques: (a) The KeyNote trust-management System that establishes trust between the user, the service provider and the user's credit institution, using the architecture we originally described in [2], and (b) we introduce the concept of *OTP Coins* that allow single microchecks to be broken into smaller payment increments. We present these two techniques in detail.

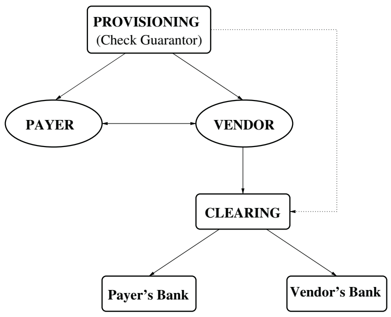

Fig. 1. Microbilling architecture diagram. We give the generic terms for each component, and in parentheses the corresponding players. Arrows represent communication between two parties: Provisioning issues credentials to Payers and Merchants; these communicate to complete transactions; Merchants send transaction information to Clearing, which verifies the transaction and posts the necessary credits/charges or arranges money transfers. Provisioning and Clearing exchange status information about Payer and Merchant accounts.

1.2 KeyNote Microchecks

The micropayments system introduced in [2] forms the basis of our approach. The general architecture of this microbilling system is shown in Figure 1. We consider an environment where Merchants and Payers sign up for service with a Provisioning Agent (PA). Merchants interact with Payers through the Merchant Payment Processor (MPP). The Clearing and Settlement Center (CSC) for reconciling transactions may be a separate entity, or may be part of the PA.

The PA issues KeyNote [1] credentials to *Payers* and *Merchants,* that describe the conditions under which a Payer is allowed to perform a transaction, and the fact that a Merchant is authorized to participate in a transaction. When a Payer wants to buy something from a Merchant, the Merchant encodes the details of the proposed transaction into an *offer,* which is sent to the Payer. To proceed, the Payer issues to the Merchant a microcheck for this offer. The microchecks are encoded as KeyNote credentials that authorize payment for a specific transaction. This credential is effectively a check signed by the Payer and payable to the Merchant. The conditions under which this check is valid match the Merchant's offer. Part of the offer is a nonce, which maps payments to specific transactions and prevents double-depositing of microchecks by the Merchant.

To determine whether he will be paid, the Merchant passes the offer description and the Payer's key along with the Merchant's policy (that identifies the PA key), the Payer credential (signed by the PA) and the microchecks credential (signed by the Payer) to his local KeyNote compliance checker. If the compliance checker authorizes the transaction, the Merchant is guaranteed that Provisioning will allow payment.

If the transaction is approved, the Merchant stores a copy of the microcheck along with the payer credential and associated offer details for later settlement. Otherwise, depending on their network connectivity, either the Payer or the Merchant can request a transaction-specific credential that can be used to authorize the transaction. This approach, if implemented transparently and automatically, provides a continuum between online and offline transactions tuned to the specific risk and operational conditions.

Periodically, the Merchant will 'deposit' the microchecks and associated transaction details to the CSC, which may or may not be run by the same entity as the PA, but must have the proper authorization to transmit billing and payment records to the PA for the customers. The CSC receives payment records from the various Merchants; these records consist of the Offer, the KeyNote microcheck, and the credential from the payer sent in response. In order to verify a microcheck, the CSC goes through the same procedure as the Merchant did when accepting the microcheck. If the KeyNote compliance checker approves, the check is accepted and the account balances adjusted.

The main advantage of this architecture is the ability to encode risk management rules for micropayments in user credentials. Other electronic systems have focused on preventing fraud and failure, rather than on managing it. As prevention mechanisms are often too expensive for micropayments, risk management seems particularly attractive.

1.3 OTP Coins

Electronic coins based on One Time Passwords (OTP) are another fundamental aspect of our approach. While the microchecks manage the risks of single transactions, OTP coins allow the cost of a microcheck to be distributed even more thinly, effectively making it possible to divide a microcheck transaction into hundreds of smaller, partial transactions. This approach is especially suitable for paying for access time, e-content, or other kinds of "continuous" goods, *i.e.,* goods that can be sold by some measure.

The basic approach, without microchecks, was outlined in [3]: an OPIE [4] OTP account was sent to the Client, who used the passwords to pay for wireless Internet access. The system was based on the IEEE 802.1x protocol, running OPIE over TLS.

When combined with microchecks, the Merchant spells out the OTP terms in the offer, *e.g.,* it might state he provides wireless Internet access time at \$0.001 per 5 seconds when bought in lots of 100 5 second units. That is, he offers 100 pieces of 5 second access time units for the price of \$0.1. If the Client accepts the offer, she generates a random number H_{100}, calculates a hash function over it 100 times, forming a reverse hash chain $H_{100}, H_{99}, ..., H_1, H_0$, where $H_i = hash(H_{i+1})$, and embeds the result H_0 into the microcheck she sends to the Merchant. The Merchant stores the hash value H_0 (called H_{check}) along with the number of remaining valid tokens, 100. At this point, the Merchant has sold to the Client 100 OTP coins, only valid with that Merchant. However, the construction allows the Client to be charged only for the actual amount spent.

When the Client wants to use the coins, she sends the next hash value to the Merchant. That is, she first sends H_1, then H_2, *etc.* The Merchant checks that the received hash value gives the previously stored value, *i.e.,* that $H_{stored} = hash(H_{received})$. If so, she decrements the number of remaining valid tokens, and stores the new received value. Thus, we have established a convention where a single OTP password represents the value for a certain commodity, *e.g.,* for 5 seconds of access time. Once the commodity has been used up, the Merchant asks for the next token, to continue service.

Once the Client has used all coins or stops using more coins, the Merchant possesses a hash value H_N where N is the number of coins used. When he deposits the microcheck to the CSC, he also sends these numbers. The CSC computes $H_{check} = hash^N(H_N)$

and compares this to the number stored in the microcheck. If they match, it can be certain that the Client has indeed bought N units of the good.

2 Architecture

We describe the TAPI architecture through an example use in pay-per-use 802.11 access. We then give a brief security analysis of our architecture.

2.1 Example Usage Scenario

As an example, we show how the system can be adapted to a public wireless Internet access using Wireless LANs. We begin with a client that has signed up with an acceptable Provisioning agent. Here, access points subsume the role of the Merchant and users play the role of the Payer. As a result of this registration process (which happens offline), the user is issued with a *clearing check,* signed with the PA's public key:

```
Authorizer: PA_KEY
Licensees: PAYER_KEY
Conditions: app_domain == "Internet Access" &&
      currency == "USD" &&
      &amount < 2.51 && date < "20031231" -> "true";
Signature: ...
```

Wireless LAN Authentication. The IEEE 802.1x standard [5] defines a means to authenticate clients in an Ethernet-like network, *e.g.,* it allows authenticating devices starting to use WLAN or a corporate LAN for Internet access. In practice, the standard defines how to run the IETF standard Extensible Authentication Protocol (EAP) [6] over raw Ethernet frames. The encapsulation is called EAP over LAN (EAPoL) [5].

Since we use the standard EAP protocol, it is possible to use any or all of its subprotocols. However, since neither EAP or EAPoL provide any cryptographic protection themselves, the security of the system depends on the security of the underlying network and on the properties of the EAP subprotocol. Thus, the risks and the protections must be matched to provide the desired level of security.

When 802.1x is used, there are two kinds of client hosts: authenticated and unauthenticated. In a wired LAN, the clients are usually distinguished based on the port: a physical port is either authenticated or not. In a shared medium, *e.g.,* Wireless LAN (WLAN), the distinction is usually based on the Layer 2 addresses. It may be possible to falsify or "steal" a MAC address, depending on the actual implementation. In the case of public WLAN, where no encryption is used, the only protection is the relative difficulty of using a MAC address at the same time another client is using it.

Buying OTP coins. Whenever a new client host wants to join a LAN that uses IEEE 802.1x, the access-point attempts to run EAPoL. The status of the client is kept unauthenticated as long as the client fails to authenticate through EAPoL. In our case, we provide unauthenticated clients limited access so that they can buy OTP coins, used for the actual EAPoL level authentication (see below). That is, any unauthenticated client is served (via DHCP) a private IP address. This address can be used only locally.

The client uses the MPP protocol to purchase a pile of OTP coins. In the simplest case (and lacking any special-purpose protocol for purchasing coins), a simple web interface can be used with the user. When the user contacts the captive portal, he sees a web page that encodes the details of the Merchant offer, *e.g.:*

```
merchant = "ADK'S WIRELESS"
currency = "USD"
product = "Internet Access"
date = "20020916"
packets_per_coin = "100"
coins_per_dollar = "10000"
amount = "2"
nonce = "eb2c3dfc860dde9a"
```

The user examines the details of the request and, if acceptable, authorizes a payment to the merchant by issuing the appropriate KeyNote microcheck:

```
Authorizer: PAYER_KEY
Licensees: "ADK's Internet"
Conditions: app_domain == "Internet Access" &&
      currency == "USD" && amount == "2" &&
      packets_per_coin == "100" &&
      coins_per_dollar == "10000" &&
      first_coin == "c637bf92f9f371dfa09\
              59bc467d04b91c2ea1b29" &&
      nonce == "eb2c3dfc860dde9a" &&
      date == "20001227" -> "true";
Signature: ...
```

The microcheck also contains the value for the first OTP Coin. This coin is not actually used, but serves as the beginning of the OTP chain. The next time the client needs to authorize a payment, she will use the next coin in the chain, *i.e.,* “310b86e0b62b82856-2fc91c7be5380a992b2786a”. The user sends this microcheck and its guaranteeing check, issued by the PA, to the access point. The latter verifies the integrity of the credentials and determines (by invoking KeyNote) whether the CSC will honor the payment terms.

Using OTP coins. Once the Client has acquired a set of OTP coins, she runs the standard 802.1x EAPoL protocol with the local access point. The access point requests a user identifier from the client, who answers with a string identifying the microcheck used for buying the OTP coins, and the merchant the coins where bought from. The access point then contacts the back-end authenticator (the Merchant). The microcheck fingerprint indicates the correct unused OTP coin pile.

Once the back-end authenticator receives the identity response, it checks the OTP coin pile and sends an OPIE request, requesting for the next unused OPIE password, *i.e.,* OTP coin. The Client responds with the next unused coin, H_{i+1}. The back-end authenticator checks the coin, records it as used, and replies with an EAP SUCCESS message. As the access point receives the EAP SUCCESS message from the back-end authenticator, it changes the status of the client into authenticated, and passes the message to the client. When the client receives the SUCCESS message, she releases her current IP address and requests a new one with DHCP. Since she is now authenticated, she gets a new IP address that she can use to communicate with the outside world. Alternatively, the client could have received a valid IP address which was appropriately filtered by the access point; on success, the relevant filters are simply removed.

Before the OTP coin is used up, the back-end authenticator sends a new OPIE request to the client. If the client wants to continue, she replies with the next OTP coin. On the other hand, if the client does not want to continue access for any reason, she simply does not respond to the request. Thus, if the client goes off-line, the access point changes the status of the client's MAC address into unauthenticated.

Clearing. Periodically, the access point provides all these microchecks along with the related transaction records to the CSC, which uses this information to verify the transaction and charge/credit the relevant accounts. The user's device (laptop, PDA, *etc.*) may also keep a record of all transactions, which can be used in case of a charge dispute. CSCs communicate with PAs to indicate the status of Payers' and Merchants' accounts. Part of the transaction records include the last OTP coin received from a user, and its serial number. The CSC can verify its validity, by repeatedly hashing it the appropriate number of times and comparing the result with the initial OTP coin included in the microcheck. Thus, the CSC can respectively debit the Merchant's account and credit the Payer's account for the appropriate amount. In case of dispute, the exact usage can be determined by verifying the credentials and the OTP coin chain. Assuming the underlying cryptography is not broken, the results are non-repudiable.

2.2 Security Analysis

The security of the system can be broken into two parts: one that relates to the security at the network level, and one that refers to the security of the payment mechanisms.

WLAN security. Wireless LANs are known to be notoriously insecure. However, their insecurity depends heavily in the way they are used. In our example case, where WLAN is used for providing public Internet access, the operator is mainly interested in collecting the access fees, while the clients are interested in getting the service they pay for. Other security concerns that the users may have (e.g. privacy) can be taken care of at an upper layer, and fall beyond the scope of this paper. Consequently, the main threats we are interested in are: (a) someone gaining access without paying, and (b) someone paying but not gaining access. Naturally, these threats may occur at the same time, through an attacker "stealing" access that another user has paid for.

Thus, it is certainly *possible* for an attacker to cause an authenticated client to disconnect from the network, and start using its MAC address. However, the access point is likely to detect the event and may require immediate re-authentication as the MAC address re-connects to the network. If the value of the OTP coins is low enough, *e.g.,* just a few seconds of access time, the gain for the potential attacker is small. Finally, it should be straightforward to detect an attacker that repeatedly steals MAC addresses.

A more powerful attacker can set up a phony access point. If he lures other clients to send OTP coins to it, he can then use these coins to pay towards the real access point, effectively riding free. The victim clients are unlikely to notice anything, since they still get the service they expect. The real access point may not notice anything either, depending on its sophistication. While this attack can be made harder, *e.g.,* by including the Client's MAC address in the microchecks, the simple nature of the OTP coins makes it impossible to block the attack altogether. However, given the current status of WLAN deployment, the cost of the attack compared to the benefits gained seems to be high enough to render the attack academic. For proper security, either the OTP coins must be replaced with something more sophisticated (and costly), or the underlying network must be secured. The additional cost should be evaluated against the expected risk and cost of fraud, and implemented only if economically viable.

Payment Framework Security. When dealing with electronic payments we must ensure that fraudulent transactions cannot take place, *e.g.,* the merchant should not be able to forge an OTP coin, nor should the user be able to deny that she has spent one.

The scheme requires that we select a non-reversible hash function. Thus, the merchant can verify that H_{n-1} was derived from H_n, but is unable to produce H_{n-1} given H_n. Similarly, if the client sends a number other than H_{n-1}, the merchant will detect that and revoke the service. Similarly, if the merchant produces H_{n-1}, the client cannot claim that she has not sent it. An extensive discussion of the security of the KeyNote microcheck architecture may be found in [7]. A key observation is that the low value of the checks and the need for light-weight verification mechanisms favor the use of credential expiration (with short lifetimes) over the use of a more heavyweight revocation mechanism such as credential revocation lists.

3 Implementation

We have implemented the IEEE 802.1x protocol and the OTP coins in the FreeBSD operating system [3]. Our initial performance measurements indicate that the effect on payload performance is negligible: a typical EAPoL transaction is performed in less than two seconds, making it possible to support re-authentication every 5-10 seconds.

Our 802.1x implementation consists of kernel code that implements the basic framing functions for the authentication protocol, plus a number of user level programs. The user-level programs implement the individual EAP subprotocols, and in particular the EAP OTP authenticator and supporting modules. These rely on a small new library, *libeap.* They also utilize the *libskey* library present in FreeBSD. To make it easy to buy OTP coins, we use a captive portal to allow users to download the 802.1x and EAP OTP implementations. Our MAC filter module forward to a web server packets arriving from unauthenticated users. Thus, it is possible to create a situation where the only services provided to an unauthenticated client are DHCP and the captive web server.

We are currently working on implementing the full-fledged MPP protocol on top of EAP, without the need for a captive portal. Users can specify their payment policies using

KeyNote (or some other front-end mechanism, which is then translated to KeyNote). On receipt of an offer from an access point, KeyNote is called to determine whether the terms are acceptable. If so, a microcheck is automatically issued, and the necessary OTP coins are generated and used without user interaction. If the offer is not acceptable, the user is notified and presented with the offer.

4 Related Work

IEEE 802.1x Security. IEEE 802.1x [5] is a forthcoming standard for authenticating and authorizing users in Ethernet like local area network (LAN) environments. It is primarily meant to secure switched Ethernet wireline networks and IEEE 802.11 based WLANs. In the typical usage scenarios, the network requires user authentication before any other traffic is allowed, *i.e.,* even before the client is assigned an IP address. This allows corporations to strictly control access to their networks. It is important to note that 802.1x implements only authentication and MAC address based access control. Since MAC spoofing is fairly easy, the resulting system may not be secure enough.

[8] argues that 802.1x security is flawed since it does not provide per-packet integrity and authenticity. Depending on the settings, this may allow session hijacking, enabling an attacker to take over a MAC address that belongs to a legitimate, authenticated user. In a shared medium such as 802.11, authentication should be tightly integrated with a link-level integrity system using different session keys for different clients.

Electronic Cash and Micropayments. NetBill [9] is a transactional payment protocol with many advanced features (atomicity, group membership, pseudonyms, *etc.*) that requires communication with the NetBill server for each transaction, thus exhibiting the same drawback with respect to micropayments as the simpler online protocols already mentioned. Other general payment protocols [10,11,12] suffer the same problem.

Digital cash-based systems (*e.g.,* [13]) do not directly address the issue of double-spending (fraud). Some e-cash systems use online checking, thus negating the off-line operation capability. Others rely on detection after the fact. This drawback is manifest in several micropayment protocols [14,15,16,17]. While double-spending is a problem in all off-line systems, none of these protocols address the issue of risk management.

NetCents [18] and Millicent [19] are scrip-based off-line-friendly micropayment protocols. As the monetary unit used in these protocols is vendor-specific, double-spending is made difficult. A hidden assumption is that merchants have "total information". If there are many points of sale, continuous communication and synchronization is required between the different points, negating the benefits of off-line operation.

MiniPay [20] was developed primarily for use with a web browser, with a lot of effort gone into the user interface aspect. Risk management is implemented as a decision to perform an online check with the billing server based on the total spending by the customer that day, and some parameter set by the merchant. The billing provider cannot customize the risk-management parameters per-customer or per-merchant. Fileteller [7] uses a scheme similar to ours for buying and selling network-based storage.

5 Summary and Concluding Remarks

We presented TAPI, a simple offline electronic payment scheme intended for general Internet-based micropayments. TAPIallows multiple partial charges on a single payment authorization by splitting a micropayment transaction into a number of minipayments for smaller amounts (totaling the amount of the original micropayment), thus accommodating multiple purchases within the original transaction.

To demonstrate our design in practice, we implemented the IEEE 802.1x protocol along with the OTP coins in the FreeBSD operating system. Our case study involved per-packet charging in a wireless network. Initial performance measurements indicated that the effect on payload performance was negligible and that the typical EAPoL transaction takes less than 2 seconds on average. We are in the process of implementing the full-fledged MPP protocol on top of EAP, without the need for a captive portal.

References

1. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.D.: The KeyNote Trust Management System Version 2. Internet RFC 2704 (1999)
2. Blaze, M., Ioannidis, J., Keromytis, A.D.: Offline Micropayments without Trusted Hardware. In: Proceedings of the Fifth International Conference on Financial Cryptography. (2001)
3. Nikander, P.: Authorization and charging in public wlans using freebsd and 802.1x. In: Proceedings of the Annual USENIX Technical Conference, Freenix Track. (2002)
4. N. Haller, C. Metz, P.N.M.S.: A One-Time Password System. RFC 2289, IETF (1998)
5. : IEEE Draft P802.1X/D11: Standard for Port based Network Access Control (2001)
6. L. Blunk, J.V.: PPP Extensible Authentication Protocol (EAP). RFC 2284, IETF (1998)
7. Ioannidis, J., Ioannidis, S., Keromytis, A., Prevelakis, V.: Fileteller: Paying and Getting Paid for File Storage. In: Proceedings of the Sixth International Conference on Financial Cryptography. (2002)
8. Mishra, A., Arbaugh, W.A.: An Initial Security Analysis of the IEEE 802.1x Standard. Technical Report UMIACS-TR-2002-10, Computer Science Department, University of Maryland (2002)
9. Cox, B., Tygar, D., Sirbu, M.: NetBill security and transaction protocol. In: Proceedings of the First USENIX Workshop on Electronic commerce, USENIX (1995)
10. Neuman, C., Medvinsky, G.: Requirements for network payment: The Netcheque prospective. In: Proceedings of IEEE COMCON. (1995)
11. Bellare, M., Garay, J., Herzberg, A., Krawczyk, H., Steiner, M., Tsudik, G., Waidner, M.: iKP – A Family of Secure Electronic Payment Protocols. In: Proceedings of the First USENIX Workshop on Electronic Commerce, USENIX (1995)
12. Foo, E., Boyd, C.: A Payment Scheme Using Vouchers. In: Proceedings of the Second International Conference on Financial Cryptography. Number 1465 in Lecture Notes in Computer Science, Springer-Verlag (1998) 103–121
13. Chaum, D.: Achieving Electronic Privacy. Scientific American (1992) 96–101
14. Rivest, R., Shamir, A.: PayWord and MicroMint. CryptoBytes (**2**) 7–11
15. Jutla, C., Yung, M.: Paytree: amortized signature for flexible micropayments. In: Proceedings of the Second USENIX Workshop on Electronic Commerce, USENIX (1996)
16. Hauser, R., Steiner, M., Waidner, M.: Micro-payments based on ikp. In: Proceedings of the 14th Worldwide Congress on Computer and Communication Security Protection. (1996)

17. Tang, L.: A Set of Protocols for MicroPayments in Distributed Systems. In: Proceedings of the First USENIX Workshop on Electronic Commerce, USENIX (1995)
18. Poutanen, T., Hinton, H., Stumm, M.: NetCents: A Lightweight Protocol for Secure Micropayments. In: Proceedings of the Third USENIX Workshop on Electronic Commerce, USENIX (1998)
19. Manasse, M.S.: The Millicent protocols for electronic commerce. In: Proceedings of the First USENIX Workshop on Electronic Commerce, USENIX (1995)
20. Herzberg, A.: Safeguarding Digital Library Contents. D-Lib Magazine (1998)