TAPI: Transactions for Accessing Public Infrastructure

Matt Blaze[1], John Ioannidis[2], Sotiris Ioannidis[3], Angelos D. Keromytis[4], Pekka Nikander[5], and Vassilis Prevelakis[6]

[1] AT&T Labs – Research, *mab@research.att.com*
[2] AT&T Labs – Research, *ji@research.att.com*
[3] CIS Department, University of Pennsylvania, *sotiris@dsl.cis.upenn.edu*
[4] CS Department, Columbia University, *angelos@cs.columbia.edu*
[5] Nomadic Lab, *pekka.nikander@nomadiclab.com*
[6] 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.

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 eas-

ily 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.

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.

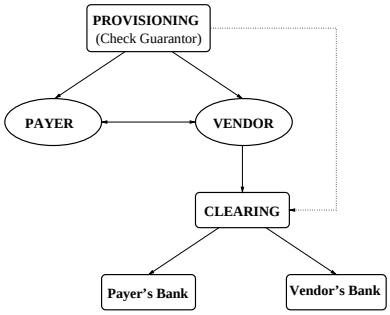

Fig. 1. Microbilling architecture diagram. We give the generic terms for each component, and in parentheses the corresponding players. Arrows represent communication between two parties: Provisioning issues credentials to Payers and Merchants; these communicate to complete transactions; Merchants send 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.

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.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.