Security Issues in M–Commerce: A Usage–Based Taxonomy

Suresh Chari[1], Parviz Kermani[1], Sean Smith[2], and Leandros Tassiulas[3]

[1] IBM T.J. Watson Research Center, Yorktown Heights NY 10598-0704
{schari,parviz}@us.ibm.com
[2] Department of Computer Science, Dartmouth College, Hanover NH 03755-3510
sws@cs.dartmouth.edu
[3] Electrical and Computing Engineering, Univ. of Maryland, College Park MD 20742
leandros@isr.umd.edu

Abstract. *M–commerce* is a new area arising from the marriage of electronic commerce with emerging mobile and pervasive computing technology. The newness of this area—and the rapidness with which it is emerging—makes it difficult to analyze the technological problems that m–commerce introduces—and, in particular, the security and privacy issues. This situation is not good, since history has shown that security is very difficult to retro–fit into deployed technology, and pervasive m–commerce promises (threatens?) to permeate and transform even more aspects of life than e–commerce and the Internet has. In this paper, we try to begin to rectify this situation: we offer a preliminary taxonomy that unifies many proposed m–commerce usage scenarios into a single framework, and then use this framework to analyze security issues.

1 Introduction

In the last few years, advances in and widespread deployment of information technology have triggered rapid progress in e–commerce. This includes automation of traditional commercial transactions (electronic retailing, etc.) as well as the creation of new transaction paradigms that were infeasible without the means of widely deployed information technology. New paradigms include electronic auctioning of purchase orders, as well as novel, money–less transaction models such as Napster [14]. E–commerce has heightened the focus on security both of systems and also for messaging and transactions [7,11].

In much the same way, recent advances in handheld *personal digital assistants (PDAs)*, wireless communication technology [17,19,26], and pervasive infrastructure [8,12,10,25,21] promise to extend this rich, comfortable environment to mobile users, and potentially to erase the distinction between the "off–line" and "on–line" worlds. As with e–commerce, we expect to see both the migration of current transaction models, as well as the emergence of new models made possible by this technology. Possible scenarios include:

– “buying soda at a vending machine with a mobile phone”
– “trading stock from a wireless laptop, in an airport”
– “picking up a coupon while surfing the Web from a desktop, then squirting the coupon from a PDA into a kiosk at the grocery checkout.”

This emerging area of *m–commerce* creates new security and privacy challenges because of new technology, novel applications, and increased pervasiveness.

Mobile applications will differ from standard e–commerce applications, because the underlying technology has fundamental differences:

– **Limitations of Client Devices.** Current (and looming) PDAs are limited in memory, computational power, cryptographic ability, and (for the time being) human I/O. As a consequence, the user cannot carry his entire state along with him, cannot carry out sophisticated cryptographic protocols, and cannot engage in rich GUI interaction.
– **Portability of Client Device.** PDAs have the potential to accompany users on all activity, even traditionally off–line actions away from the desktop. Besides creating the potential for broader permeation of e–transactions, this fact also makes theft, loss, and damage of client devices much more likely.
– **Hidden and Unconscious Computing.** Both to compensate for limited PDA storage, as well as to provide new ways to adapt a user’s computing environment to her current physical environment, *pervasive computing (PvC)* often permits client devices to transparently interact with the infrastructure—without the user’s direct interaction. This unconcsious interaction can include downloading executable content.
– **Location–Aware Devices.** When the user is mobile, the infrastructure can potentially be aware of the location of the user (e.g., in a particular telephone cell). This knowledge introduces a wide range of applications which have no analogue in the stationary user model.
– **Merchant Machines.** In the e–commerce world, the merchant (i.e., the party that is not the user) has powerful machines, with ample storage and computation, usually in a physically safe place. However, to fully exploit the potential interacting with mobile, PDA–equipped users, merchant machines may move out into the physical world. This move brings with its own challenges of increased physical exposure, limited computation and state, and limited interconnection.

Several of these applications scenarios have no analogues in stationary models. They give rise to new kinds of security issues—which need to be considered *before* designing protocols and deploying solutions. In this paper we aim to:

– to study the possible range of mobility scenarios,
– to propose a rudimentary taxonomy of possible application classes based on connectivity scenarios,
– to identify the security exposures and issues for each connectivity scenario
– to draw inferences from the discussion on exposures.

In Section 2, we introduce a set of entities which we are the main participants in the m–commerce world. In Section 3, we enumerate m–commerce usage scenarios based on these entities and their interaction. In Section 4, we consider the security and privacy implications of these scenarios. In Section 5, we try to generalize from this specific analysis to broader principles. In Section 6, we consider some directions for future research.

2 Entities

By definition, commerce involves commercial transactions between two or more entities. E–commerce moves at least some portion of this interaction to an electronic, computational setting. M–commerce goes further by moving some of this computation to a mobile platform. This characterization still leaves many variables open: Who are these entities? What are their computational platforms? How do these platforms interact during the course of an m–commerce transaction? We visit each question in turn.

2.1 Client Devices

Our basic model for a *client device* $\mathcal{C}$ is portable, moderately powerful computational device such as a PDA with a range of possible connectivity options which allows us to explore a wide range of mobility scenarios:

– $\mathcal{C}$ may have a physical connection to the client's desktop—but only when the client is at his desktop.
– $\mathcal{C}$ may itself be a wireless phone with a long–range wireless link.
– $\mathcal{C}$ may be equipped with a short–range link such as Bluetooth [17].

We make the simplifying assumption that the client device $\mathcal{C}$ is identified with the user of this device. At the application level, the authentication mechanisms for use in transactions will be different from those used at the link level to obtain services such as data transport. In some applications, this distinction will blur such as when the user uses the link–level authentication and authorization mechanisms to participate in transactions.

Physical security of the client device is a very useful property for the user to participate in protocols that can be proven, with high assurance, to be secure. While we do not assume that inexpensive devices such as current off–the–shelf cell–phones or PDAs (or, for that matter, desktop machines) offer sufficiently high resistance to physical attacks, we identify security exposures in protocols which could be avoided if $\mathcal{C}$ or $\mathcal{K}$ had these properties. With care, physical security can be imported into the client device via a secure hardware token such as a smart card or a smart button. Although the physical security of such devices is still in question [1,2,24], some smart buttons have had physical security independently evaluated at FIPS 140–1 Level 3 [15]. Physical security can be imported into desktops via a high-end, FIPS Level 4 secure coprocessor [23,27].

We note that, although many discussions of the physical security of client devices center on the resistance of the device to attack by a party other than the user, for many applications, it is important to consider resistance against attacks by the user himself. For example, an electronic wallet application may be subverted if a user can insert cash back into his wallet.

2.2 Kiosk Devices

A *kiosk device* $\mathcal{K}$ is our abstraction of the computational entity embodying the merchant with which the user participates in a transaction. In the normal e–commerce setting, the kiosk $\mathcal{K}$ is the server in the client–server model of interaction. However, in the broader world of m–commerce application scenarios, the kiosk $\mathcal{K}$ can have many form factors, such as:

– the normal web server/site which the client connects to either directly or through a proxy;
– an intelligent vending machine which the client can connect to using a local connectivity link;
– a passive identifying tag attached to painting in a museum.

From a security standpoint, there are different exposures depending on the entity that controls the physical location of the kiosk $\mathcal{K}$, and the points of attack against it. On one end, we have the kiosk as a backend server which can talk the WAP [26] protocol suite. In this setting, the client can directly establish a secure connection directly with this kiosk. This is similar to the traditional e–commerce model, where the kiosk $\mathcal{K}$ is a machine with large computational resources and broadband connections to other servers which it may contact to complete the transaction on behalf of the client such as a bank. The backend server can have strong security features such as being located behind a firewall, with access to a physically secure device. These features mitigate a number of security threats. However, a backend server with sophisticated connectivity might be subject the vulnerabilities that repeatedly surface in such complex boundaries.

At the other end, m–commerce introduces new scenarios where the kiosk $\mathcal{K}$ is a disconnected machine physically located in a remote setting. For instance, the kiosk $\mathcal{K}$ could be a vending machine and the client may connect to it using a local link. Here, the kiosk $\mathcal{K}$ does not have rich computational resources, and may also lack the ability to connect directly to a bank/payment center to verify the user's payment on–line. Since those with direct physical access to the device may not share the same interests as the owners of the kiosk $\mathcal{K}$, physical security issues [15] and physical protection of sensitive computation and data [23,27] become much more critical. Lack of high bandwidth connectivity also creates issues with maintaining and monitoring state at the device.

Early e–government initiatives gave serious consideration [9] to the security and usability issues of kiosks, since these were an avenue to bring information technology to broader populations than desktops could reach.

2.3 Infrastructure Servers

A fundamental difference between traditional e–commerce scenarios and their mobile counterparts is the role of the infrastructure. In mobile settings, the client device $\mathcal{C}$ and, very possibly, the kiosk $\mathcal{K}$ may suffer from fundamental restrictions such as a limited form–factor, very limited computational resources, small amounts of memory, lossy and high–latency connectivity., etc. Due to these limitations, the application frameworks are fundamentally different. For instance, a wireless–PDA wishing to access a web–site requires an intermediate machine to *transcode* the content so that it fits into its limited form–factor.

We abstract this external entity, which is part of the infrastructure potentially participating in mobile commerce transactions, as the *infrastructure server* $\mathcal{S}$. Examples of the infrastructure server include:

– **WAP Proxies.** In the WAP [26] architecture, the client device contacts a proxy using the WAP suite of protocols which then contacts a web server on the client's behalf. Besides bridging transport level differences in protocols on either side, the proxy also decrypts and re–encrypts data from the web server to the client. Other examples of such proxies are the Palm.Net proxy which manages user profiles besides transcoding content.
– **Wireless Gateway.** In another example, the user could potentially use the link level authentication mechanisms such as the *Subscriber Information Module (SIM)* of the wireless phone to pay for transactions. Here the infrastructure server is the wireless gateway which authorizes the use of the SIM for this transaction. This is the canonical "using a wireless phone to pay for soda at the vending machine" example.

By definition, the infrastructure machines is much more powerful than the other players in the m–commerce transaction. It can derive considerable physical security from its location but in a number of scenarios it can benefit from having physical security such as secure coprocessors.

Thus, we see that m–commerce introduces a critical distinction: the nature of client and kiosk technology introduces the need for this third element *which may be controlled by a party other than the user and merchant.*

Payment Gateways. Since commerce involves the exchange of services and merchandise for payment, implicit in all our discussions of m–commerce is the issue of electronic payment. (See [4] for an excellent overview of electronic payment systems.) We assume that financial institutions which facilitate payments are part of the infrastructure. In this section, we survey existing techniques for electronic payment systems.

We categorize electronic payment schemes based on what options are chosen for the following paramters:

– **Anonymity.** Can the spender can be tracked by the financial institution? (The scheme is *identifiable* if it can, *anonymous* otherwise)
– **On–line/Off–line Checks.** Is a connection with the bank required during the transaction?

Identifiable payment schemes contains sufficient information to reveal the identity of the person who originally withdrew the money from the financial institution, which can then further track the money as it moves in transactions. In contrast, anonymous payment schemes work more like paper cash; money can be spent without leaving a transaction trail. (Indeed, many researchers are particular about using "e-cash" or "e-money" *precisely* for those electronic payment schemes which are anonymous, atomic, and widely-accepted.)

As tempting as it may be to assert equivalence to paper, we note that real implementations of e-cash suffer from substantial differences [22]. For example, in the seminal DigiCash protocol [6]:

– Critical atomicity and anonymity properties will fail if the connection drops while Alice is receiving an e-dollar from Bob. [27]
– When Alice receives an e-dollar from Bob, she cannot turn around and spend it without first going to the bank.
– Alice cannot easily make change, without revealing her identity.

Once identified, many of these drawbacks can be overcome (e.g., [3].)

The main security exposure with anonymous e–cash is the problem of double spending. On–line schemes prevent double spending by requiring vendors to contact the financial institution during every sale, in order to determine whether a particular electronic dollar is still good. (In some schemes, the financial institution maintains a database of all the spent pieces of e-money; in others, it maintains a database of the unspent pieces.) This is very similar to the way vendors currently verify whether a particular credit card number corresponds to a valid account. Off–line schemes handle double spending in different ways:

– **Prevention.** Cash is stored on tamper–resistant smart cards which contain a small database of all transactions. Double spending can easily be detected since the hardware token (in theory) prevents tampering [13,27].
– **Detection.** Another approach is to *detect* double spending: a double spender is *identified* when the cash is redeemed at the bank. But rather than stopping double spending, this technique merely deters it. The main advantage of these schemes is that they do not require tamper–proof hardware.

3 Usage Models

In our abstraction of m–commerce, we thus have three entities: the client device $\mathcal{C}$, the merchant device $\mathcal{K}$, and the remote infrastructure $\mathcal{S}$. These entities interact during a transaction. For example:

– $\mathcal{C}$ may scan a barcode on $\mathcal{K}$
– $\mathcal{C}$ and $\mathcal{K}$ may establish a wireless Bluetooth link;
– $\mathcal{C}$ may establish a telephone link to $\mathcal{S}$
– $\mathcal{K}$ may establish a telephone link to $\mathcal{S}$
– $\mathcal{K}$ may be connected to $\mathcal{S}$ permanently via a LAN.

The interaction may not always involve the conscious action of the user: e.g., PvC's unconscious computing, where a PDA interacts on its own. The interaction may be unidirectional: e.g., client GPS. Furthermore, the user's perception of the interaction may differ substantially from the actual connection: e.g., the $\mathcal{C}$

may appear to connect to $\mathcal{K}$ directly, even though the connection is via a WAP gateway $\mathcal{S}$ that may in turn transparently redirect $\mathcal{C}$ to a different $\mathcal{S}$.

These three entities can interact with $2^3 = 8$ possible combinations of connectivity scenarios. For simplicity, we only consider connectivity at the time the transaction is being performed, and omit issues such as off–line connections. For example, in the soda–buying example, the kiosk $\mathcal{K}$ (vending machine) may be disconnected from the infrastructure when the transaction is actually performed, but may intermittently synchronize with a server to deposit the tokens to be redeemed for payment or these tokens may physically be removed from the kiosk through other means.

To formalize these scenarios consider a bit vector, where the most significant bit represents $\mathcal{K} - \mathcal{C}$ connectivity, the next $\mathcal{K} - \mathcal{S}$ connectivity, and the last $\mathcal{C} - \mathcal{S}$ connectivity. We case out the interesting connectivity scenarios. To be meaningful for m–commerce, both the client device $\mathcal{C}$ and the kiosk device $\mathcal{K}$ should be session–connected with at least one other entity. This gives a rudimentary taxonomy of five basic scenarios. In the following Section 3.1 through Section 3.5, we consider each in turn.

3.1 $\mathcal{K} - \mathcal{C}$ (100): Disconnected Interaction

In the *disconnected* case, illustrated by Figure 1, the client $\mathcal{C}$ and the kiosk $\mathcal{K}$ are disconnected from the infrastructure, and communicate with each other directly using a local link such as Bluetooth. Compelling examples include:

– **User at Vending Machine.** Here the client device $\mathcal{C}$ is a PDA, and the kiosk $\mathcal{K}$ is a vending machine. The user connects using a local connection, such as Bluetooth. A payment scheme for such a transaction needs to take into account that the kiosk $\mathcal{K}$ can perform no on–line checks (see Section 2.3).
– **Coupons.** Many payment–free examples are also possible: For example, schemes where a user retrieves *coupons* from one kiosk $\mathcal{K}$ and deposits them at others are arguably m–commerce.
– **Parking Validation.** Intermediate schemes are also possible. For example, a user's $\mathcal{C}$ might pick up a parking ticket in a parking garage, validate this ticket by interacting with a kiosk $\mathcal{K}$ in one of several nearby stores, and then inject his ticket (indicating he owes full value, or no value, if the ticket was validated) when he leaves the garage.
– **Local Services.** In many pervasive computing scenarios, traveling users access the devices they need—printers, video projectors—simply by pointing their PDA at them. When this access moves from "research lab conference room" to "cybercafe," then this money–less transaction can arguably become m–commerce.

3.2 $\mathcal{K} - \mathcal{S} - \mathcal{C}$ (011): Server-Centric Case

In the *server-centric* case (shown in Figure 2) the infrastructure server $\mathcal{S}$ is connected to both the client $\mathcal{C}$ and the kiosk $\mathcal{K}$. The $\mathcal{S}$ then acts as a proxy. Some examples are:

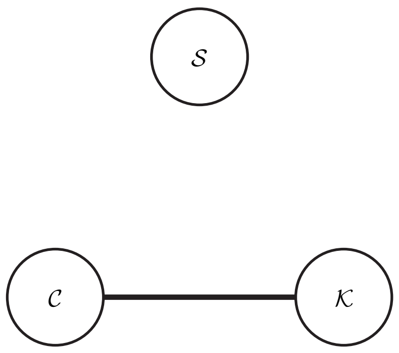

Fig. 1. The Disconnected Case: the client $\mathcal{C}$ and $\mathcal{K}$ interact directly, without on–line connection to the infrastructure $\mathcal{S}$.

- **WAP/Palm.Net.** Here the client device $\mathcal{C}$ is a wireless phone/PDA talking to a kiosk $\mathcal{K}$ which is a backend website. Since $\mathcal{K}$ and $\mathcal{C}$ do not talk the same protocol suite, the infrastructure server $\mathcal{S}$ performs the tasks of transcoding, splicing security protocols, etc. The three entities need not be in the same security domain, although this creates the obvious lack of end–to–end security since the proxy will decrypt and re–encrypt the content. This is the architecture of the emerging Wireless Applications Protocol suite [26]. The Palm.net [18] mode of interaction is similar, although in this case the server and the client device $\mathcal{C}$ talk the same set of protocols.
- **Vending Machines with On–line Checks.** The vending machine can make an on–line check when the user makes a payment for the transaction. In the example of buying soda using a cell–phone: the user calls a backend server, which transfers funds and calls the vending machine, to inform it to dispense the soda.
- **Location–Based Applications.** Other applications fitting this interaction model are location–dependent applications. If the user device is in contact with the infrastructure (for example, when the user's cellular PDA is turned on), the infrastructure can track the user's location, establish connection with a physically close kiosk, and assist in a potential transactional interaction. Examples of such transactions are the i–mode transactions deployed by NTTDoCoMo[16].

3.3 $\mathcal{K} - \mathcal{C} - \mathcal{S}$ (101): Client–Centric Case

In the *client–centric* case (shown in Figure 3), the client device $\mathcal{C}$ is connected to both the other entities, with a local link to the kiosk $\mathcal{K}$ and a wireless link to the infrastructure. Applications fitting this mode include: Imaginary Buffer Line

- **User State on Server.** The infrastructure alleviates the lack of memory on the client device $\mathcal{C}$ by keeping the user state. For instance, a user could

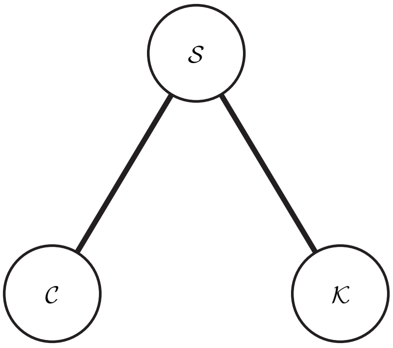

Fig. 2. The Server-Centric Case: the client and kiosk connect indirectly through the infrastructure server.

keep a repository of coupons on the server; when it is physically close to a kiosk $\mathcal{K}$ and wishes to participate in a transaction, the user device $\mathcal{C}$ can retrieve the appropriate coupon

- **Device State on Server.** The infrastructure could also store executables, so that the client device $\mathcal{C}$ only need store what its current context requires. (This model is often envisioned for speech-enabled PDAs: they pick up application-specific vocabulary from a kiosk in the user's physical environment.)
- **Cooltown.** The Cooltown scenarios ([8]) can fit into this case. The kiosks consist of very simple, passive tags. A client $\mathcal{C}$ interacts with a kiosk $\mathcal{K}$ by reading an identifier from the tag, then going into the infrastructure to retrieve the kiosk application.

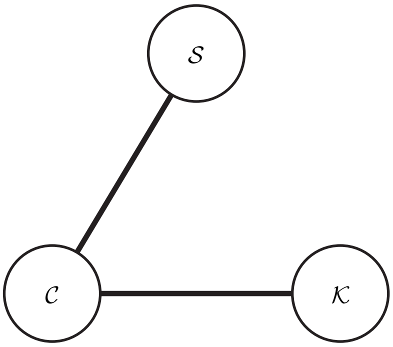

Fig. 3. The Client–Centric Case: the client device $\mathcal{C}$ connects both to the infrastructure and to the kiosk, but the kiosk is otherwise disconnected.

3.4 $\mathcal{C} - \mathcal{K} - \mathcal{S}$ (110): Kiosk–Centric Case

The *kiosk–centric* case (shown in Figure 4) is very representative of mobile application frameworks. The following are some examples:

– **ATM/Point–of–sale Terminal.** Here the user $\mathcal{C}$ communicates to the ATM or the Point–of–Sale terminal $\mathcal{K}$, which is connected to the infrastructure and can make on–line checks about the user during payment.
– **Traveling User.** Many proponents of pervasive computing predict scenarios where mobile users access remote infrastructure via local devices. For example, rather than carrying his file system—or a replica of a shared file system—with him, a user might simply point his $\mathcal{C}$ at a portal in a coffee shop or airport club, which in turn retrieves the data and potentially writes back changes. [20] Additionally, the backend server might use a large display on the kiosk to display non–secure information and display the sensitive information only on the user device.

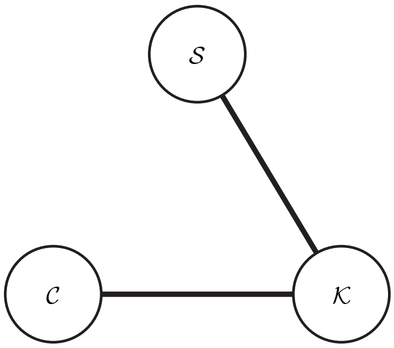

Fig. 4. The Kiosk–Centric Case: the client device $\mathcal{C}$ interacts directly with the kiosk, which in turn connects to the infrastructure.

3.5 Full Connectivity (111)

In the *fully connected case* (shown in Figure 5), all the entities are directly connected to one another. For example, suppose the client device $\mathcal{C}$ is a PDA with both Bluetooth and cell—phone capabilities, and the kiosk device $\mathcal{K}$ is network-connected.

The application scenarios here can include all the examples of the previous cases—if the technology supports it. Functionally, *any* transaction that can be performed in the fully connected case can be performed (with substantially more complicated protocols) in the biconnected models discussed above. However, full connectivity allows us to substantially simplify the protocols and obtain stronger security guarantees than similar applications in the other models. Full connectivity can also allow us to obtain other subtle security guarantees such as privacy and anonymity. For example, the user and kiosk might want to engage in some anonymous interaction, but both require backend resources to carry out this interaction. In this situation, using the infrastructure instead of a local connection to carry out the $\mathcal{K} - \mathcal{C}$ interaction would require some effective anonymous routing scheme, which can be expensive. The existence of all connectivity substantially reduces denial–of–service type attacks possible by implementing the same applications in other connectivity scenarios. (Section 4 will provide more discussion of these security issues.)

An application scenario possible with full connectivity, but not (easily) possible with weaker scenarios, is:

– **Proximity–Triggered Applications.** A kiosk and a client device both equipped with very short range connectivity like Bluetooth can detect when they are in close physical proximity. Additionally, if the client device $\mathcal{C}$ is connected to the backend through a wireless link and the kiosk $\mathcal{K}$ is connected to a LAN, one can think of several applications triggered by the kiosk $\mathcal{K}$ and client device $\mathcal{C}$ detecting physical proximity. (Detecting proximity can be possible without a direct $\mathcal{K} - \mathcal{C}$ interaction—e.g., both could use GPS—but this is much more complex.)

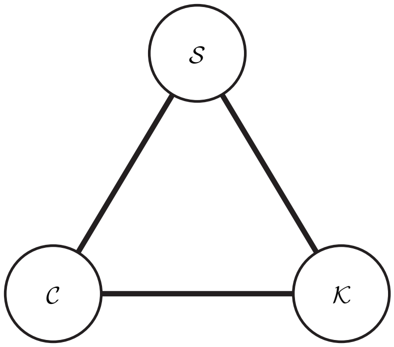

Fig. 5. The Fully Connected Case: all three entities are directly connected.

4 Security Exposures

In this section, we consider some of the security and trust issues raised by the cases and examples of Section 3. Besides the characteristics of the individual entities themselves, the security exposures and protocols are mainly decided by the connectivity between the entities. We consider each of the connectivity scenarios in turn. Essentially, the connectivity scenarios can be partitioned into three distinct classes:

– the disconnected case (Section 3.1);
– biconnected cases (Section 3.3, Section 3.2, Section 3.4)
– the fully connected case (Section 3.5).

We group the biconnected cases into one class since a number of exposures will be common. However, we will also point out individual differences in each of the cases.

4.1 Disconnected Interaction

At first glance, this case seems like the straightforward client–server interaction. However, the limitations of the PDA, and the fact that kiosks may be distributed in many physical locations, make this very challenging from a security perspective. We consider some problems.

Disconnection. In the disconnected case, by definition, both entities are disconnected from the infrastructure. Even if the kiosk $\mathcal{K}$ intermittently synchronizes with the infrastructure, neither it nor the client device $\mathcal{C}$ can have access to the latest state information. Conversely, neither the user nor the kiosk can update the information in the infrastructure. This immediately introduces a number of security exposures:

– **Double–Spending.** In the vending machine examples, the kiosk cannot check if the e-dollar or the coupon offered by $\mathcal{C}$ has already been spent somewhere else. Note that in some of the anonymous cash schemes, the protocol can detect *a posteriori*, if necessary, that double–spending had been done and identify the user [6].
– **Credential Freshness.** Most cryptographic protocols function by having one of more parties authenticate themselves during the transaction. With public keys, this is usually done by presenting a certificate which binds a public key with a name. Being disconnected, neither the client nor the kiosk can verify if the certificate is still valid and not been revoked.

Although these primary examples pertained to read-access (i.e., the client and kiosk were not able to read the current state in the infrastructure), write-access scenarios exist as well. For example:

– **Lack of Updates.** The kiosk cannot record that a particular e-dollar has been spent.

Many of the difficulties caused by disconnection can be characterized as a lack of *freshness*: the state at one $\mathcal{K}$ (or $\mathcal{C}$) lags behind or ahead the global merchant (or user) state. However, disconnection also eliminates economies of “information scale.” For just example: the price a merchant charges for a commodity may directly depend on his cost, which can drop with quantities he buys. But if the merchant cannot know that each of 100 kiosks has each sold 10 units, his kiosks cannot reflect his wholesale price on 1000 units.

Integrity of State. The second related exposure is the possibility of *rollback.*

Due to the limited physical security of off–the–shelf PDAs and the potentially exposed nature of kiosks, we must assume that, in the absence of extraordinary countermeasures, a dedicated adversary will be able to manipulate the internal state of both $\mathcal{C}$ and $\mathcal{K}$. This ability leads to a particularly acute security exposure. Interaction changes the state of the kiosk and the client in some well-defined way. However, because of the lack of connection, a delay exists between synchronizing this state with remote storage, and because of the lack of physical security, this uncommitted state may be subject to manipulation. Cryptographic techniques can prevent some types of manipulation. However, a device state can always potentially be rolled back to pre-interaction state. This creates a host of security and privacy challenges, typified by the following example:

– **Active Double Spending by Rollback.** Before spending e–cash at the vending machine, the client can record the complete internal state of the PDA, then restore this stored state afterwards and spend this cash again. Other variations exist—for example, a coupon marketing scheme might allow for unlimited distribution of a coupon, but require that a customer only use such a coupon on their first purchase.

Secure coprocessors and hardware tokens offers a potential solution to the problem of rollback, as discussed later.

Privacy of State. Another consequence of the lack of physical security is that the state at a device can be read and cloned. This exposure manifests itself in authentication. In the general case, if one entity cannot be trusted to keep its authentication secrets secret with high assurance, then the other entity can compensate by checking with some global state. But in the disconnected case, on–line checks are not possible.

Consider a user with a PDA interacting with a vending machine, with both entities disconnected from the infrastructure. How can the kiosk authenticate the client? Firstly, as pointed out, the kiosk can not check the freshness of the certificate presented. Secondly, even assuming that the certificate was not revoked, the certificate binds just binds a name to a public key which is confirmed by the possession of an equivalent private key. Without secure storage, this private key could easily have been obtained illegally.

4.2 The Biconnected Cases

The biconnected cases provide the potential to overcome some of the principal shortcomings of the disconnected case. Given our implicit assumption that both the user and the merchant have some portion of safe trusted storage in the infrastructure, then the entity that has a link to this infrastructure can also be a proxy extending this link to the other entity. The most natural example of this is in the server-centric case, where the server can route or forward $\mathcal{C} - \mathcal{K}$ communications to $\mathcal{K}$, and vice-versa. Note that the kiosk-centric and even the client-centric cases can carry out process of forwarding the messages to simulate full connectivity (if the device technology actually supports this.)

The main security issues which arise in this context are:

– **Active Translation.** The best example of the server-centric model is the Wireless Applications Protocol suite. Here the server (WAP proxy) needs to translate between the user's protocols and those of the kiosk since the client device $\mathcal{C}$ and kiosk do not speak the same protocols.
– **Privacy and Authenticity of Communications.** The intermediate entity can potentially try to attack the communications between the other two parties. Typically attacks involve altering the contents or the order of messages and replaying messages sent earlier.
– **Traffic Analysis.** Even if the intermediate entity does not actively modify the messages passing through, it will be aware of the frequency and length of the messages being exchanged. This can potentially lead to a substantial breach of privacy.
– **Denial of Service.** The intermediate entity can suppress messages meant for the other parties. This attack can especially crucial if one considers transactional protocols where the attacker may choose to stop sending "commit" messages.
– **More Points of Attack.** More complex message paths typically bring in more points of failure—and more points of attack. For example, if a portion of the link involves a LAN, then other devices on this network might carry out any of the attacks above.
– **Increased Failures.** The existence of more points of failure can increase the likelihood of failure of communications during a protocol. This can be a serious issue, since standard fault-tolerance techniques are not always orthogonal to security protocols—indeed, their naive application can actually subvert protocols (e.g., e–cash [27]).

Although we have grouped all the biconnected cases into one class, individual scenarios within this class bring their own concerns.

– **Client-Kiosk Identification.** Standard authentication techniques can address the problem of whether one party is in fact communicating with another party with a specific identity. But these techniques are effective only if one knows the identity in the first place.
In many m–commerce interactions, the pairing of a $\mathcal{C}$ and $\mathcal{K}$ occurs because of physical co-location and user intention. For example, the user wants the system to know that he wants to buy from the vending machine in front of him—the user doesn't want to pre-load the serial number of a particular vending machine or to stand and punch in its serial number. Direct $\mathcal{C} - \mathcal{K}$ interaction techniques provide a painless way to address this problem—for example, the user communicates his intention by pointing his PDA at the vending machine, and a Bluetooth exchange establishes the rest. However, the server-centric case dispenses with this link. This situation can create some interesting risks. For example, a server trying to match a user to the closest vending machine may not necessarily be aware of the locked door separating the user from the physically closest machine.
– **Privacy Issues.** In applications based on cell–phones, by definition, the $\mathcal{S}$ will know the physical location of client device $\mathcal{C}$. This creates privacy

risks: for example, it might make it easier for a terrorist group to locate a particular targeted dignitary. With current technology, $\mathcal{C} - \mathcal{S}$ interactions appear to inevitably involve this technology, so we might reasonably conclude that complete user privacy cannot be possible in any server–centric or user–centric scenario.

4.3 Full Connectivity

In the fully connected case, the security exposures for all the above scenarios are substantially mitigated by the fact that there are direct connections between the different entities. For instance, several of the denial–of–service attacks possible in the biconnected cases are not possible in the fully connected case. In the biconnected case where the kiosk is forwarding user messages to the infrastructure, the kiosk can selectively drop messages. The protocol for the biconnected case has to be resistant to these attacks. However, with full connectivity, the protocols could be substantially simpler.

5 Discussion

In this section, we abstract from the discussion of security exposures in the previous section and discuss some approaches to mitigate these exposure. As before, we consider these topics based on connectivity.

5.1 Disconnected Case

As seen in the previous section, this case offers the most challenging scenario for security measures. At first glance, it would appear that the challenges in the disconnected case leave us with the following impossibilities:

- There is no effective way to do e-cash—or any other payment scheme—without risk of attacks such as double-spending.
- There is no effective way to prevent malicious rollback of a critical state transition.
- There is no effective way to authenticate the other party, against an adversary who may have extracted its secrets.

However, hardware which can guarantee physical security give us some mechanisms to address these problems:

- If both $\mathcal{C}$ and $\mathcal{S}$ were equipped with high–end secure coprocessors [23,27] that could be trusted to carry out their computation unmolested despite physical attack, then we could solve the cash problems. The technique could also extend to other state transitions—except, of course, where an adversary could benefit from simply destroying the device and losing all state.
- Suppose instead that the devices possessed "secure hardware" that takes the form of a non-reproducible label: for example, the random physical structure of the surface of the device contains some random number, that the other device can reliably read. In this scenario, the authentication problem could be solved.

Embedding secure coprocessors or secure hardware is essentially a way to deploy the infrastructure into the user device or the kiosk. Imaginary Buffer Line One might argue that a secure coprocessor at either entity in the disconnected case would transform the model into the equivalent of the biconnected case, because the coprocessor can become a trusted proxy for the missing party. (But on the other hand, one might argue that some problems of disconnection still could not be solved: two distinct coprocessors still cannot synchronize state.)

This reasoning suggests a potentially interesting avenue of theoretical work:

– What is a precise formulation of the transaction properties that cannot be achieved with the base disconnected case? Can we prove that this base case cannot achieve these properties?
– What minimal amounts of secure hardware (and of which type) do we need to add (and to how many entities) in order to achieve these properties?

In some sense, one can imagine a theory here similar to distributed and disconnected file systems: e.g., transactions that could be subverted if the other party has rolled back state require synchronization beforehand. It would be interesting to see if this reasoning could be formalized, and some general principles proven with this formalism.

Disconnected operation has one advantage over the other connectivity cases: transactions can be anonymous and privacy–preserving. Also, since no connections are made to the infrastructure, fault tolerance can be increased, since the trustworthiness or reliability of the infrastructure is irrelevant.

5.2 The Biconnected Cases

The connectivity models and examples described in the biconnected cases (Section 4.2) are in our opinion very representative of the security concerns in mobile applications.

Biconnected cases can simulate full connectivity by the entities establishing a three–party protocol: standard encryption and message authentication techniques can address the confidentiality and message ordering issues. However, the protocols must be carefully designed since the intermediate entity controls the entire data flow and can mount denial–of–service attacks. Care must be taken since not every protocol can be systematically hardened against selective denial-of-service. Techniques to hide the identity of parties in a networked connection are expensive and not always effective in practice. Techniques to hide the existence of communication are extremely expensive.

The case of the Wireless Applications Protocol [26] is an interesting example of the server–centric model. The setting here is the wireless client requesting a web page from the kiosk(the backend server) through the infrastructure server(the WAP proxy). Since the kiosk $\mathcal{K}$ and client device $\mathcal{C}$ do not speak the same protocol suite, the proxy has to translate and in the case of secure communication decrypt and re–encrypt the data. This is an obvious security hole unless the proxy and the kiosk are in the same security domain. One way to address this problem is for the server to contain a secure coprocessor which is controlled by the kiosk (backend server). The actual translation is done inside the secure coprocessor which essentially acts as an agent of the kiosk residing at the proxy [5].

Lastly, arguing that the biconnected cases are equivalent also overlooks the vastly different computational and I/O properties of the entities. It is impractical to assume that in the client–centric case that the kiosk could communicate to the infrastructure through the client: the client device $\mathcal{C}$ might not have the capability to support robust cryptography and, also, the user might not be willing to pay for the bandwidth required.

5.3 Full Connectivity

We only wish to note that privacy and the integrity of protocols can be substantially enhanced in the full connectivity model. Protocols can be proved to be correct without the existence of high assurance secure coprocessor.

6 Future Research Directions

As wireless and mobile computing continues to evolve, m–commerce application scenarios that go beyond the framework described above will emerge. Examples include:

- Ad-hoc networking of mobile devices will facilitate user-to-user connectivity without the intervention of more powerful entities. Hence there will be the opportunity for more casual interaction and new m-commerce models, directly between the users $\mathcal{C}$, will emerge. For instance, when two users happen to be in the same environment may sell stock quotes one to the other or even sell access to the Internet (e.g., by sending email using the other user's laptop that is connected to the Internet). Again, an important issue here is the advertisement of services. The PDA of an individual may broadcast continuously the type of services that are available for use by other users and the "price" of those services. At the same time an individual may broadcast the type of services that he needs and they are not available to him from his own resources. Hence the PDA's will attempt to match services and requests when the users are in proximity, without the direct involvement of the individuals. Furthermore such a matching may occur even when the individuals are not in direct contact but they communicate through intermediate forwarding nodes.
- More sophisticated m–commerce models will enhance the flexibility of existing transaction paradigms. For instance the right m–commerce model might enable flexible subscription models for frequent travelers. When somebody registers and pays for a certain magazine, instead of having each issue of a magazine mailed to his house, he would like the flexibility to grab the magazine from the news stand in the airport if he wishes to do that instead or download it to his e–book from the info–kiosk.
- Finally the proliferation of micro–sensor technology in general and in particular the integration of sensors with communication transceivers will have

a significant impact on human I/O to the machine. Micro–sensor technology in combination with active badges will enable ubiquitous input to the computer without human intervention. Hence applications like automatic inventory and automatic ordering systems will become feasible. This will be useful for new distribution models for goods directly to the consumer.

7 Conclusion

Mobile application frameworks will likely be considerably different from the stationary user model, which creates a range of new security exposures. It is imperative to understand these exposures and design frameworks with security in mind *before* deploying applications and then retro-fitting security. In this paper we have identified some frameworks and their inherent exposures. We hope this spurs new research in secure protocols for mobile applications.

References

1. Anderson, R., Kuhn, M. Tamper Resistance—A Cautionary Note. 2nd USENIX Workshop on Electronic Commerce, 1996.
2. Anderson, R., Kuhn, M. Low-Cost Attacks on Tamper Resistant Devices. Preprint, 1997.
3. Camp, L.J. Reliability, Security, and Privacy in Electronic Commerce. Ph.D. thesis. Engineering and Public Policy, Carnegie Mellon University.
4. Camp, L.J., Sirbu, M., and Tygar, J.D. Token and Notational Money in Electronic Commerce. First USENIX Workshop on Electronic Commerce. July 1995.
5. Chari, S., Kaiserswerth, M., Rao, J.R. Network Security Issues in Pervasive Computing Devices. IBM Research Report RC 21592.
6. Chaum, D. Security without Identification: Transaction Systems to Make Big Brother Obsolete. Communications of the ACM, 28:1033-1044. October 1985.
7. Dierks, T., Allen, C. The Transport Layer Security Protocol. IETF Request For Comments 2246. Available online at `ftp://ftp.isi.edu/in-notes/rfc2246.txt`.
8. Hewllet Packard Laboratories Cooltown Appliance Computing. `cooltown.hp.com`
9. Hochberg, J., Smith, S., et. al. Kiosk Security Handbook. Los Alamos Unclassified Release LA-UR-95-1657, 1995. Los Alamos National Laboratory.
10. IBM Pervasive Computing, online at `http://www.ibm.com/pvc`
11. Kent, S., Atkinson, R. Security Architecture for the Internet Protocol. IETF Request for Comments 2401. Available online at `ftp://ftp.isi.edu/in-notes/rfc2401.txt`.
12. Kleinrock, L. Nomadic Computing & Smart Spaces. Keynote speak at Infocom 2000, Tel Aviv, Israel, March 2000. `http://www.cse.ucsc.edu/ rom/infocom2000/`.
13. The Mondex Electronic Cash Scheme. Documentation available online at `http://www.mondex.com`.
14. The Napster.com home page. `http://www.napster.com`.
15. National Institute of Standards and Technology. Security Requirements for Cryptographic Modules, Federal Information Processing Standards Publication 140-1. 1994.

16. The NTT DoCoMo i-mode applications. Documentation available online at `http://www.nttdocomo.com/imode`.
17. The Official Bluetooth SIG Website. Online at `http://www.bluetooth.com`.
18. The Palm VII handheld organizer. Documentation available online at `http://www.palm.com`.
19. Salonidis, T., Bhagwat, P., Tassiulas, L., LaMaire, R. Distributed Topology Construction of Bluetooth Personal Area Networks Preprint.
20. Satyanarayanan, M. Caching Trust Rather than Content. Carnegie Mellon University. Preprint, 2000.
21. Satyanarayanan, M. Fundamental Challenges in Mobile Computing. Fifteenth ACM Symposium on Principles of Distributed Computing May 1996, Philadelphia, PA Revised version appeared as: "Mobile Computing: Where's the Tofu?" Proceedings of the ACM Sigmobile April 1997, Vol. 1, No. 1.
22. Smith, S. Expressing and Enforcing Robust Behavior for Electronic Objects. The Federal Networking Council/MIT Internet Privacy and Security Workshop. May 1996. (Also: Los Alamos Unclassified Release LA-UR-96-1238.)
23. Smith, S., Weingart, S. Building a High-Performance, Programmable Secure Coprocessor. Computer Networks (Special Issue on Computer Network Security). 31: 831-860. April 1999.
24. Weingart, S. Physical Security Attacks and Defences. Cryptographic Hardware and Embedded Systems, August 2000.
25. Weiser, M. The World is not a Desktop. Interactions, Jan. 1994, pp. 7–8
26. The Wireless Applications Protocol Suite. Specifications available online at `http://www.wapforum.org`.
27. Yee, B.S.. Using Secure Coprocessors. Ph.D. thesis. Computer Science Technical Report CMU-CS-94-149, Carnegie Mellon University. May 1994.