I, Me and My Phone: Identity and Personalization using Mobile Devices

Riddhiman Ghosh, Mohamed Dekhil
Digital Printing and Imaging Laboratory
HP Laboratories Palo Alto
HPL-2007-184
November 8, 2007*

mobile authentication, bluetooth, personalization, retail

In this paper we present a framework for ubiquitous authentication using commodity mobile devices. The solution is intended to be a replacement for the proliferation of physical authentication artifacts that users typically have to carry today. We describe the authentication protocol and its prototypical implementation for a solution designed for the retail industry. We also propose a means of personalizing user-service interactions with embedded user-controlled profiles. A prototype of this solution has been built and demonstrated on the HP Labs Retail Store Assistant.

Internal Accession Date Only

Approved for External Publication

© Copyright 2007 Hewlett-Packard Development Company, L.P.

I, Me and My Phone

Identity and Personalization using Mobile Devices

Riddhiman Ghosh, Mohamed Dekhil
Hewlett-Packard Labs
1501 Page Mill Rd., Palo Alto, CA
{ riddhiman.ghosh, mohamed.dekhil } @hp.com

Abstract

In this paper we present a framework for ubiquitous authentication using commodity mobile devices. The solution is intended to be a replacement for the proliferation of physical authentication artifacts that users typically have to carry today. We describe the authentication protocol and its prototypical implementation for a solution designed for the retail industry. We also propose a means of personalizing user-service interactions with embedded user-controlled profiles. A prototype of this solution has been built and demonstrated on the HP Labs Retail Store Assistant.

1 Introduction

Personalization of content, services and user experiences is considered to be an effective method of fostering customer loyalty and building one-to-one relationships with users; it provides benefits both to the service providers and their customers [13]. Indeed attempts at personalization can be widely seen in online services such as the delivery of music, news, and in electronic commerce. This trend is not confined to the online realm alone however. When Chris Anderson, alluding to online marketplaces in The Long Tail [1] says, "the era of one-size-fits-all is ending", it is also what customers have come to expect in offline interactions and attempts at personalization are now also increasingly visible in brick-and-mortar environments. This, combined with the emerging trend of increased self-service options, has led to the creation of a new set of customer touch-points, typically in the form of kiosks. The HP Labs Retail Store Assistant [6] is one such example targeted at the retail domain; several other examples exist whether it is for shopping, or checking out books or buying movie tickets.

As a pre-requisite for many personalization techniques, or simply for authentication purposes, users have to first identify themselves to these touch-points. For users, assertions of identity—of *I-me-myself*—to a service can take various forms. Typically the inconvenience of entering a username/password combination on these service front-ends is avoided by giving users a physical token, such as a badge or card, with which they can authenticate themselves, following the familiar 'you hold this token so it is you' model. The problem presently however is that users have to carry a proliferation of these different physical tokens in their pockets for use with various services, e.g. multiple grocery store loyalty cards for different store chains, library cards for different libraries,

ATM cards, etc. Typically these tokens exist in various form factors such as RFID badges, magnetic stripe cards, barcode passes, etc. and are also service-specific, as a result of which they cannot be easily used across different services or be repurposed.

In this paper we propose a solution that intends to do away with this proliferation of physical identity tokens. Our principal contributions in this work fall into the following three themes:

- *Ubiquitous Identity Token:* We describe an approach to replace multiple physical identity tokens with a single Bluetooth-enabled ubiquitous computing device such as a user's cell-phone or PDA. Specifically, the BlueCard[1] is a device-derived certificate encoded with authentication and profile information, which is stored on a user's mobile device. This single token can be used with multiple services, and can be easily updated or repurposed in the future.

- *Addressing the 'Deployment Problem':* A significant problem with mobile devices is solution deployment due to the large diversity of device operating systems and runtimes—the platforms are varied, proprietary or obscure. While some phones support a Java Virtual Machine (Java ME) the VMs supported vary in flavor (e.g. PP or MIDP on CDLC or CDC) and version [10]. Still others may use Microsoft Mobile, Palm OS, Symbian OS, or the BREW (Binary Runtime Environment for Wireless) platform. This complicates deployment since one solution cannot effectively run on these different platforms without significant effort in re-designing and porting the solution to work with the specific capabilities of each platform. There are business issues that contribute to the deployment problem as well—for largely commercial (and arguably security) reasons most cell phone carriers in the United States have a "walled garden" approach when it comes to the software applications that can run on a phone and only carrier-approved software can be loaded. Thus for a custom software solution to work on most phones, one would need to negotiate with different carriers or device manufacturers. Our solution addresses the deployment problem and is designed in such a way so as to work on these diverse runtimes on common mobile devices (any Bluetooth phone with the Object Push Profile) without having to port and maintain independent code-bases for each of these different platforms.

- *Portable Profiles:* We also propose that the BlueCard be a portable way for users to carry their profile or preference information (or a subset thereof) along with themselves on their mobile devices so as to enhance user-service interactions.

2 Application Scenario

The application scenarios that we have focused on, and which have motivated this solution, are based in the retail industry. The HP Labs *Retail Store Assistant* [6] is a platform designed to enhance total customer experience in the retail environment and create a personalized shopping experience for customers. The primary delivery mechanism of these services for users is through an in-store kiosk, as shown in Figure 1, that is integrated with store-owned information systems, external web services and user-owned devices. Our initial focus is targeted towards this retail solutions platform. Consider a user who walks into a store, clicks a button on a kiosk and uses his phone to identify himself. The kiosk recognizes him, pulls his service preferences from his

[1] portmanteau of the terms Bluetooth and 'loyalty card'

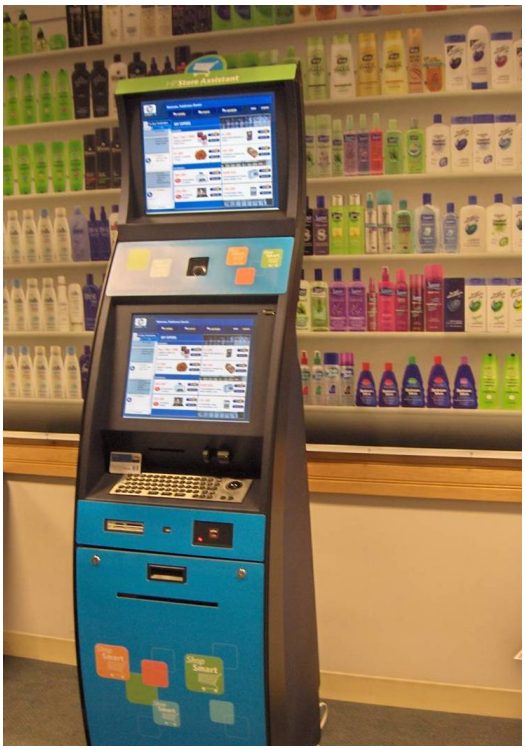

Figure 1: The HP Labs Retail Store Assistant kiosk.

phone and presents him with a personalized shopping list and product recommendations. Later if the user goes to a different chain of stores, or to the library to check out books, or to the video store to rent DVDs—all he needs to identify himself to these different services is his cell phone; he does not need to carry a bunch of ID cards or remember various combinations of usernames and passwords. We can state that the desiderata of a solution to be used in these scenarios are: a ubiquitous and repurposable authentication token that solves the mobile deployment problem, can aid in personalization and is easy to use with a familiar use model.

3 The BlueCard Approach

The BlueCard is a certificate which is derived from the hardware address of a mobile device and stored on the device itself. Similar to a X.509 digital certificate [7] which binds a public key to an identity, the BlueCard binds a *device* to an identity; the device is thus a first-class participant in the authentication transaction rather than merely a bearer of an identity certificate. The major components of the overall system are shown in Figure 2. The user-owned mobile device (P) is any Bluetooth enabled device such as a cell phone that supports the OBEX Object Push Profile (OPP) [4]. The OPP standard allows for the transfer of high-level objects between devices, and along with the headset profile, is amongst the most widely implemented of the Bluetooth specifications on cell phones. The service front-end (K), such as a kiosk as shown in Figure 2, is the touch-point to the customer and is the means of delivery of the services or content provided by the service and authentication back-end (S).

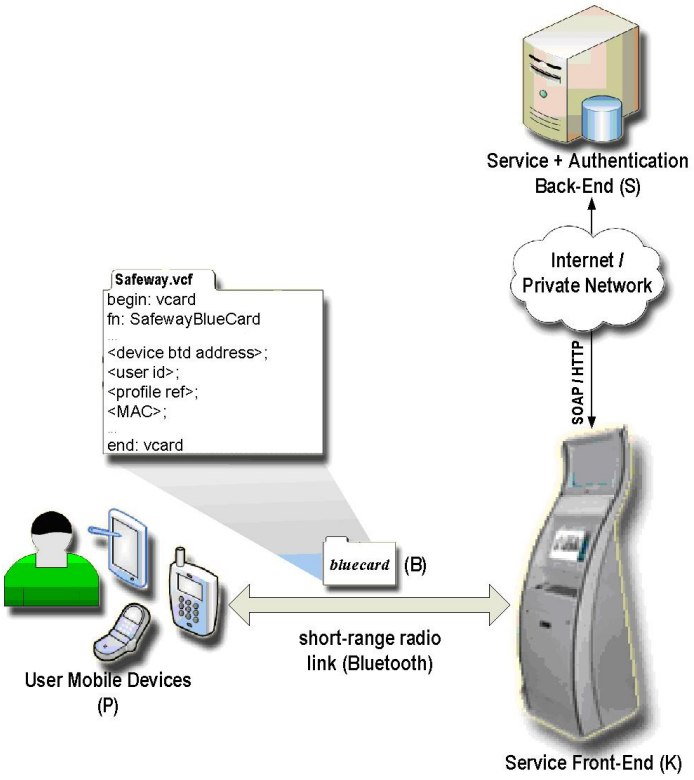

Figure 2: Overview and major components of the BlueCard system

3.1 Certificate Content and Format

The BlueCard (B) at a minimum encodes the Bluetooth hardware address of the containing device, an identifier of the subject owning the device and a message authentication code for tamper-detection of the contents of the certificate. Also optionally encoded is user profile information (or a subset thereof) of the subject, which can be used by the service. We discuss the benefits of this optional information later in the paper in Section 3.3.

Our choice of the certificate encoding format for the BlueCard is constrained by our requirement of designing a mobile-platform independent solution as regards to the deployment problem mentioned in Section 1. For instance, the storage, parsing, loading or transfer of a DER/PEM or XML encoded certificate from a device's persistent store would require a platform specific implementation for the device. However this software footprint on the device can be avoided if one is able to use a format for storing structured information that is natively supported by mobile devices. The vCard format (VCF) [15] is one such format that is natively supported by all OPP devices. VCF was designed as an open specification for personal data interchange and it specifies an interchange format for collecting and communicating information about people and resources (typically contact information). A VCF object is a structured collection of properties that may include not only information usually found on a business card such as names, addresses, telephone numbers but also other types of information describing the resource such as audio, graphical objects or geo-positioning data. One of the reasons VCF is widely supported on mobile devices like cell phones and PDAs is because the specification allows for a simple clear-text encoding. We have

chosen to use VCF as the persistence format for the BlueCard, and its contents are packaged into the existing fields of a VCF contact object, as can be see in Figure 2.

3.2 BlueCard Protocol

The BlueCard protocol defines the following two main operations between a user's mobile device and the service it wishes to authenticate with (via the service front-end):

- *One-time registration* of a mobile device for use with the service; this involves the creation of a new BlueCard.
- *Subsequent recurrent logins*, where the BlueCard stored on the device is used for authentication.

3.2.1 Certificate Creation

A graphical representation of the registration step of the BlueCard protocol is shown in Figure 3, along with the sequence of events and messages between the different entities involved in the creation of a new BlueCard. Typically an existing account/identity of a user with the service is to be associated with the new BlueCard. The user initiates the registration process at a Bluetooth enabled front-end such as a kiosk. The front-end then starts a discovery of its Bluetooth neighborhood by broadcasting 'inquiry' messages in order to locate proximal devices. If not already so by default, the user needs to put his mobile device in Bluetooth 'discoverable' mode to ensure that it can respond to 'inquiry' scans. (This is only required during the initial registration process, and the user's device need not be in discoverable mode during subsequent uses of the BlueCard.) On completion of discovery the front-end presents to the user a list of the friendly names [8] of the proximal devices that have been discovered and asks the user to choose his device from the list. Since the range of a Bluetooth receiver is typically around 10 m., devices in addition to the one the user is holding may have been discovered—the user is therefore asked to identify the device he holds. The user's claim that he indeed holds the particular device he has selected is then verified in order to prevent the user from (inadvertently or otherwise) selecting a device belonging to someone else. This is achieved by employing a type of challenge-response test where the front-end generates and displays a random challenge code (say a string of around 5 or more digits), and the user is then prompted to enter this code on his mobile device numeric keypad. The response that is transmitted through the mobile device is then matched with the issued challenge code for verification and thus only the user who is at the front-end and is able to operate the claimed device will be able to successfully complete this verification. The entire challenge-response process can be realized by leveraging the built-in bonding capabilities of the Bluetooth stack of the device and the front-end, with custom implementation only on the server-side (in this case the front-end) and not the user's device.

The service front-end then interrogates its Bluetooth stack for information about the user's mobile device, including its Bluetooth hardware address. This address (or a unique identifier derived from this address by combining it with other information) is used in the generation of the BlueCard, and is sent off to the service by the front-end. It should be noted that there are other alternatives to the front-end obtaining the hardware address of the user's Bluetooth device. For instance [14] describes an approach of using visual tags such as special barcodes encoded with the Bluetooth address, intended to be read by a suitable visual tag reader such as a camera, to

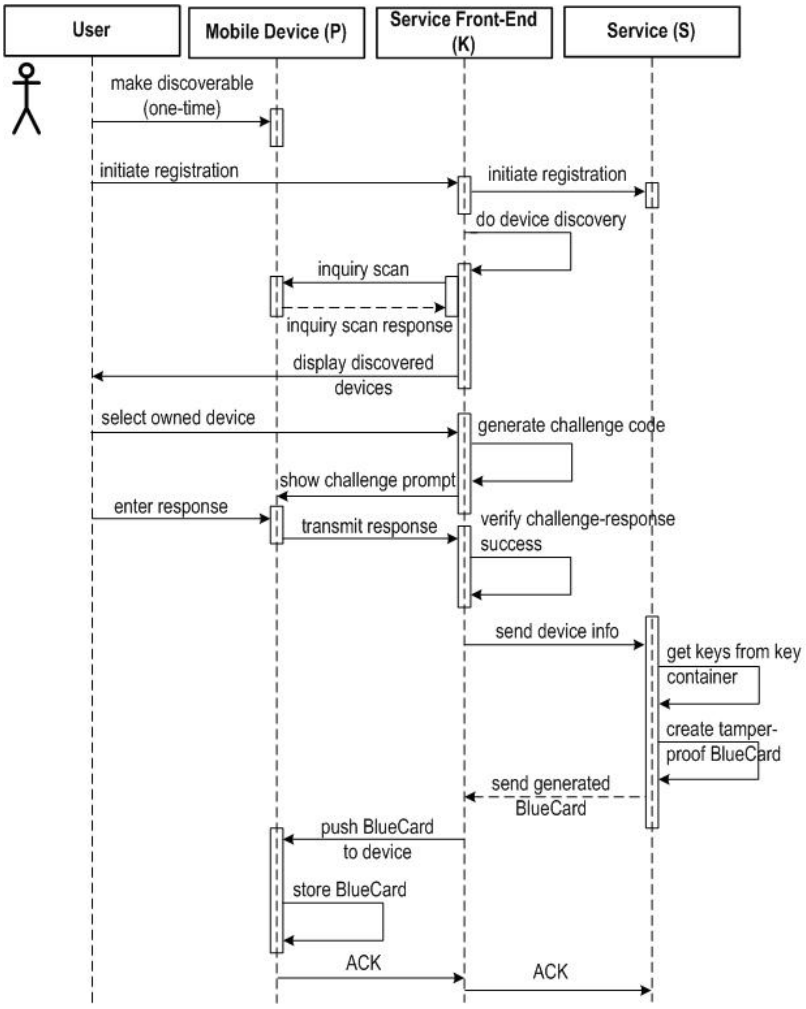

Figure 3: BlueCard Protocol — registration and certificate creation.

bypass the built-in Bluetooth discovery mechanisms. We chose not to adopt this approach as user devices cannot be used 'out-of-the-box' without affixing special tags, and would also need the use of suitable tag readers.

The service creates a tamper-proof BlueCard certificate in the VCF format as mentioned in Section 3.1, with the existing subject identifier of the user and the obtained device Bluetooth hardware address. The certificate is then "pushed" onto the user's device using the Bluetooth Object Push services.

3.2.2 Certificate Integrity and Authenticity

It is important for the authentication back-end to be able to both, ensure that the BlueCard has not been tampered with, and also confirm that it is the authority that generated the BlueCard and that its contents are authentic. This can typically be achieved by using digital signatures, with the signature embedded along with the content in the BlueCard itself to allow verification of integrity

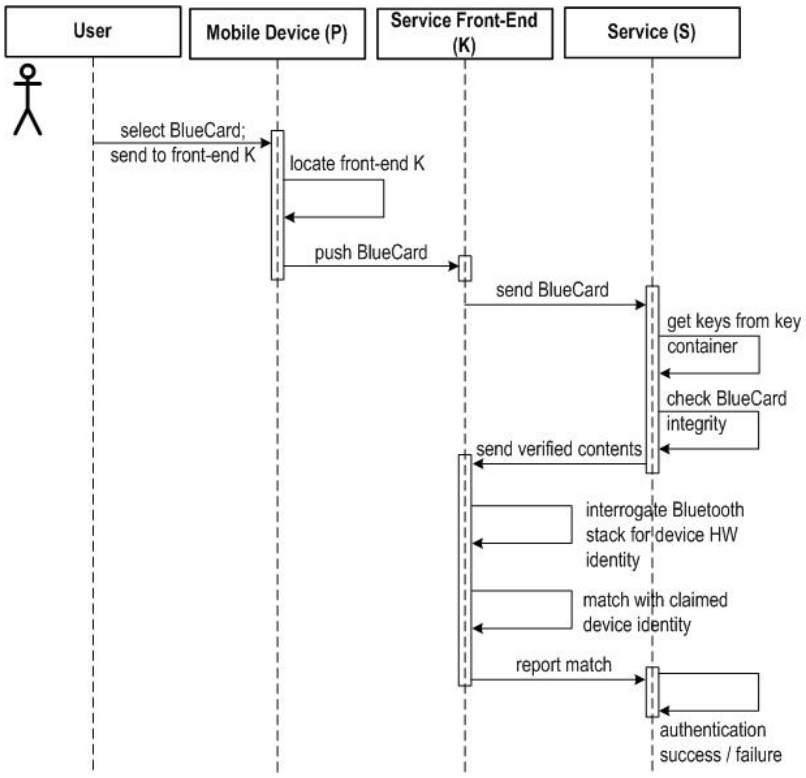

Figure 4: BlueCard Protocol — login with mobile device.

and authenticity. However there are implementation concerns with this approach. Consider the contents of a BlueCard that has been MD5 hashed (128 bit), RSA encrypted (1024-bit key) and base-64 encoded-the result will be a string 172 characters in length. This will need to be packed into an existing VCF field in the BlueCard. While the VCF specification does not prevent us from storing strings of this length or longer, we have seen through practical experience with different commodity cell phones and PDAs that many do not handle well strings of this length in a VCF field and truncate anything above a certain length, e.g. 50.

We therefore instead choose to embed Keyed-Hash Message Authentication Codes (HMAC) [9] in BlueCards to provide for integrity and authenticity checks. This results in 'signatures' that are much smaller instead (24 characters in length) and well within the practical VCF field length limits seen on commodity mobile devices. HMAC creation and verification rely on a shared secret; in our case the same logical authority that is generating the BlueCard is also the one that verifies it. For a BlueCard B, if B_{all} represents all fields except the HMAC B_{hmac}, then,

$$B = Concat(B_{all}, B_{hmac}),\ where\ B_{hmac} = HMAC(B_{all})$$

3.2.3 BlueCard Login

When a BlueCard is pushed onto a user's device it is stored as a regular VCF contact, and the user can access it just as he would any other contact. A mobile device may have more than one BlueCard for use with different services—they can be identified based on the name of the service

with which they are intended to be used (this is possible since the BlueCard stores the service name in one of the VCF name fields, such as the formatted name field FN). Figure 4 shows the steps and sequence of messages involved in a BlueCard login. The user uses his mobile device to send the stored BlueCard to the service via the front-end. While the shortcut keys and specifics of the user interface to accomplish this task vary between different mobile devices, they essentially involve selecting a proximate Bluetooth device (the kiosk / other service front-end identified by a displayed friendly name), selecting the service's BlueCard and pressing 'send'. On receiving the BlueCard the service checks it for integrity and authenticity. It then uses the information contained in the certificate, and matches it with the device information available from the Bluetooth stack. Since the certificate is bound to the device, authentication will succeed only if the registered device is used. Upon successful verification, the service uses the identity information contained in the BlueCard to successfully complete the authentication.

Compared to the one-time registration, login is a much simpler process (and much more frequent activity) with only one step required to be performed by the user—to send the service BlueCard. Most OPP devices have built-in shortcuts to achieve this effortlessly.

3.2.4 Pairing Considerations

Pairing (or bonding) in Bluetooth is a process where the two end-points of a Bluetooth link can agree to exchange data and form a trusted 'pair' by exchanging passkeys. This allows for a trust relationship to be maintained between the two devices should they wish to securely exchange data in the future.

However requiring users to pair every time they intend to login with their mobile devices is unreasonably cumbersome. This is especially true in the application scenarios envisaged for BlueCard, since it is unlikely a user will always login at a kiosk or front-end he has visited (and paired with) before. We have therefore designed the login protocol to not require paired connections, and users can interact and login with front-ends they have not encountered before without having to first pair with them. This makes the entire login experience simpler and faster. Note that a link protected with pairing passkeys provides us no advantage since a BlueCard, like an X.509 certificate, is public and does not contain information that needs to be encrypted. If the policy of a user's mobile device does not allow non-paired links by default or design, and if the user is unwilling to change this policy, he can perform pairing on-demand on encountering a service front-end for the first time.

3.3 Profile in your Pocket

We propose in this paper that the BlueCard be a portable means for users to carry their profile and preference information along with themselves on their mobile devices so as to enhance user-service interactions. Currently there are two approaches to storing user profiles [11]—they can be stored on the service-side, with users not having much control over this information, or stored entirely user-side, with the disadvantage that the information is not diverse or rich enough to be useful to a variety of different services. We believe in a combination of these two approaches, with users using the BlueCard to assert general preferences and information about themselves, which can then be combined with service-specific user profiles that are maintained at the service end. Given the storage limitations that exist on many mobile devices, the BlueCard stores profile information referentially, i.e. it contains a link to a user-authored and controlled profile document, retrieved by the service. The are multiple advantages of the profile-in-your-pocket approach: meaningful personalization can

Figure 5: Screen-shots of BlueCard: (a) Login using BlueCard on device. (b) Device discovery during registration. (c) Challenge-response verification during registration.

be offered by services with which the user has no established interaction history or user account; the user can adopt different *personas* while interacting with different services by changing the profile linked to by their BlueCard (thus allowing them assume a different role or persona for a particular service); the profiles are portable between multiple online and offline interaction touch-points. While there are several options for expressing BlueCard user profiles, we have been exploring the use of the FOAF (Friend-Of-A-Friend) format [5] (which is an open standard to describe a person and his interests, with tools support to easily create and maintain this information) and also the RUPO format (Retail User Profile Ontology—an ontology we have designed for capturing user profiles in the retail domain). A related approach in using FOAF for user-side personal profiles can be seen in [2]; however their focus has been on modifying the HTTP protocol to include additional headers so as to have GET requests to web servers transmit a user's FOAF file. The BlueCard can be seen as extending this approach to a personal mobile device, thus unifying personalization of online and offline user-service interactions. We have prototyped simple content customization scenarios using BlueCard profiles based on interests, location and group membership information.

4 Implementation

In this section we present some of the pertinent details of our prototype implementation of the BlueCard system, which has been integrated with the HP Labs Retail Store Assistant to provide for authentication and profile exchange, as shown in Figure 6. The 3 main entities we will discuss here are: the service front-end; the service and authentication back-end; and the mobile device.

The service front-end in the current prototype is a kiosk-machine running Windows XP, with the application written in the C# programming language on the .NET framework. All of the Bluetooth-related operations such as device discovery, service discovery, implementation of object push server and client capability and the bonding-based challenge and response functionality is implemented in C++ and exposed through a library. It uses the Bluetooth OBEX, RFCOMM and SDP interfaces provided by the OS/Widcomm running over the lower layers of the protocol stack. We have exposed the functionality of this C++ library to the front-end C# code using managed-unmanaged Interop based on the Platform Invoke Services of the .NET framework, with custom data marshallers written to marshal the Bluetooth protocol stack data structures. The front-end

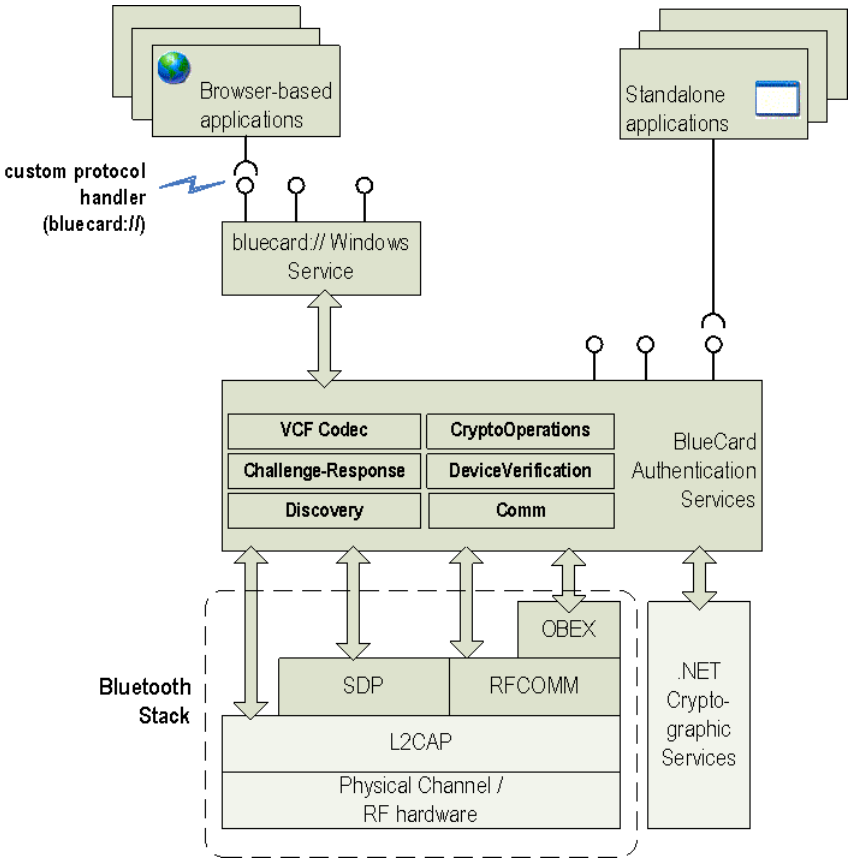

Figure 6: Overview of BlueCard implementation

communicates with the service and authentication back-end via SOAP-based web services.

Applications on the front-end such as a kiosk can be commonly written as both, a browser-based application, or a standalone application. To make the services of the BlueCard system easily available to web-based applications that are limited by the browser sandbox, we have written a custom protocol handler tied to a Windows OS service that handles the "bluecard://" protocol. This allows web-browser applications to place, for instance, login links containing "bluecard://", which then invokes our Windows service that executes the BlueCard registration or login protocols, as the case may be.

The service and authentication back-end is implemented as a collection of hosted SOAP web services through which it is accessible to the front-end. It is written in C# and utilizes the cryptographic support in the .NET Framework Class Library and extensions for certificate store and key management, and also for generating BlueCard HMACs using the HMAC-MD5 algorithm. The service component that utilizes the FOAF-expressed BlueCard profiles, is implemented as a web-service wrapper around a Java component using Jena—HP Labs' Semantic Web framework.

On the mobile-device side we can use any Bluetooth-enabled device, such as a cell phone or PDA, which supports the OPP standard. We have used BlueCard with multiple devices running different mobile runtimes such as BREW, J2ME and Windows Mobile. Figure 5 shows some screenshots during login and registration using BlueCard.

5 Discussion

In this section we will discuss the common threats we anticipate in the BlueCard system and our assumptions. The BlueCard certificate binds a particular user identity to a device. A tampering threat constitutes the modification of this identity-device tuple in the certificate. This is prevented by embedding an HMAC in the certificate and verifying it to guarantee integrity and authenticity of the certificate contents. Transferring the BlueCard intended for one device into another and using it to attempt login fails on account of the verification of hardware information of the device taking part in a run of the protocol—however this verification relies on our assumption that the operating system (or third-party) Bluetooth stack accurately reports link and device information. A compromised stack, on being interrogated about device information, may not respond faithfully. This threat maybe avoided by requiring the stack implementation to be cryptographically signed and to load only a trusted stack implementation on the service front-end (which we control). An impersonation threat exists during the BlueCard registration process, when a discovery of the Bluetooth neighborhood is done and the user is asked to select his device from the list of discovered devices—he may intentionally or otherwise select any other device from the neighborhood that does not actually belong to him. The protocol is designed to prevent this by issuing a random numeric challenge on the front-end user interface that the user is expected to enter on his device, and only if the response is accurately transmitted from the claimed device is the registration successful. That only a user who both, sees the random challenge and correctly enters the response on the claimed device, can actually hold the device, we believe is a reasonable assumption to make. Since BlueCard follows the familiar 'you hold this token so it is you' model another threat that exists pertains to the loss or theft of the user's mobile device. A common way to deal with this threat is to require a knowledge-based secret such as a PIN, in combination with the device, and the same applies for BlueCard. This is also useful if we wish to confirm the source of incoming data connections. This tradeoff between security and usability (requiring additional input) we believe is best left to the requirements of the particular application domain. For instance, in the retail domain loyalty programs, it uncommon to see knowledge-based input requested in addition to the presence of a physical token, while in other domains both are required. The BlueCard is suitable in both types of applications.

6 Related Work

Physical authentication artifacts such as RFID-, barcode- or magnetic swipe-badges are currently widely used; however the disadvantage for users using these schemes is that they have to remember to carry a special per-service token to authenticate, these are not repurposable, and this has resulted in a proliferation of ID badges in users' wallets. There exist mobile solutions for authentication that try to recreate the PC/browser experience at the device by asking users to visit special authorization web-pages on their cell phones. For instance [16] describes a system where a user receives a special URL using SMS on his phone, to which he then needs to navigate using his device WAP browser and allow or disallow a proxy to authenticate on his behalf. Most mobile browser interfaces are not suitable for complex interactions. These solutions not only suffer from a usability perspective, but also need a carrier connection for WAP / SMS access, for successful authentication. Similar SMS schemes are also used for two-factor authentication, where a special code sent through SMS has to be entered to authenticate, as in [12]. The disadvantage of SMS solutions is the loss of privacy—

users have to divulge their cell phone numbers to the service they are authenticating with, which can then be used in perpetuity by the service to send unwanted spam. Also SMS and network access over the carrier are typically fee-based and hence the user is essentially paying for every authentication transaction. Our solution is privacy preserving since it does not need revelation of a user's cell phone number, is based only on a proximity sensitive ad-hoc connection between a user's device and the service, and is also not fee-based.

Near-Field Communication (www.nfc-forum.org) enabled phones can be used as contact-less smart cards, and thus offer a means of authentication using mobile devices. In our solution the mobile device is not just a bearer of an identity certificate, but rather we use a device-derived certificate making the mobile device a first-class participant in the authentication transaction. NFC phones apart from being expensive are far from wide market adoption and presently very few companies manufacture or sell them. What is needed is a solution using currently widely available commodity devices.

Reference [3] describes a smart-phone based solution for physical access control. Their solution requires custom code to be running on the phone which complicates deployment on a wide variety of mobile devices given diverse device platforms. Moreover their solution requires multiple modes—camera, Bluetooth, SMS and MMS to work together for their distributed proving scheme for authentication. Our hardware and phone requirements are much more modest and can work on commodity Bluetooth cell phones with OPP support; we also provide service personalization support. For every login their solution requires a GPRS data connection since it requires the grantors and grantee to all be available and the authorization is done in a distributed fashion. Apart from latency and data communication costs, a scheme such as this where multiple parties are required for logins does not fit in with our application scenarios.

7 Conclusion

In this paper we describe BlueCard, a ubiquitous system for authentication and personalization using commodity cell phones. The authentication protocol relies on a device-derived certificate and is designed in such a way as to not require a custom software implementation on cell phones. We also propose a way for BlueCard to be used to personalize user-service interactions through user-controlled embedded profiles. We have presented a discussion of the anticipated threats and assumptions and also discussed our prototype implementation on the HP Labs Retail Store Assistant kiosk. We would like to further examine our solution from a usability perspective, conduct user studies to determine how the BlueCard user-experience can be improved, and explore other alternatives in BlueCard personalization.

References

1. Anderson, C., The long tail: why the future of business is selling less of more, Hyperion, 2006.

2. Ankolekar, A., Vrandecic, D., Personalizing web surfing with semantically enriched personal profiles, in Bouzid, M., Henze, N., editors, in *Proceedings of the Semantic Web Personalization Workshop*, Budva, Montenegro, 2006.

3. Bauer, L., Garriss, S., McCune, J., Reiter, M., Rouse, J., Rutenbar, P., Device-enabled

authorization in the Grey system, in *Proceedings of the 8th Information Security Conference* (ISC'05), Singapore, 2005, pp. 431–445.

4. Bluetooth SIG, Object Push Profile, in *Specification of the Bluetooth system: profiles*, 2001, pp. 339–364, available at http://www.bluetooth.com/Bluetooth/Learn/Technology/Specifications/

5. Brickley, D., Miller, L., FOAF vocabulary specification 0.9, available at http://xmlns.com/foaf/0.1/.

6. Hewlett-Packard Press Release, HP shows off system that affords every customer a personal shopper, May 2007, available at http://www.hp.com/hpinfo/newsroom/press/2007/070529b.html.

7. International Telecommunication Union, ITU-T recommendation X.509, 2005, available at http://www.itu.int/rec/T-REC-X.509/en

8. Kindberg, T., Jones, T., Merolyn the phone: a study of Bluetooth naming practices, in Krumm, J. et al., eds., *Proceedings of the 9th international conference on Ubiquitous Computing*, Innsbruck, Austria, 2007, pp. 318–335.

9. Krawczyk, H., Bellare, M., Canetti, R., HMAC: keyed-hashing for message authentication, IETF RFC 2104, 1997.

10. McCready, J., Integral Java: a single solution for bypassing the pitfalls of split stacks, in *Java Developers Journal*, 11(8), August 2006.

11. Mulligan, D., Schwartz, A., Your place or mine?: privacy concerns and solutions for server and client-side storage of personal information, in *Proceedings of the 10th conference on Computers, Freedom and Privacy: Challenging the assumptions*, Toronto, Canada, 2000.

12. Pullar-Strecker, T., NZ bank adds security online, in *Sydney Morning Herald*, Wellington, November 8, 2004.

13. Riecken, D., Personalized views of personalization, in *Communications of the ACM*, 43(8):27–28, August 2000.

14. Scott, D., Sharp, R., Madhavapeddy, A., Upton, E., Using visual tags to bypass Bluetooth device discovery, in *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(1), January 2005.

15. Versit Consortium, vCard: the electronic business card, version 2.1, 1996, available at http://www.imc.org/pdi/vcard-21.txt.

16. Wu, M., Garfinkel, S., Miller, R., Secure web authentication with mobile phones, in *DIMACS Workshop on Usable Privacy and Security Software*, 2004.