# Touch and Run with Near Field Communication (NFC)

Ben Dodson Hristo Bojinov Monica S. Lam
Computer Science Department
Stanford University
{bjdodson,hristo,lam}@cs.stanford.edu

## ABSTRACT

In this paper, we explore new ways in which Near Field Communication (NFC) can be used on smart phones. NFC allows for *contextual application invocation* (CAI)—the execution of code on the phone as a result of our environment. We can launch applications because of contextual information we learn from another transaction on our phone, or we can associate context with a virtual token to recall at a later time. We can also pass context from one phone to another so the devices can interact in a multi-party session. This paper presents a number of compelling applications using CAI and addresses the associated security and usability concerns.

## 1. INTRODUCTION

NFC is a radio technology that supports transactions at distances of a few centimeters. NFC is designed to support existing RFID transactions including contactless payments and some ticketing systems, as well as being a generally programmable platform. During a transaction, one party can be completely inactive, drawing power inductively from the active party. Even the active party draws little power and can be left on all the time with minimal effect on the phone's overall power draw. Also, the nearness of NFC transactions creates the possibility of using proximity as context and triggering an appropriate action almost instantaneously.

We envision widespread adoption of NFC in future generations of smartphones. The primary driver for the adoption of NFC on cell phones is contactless payments and ticketing. NFC, in the form factor of a credit card, has been used widely in Japan and Hong Kong for many years: for public transportation, vending machines, and convenience stores. Standards have also been created for "smart posters" [11]; posters, signs, and magazine pages can possess cheap, embedded data tags that contain information such as details of museum exhibits, transportation schedules, discount coupons, movie clips, or links to e-commerce sites. A third important use of NFC is for making connections between electronic devices—simply touching the devices together will configure them to connect over a longer-range protocol such as Bluetooth or Wi-Fi.

### 1.1 NFC on Smart Phones

Availability of NFC on smart phones presents an exciting opportunity for system and application designers, because not only can phones scan in information, but also programmatically generate new information to be presented for scanning. Furthermore, information received can be processed by the many available applications on the phone, facilitated by NFC's RTD architecture [12]. The ubiquity of mobile phones means that most consumers in the future will have access to this technology. The programmability means that many applications can be developed to facilitate peer interactions. They can communicate directly without requiring a third-party server. The effortless connection of NFC opens up many opportunities for the phones to be used to enhance physical social encounters.

This paper suggests many new and compelling mobile applications that can be built using NFC. With NFC, we can *touch* another phone (or any other NFC device) and we can *run* all kinds of applications without having to find the application of interest and painstakingly type in URLs or any other parameters. This is particularly important because we are often in a hurry on the go, which is distinctively different from how we use a PC. We are not sitting down; we do not have a keyboard; and we are always running out of time.

When we touch our phone with another NFC device, the other device provides the context that can be used to automatically invoke one or more applications on our phone with appropriate parameters. We refer to this invocation method as *contextual application invocation* (CAI).

### 1.2 NFC Applications on the Phone

Analyzing the many applications we came up with using NFC, we have identified three important classes of CAI:

*Transaction attachments.* There is a class of useful applications that can be classified as *attachments* to traditional NFC payment and ticketing transactions. The products we pay for or the events we attend provide the *context*, the device we connect with via NFC can supply us with additional information relevant to the context. For example, the bill associated with my payment can be transferred via NFC to my phone so that my phone can later submit it to my employer for reimbursement.

*Virtual tokens.* NFC can be used to replace various applications involving physical tokens: from getting a claim check for valet parking to getting loyalty cards from restaurants for attracting repeat customers. By using NFC on the phone, we do not have to worry about misplacing physical tokens; furthermore, these *virtual tokens* can be entered into our databases and tracked automatically. Here, the token grantor provides the *context* that defines the relevant interaction. We need to define a secure protocol that protects the interests of both the grantor and grantee of the tokens.

*Junctions*. Friends can also take advantage of NFC to have their phones interact in peer-to-peer multi-party applications. For example, people may want to play a peer-to-peer game, share their playlists, or exchange photos. It is simple and direct if we can just launch an application, touch our phone with our friends, and have their phone automatically run the same application (after user confirmation). To facilitate this class of applications, we propose the notion of a *Junction URI*, which provides the *context* necessary for a device to join a peer-to-peer application in progress. Because phones do not have static IP addresses, a Junction URI specifies a (secure) channel, consisting of the chat session on a rendezvous server and an ID for the session. From the Junction URI, a device can also find out where the application can be downloaded.

## 1.3 Contributions

This paper describes and analyzes a large number of novel applications made possible by integrating NFC into the smart phone. We classify these applications into three categories depending on the kind of contextual application invocations (CAI) used: attachments to transactions, virtual tokens for replacing physical real-world tokens, and junctions for connecting arbitrary peer-to-peer applications on mobile devices. We have created a Junction programming platform and have created prototypes of a large class of multi-party applications on the platform including profile exchanges, games, and collaborative playlists.

Contextual application invocations can potentially be dangerous too; we need to prevent malware from triggering unwanted applications and actions on our phone through NFC. This paper discussed security measures to protect against security attacks on various CAI methods, including a use of NFC itself to protect the loss of our phones.

The organization of the rest of the paper is as follows. Sections 2 to 4 describe the three kinds of CAI: transaction attachments, peer-generated virtual tokens, and the Junction platform. Section 5 describes how we secure the phone using NFC itself. Section 6 discusses related work and Section 7 concludes.

# 2. TRANSACTION ATTACHMENTS

NFC was designed to interoperate with existing deployments of near-field radio technologies, including contactless payments and access to public transit systems. Moving these transactions to the phone may help reduce the number of things a person carries, but there are other more significant benefits.

We can improve the usability of ticketing for a public transit system by using the phone as our pass. The connectivity on the phone allows us to purchase the pass from anywhere, without waiting in line at a kiosk. We can also see how many rides we have available or how much credit is left in our account from anywhere. All the while, we can still swipe into the transit system quickly and also verify our ticket to a conductor onboard. [7]

We can improve the security of credit card transactions by moving the contactless payment to an active, programmable device by supporting one-time use credit cards. One-time use credit cards are tremendously useful for reducing credit card fraud—instead of giving a merchant our credit card numbers, we can request our bank to give us credit card numbers that can be used only once. So far the cumbersome procedures required to get single-use cards have limited adoption. With NFC on a phone, users can run an application that stores one-time credit card numbers securely and easily, and the application can present these one-time numbers to merchants, on behalf of the user. Users don't have to know about the added layer of security, using their phone to make payments as they would a contactless card. The phone may negotiate several one-time use numbers in advance so that payments can be made with the phone offline.

With these applications moved to the phone, we can further enhance the mobile experience by leveraging the contextual information gleaned from them. We first describe several such applications, then discuss the security considerations for NFC transactions.

## 2.1 Applications

**Receipts, reimbursements, and money management**. As an add-on to contactless payments, we imagine the transaction results in a receipt being sent to the user's phone. The receipt may be transmitted as part of an enhanced standard for contactless payments, or may occur as an additional transaction during the same NFC scan. The phone keeps a local database of transactions and receipt objects, and allows programmatic access to them (with appropriate security restrictions). This will enable, for example, an application for managing receipts.

Another application can help file reimbursement claims. After a business trip, a user could select purchases from a list of gathered receipts over some span of time. With a few clicks, she can email this list to file a reimbursement claim. The receipt data is stored privately on the phone, and is only released at the user's discretion.

**In Situ check-ins**. Check-in services such as foursquare and Facebook Places have grown in popularity. If a user makes a payment at a restaurant, for example, the phone can receive details about that user's whereabouts. If a user labels an establishment as a "favorite," the check-in may occur automatically, or an application can make the check-in available with the press of a button. Using NFC for contextual awareness is a much lower-power solution than using GPS and is also more accurate. We also imagine dedicated NFC tags that a business may put out explicitly for making check-ins easy.

**Reviews**. Our phones will be able to determine the products we buy, the restaurants we visit, and the movies we see. The data can be kept privately, and applications can request permission to view different classes of data. If the user has a movie application installed, it may request access to movie-related events from the user's activity stream. This allows the user to plug into any of her favorite sites.

**Sporting events**. We can use our NFC-enabled smart phones as a ticket for entry into sporting events. After scanning in, the phone launches an application associated with the event. It is loaded knowing the user's seat, and can be used to order concessions for delivery. Payment can occur through the application as well for a smoother user experience. The application can also better connect the user to the

event, providing video replays and letting them interact with events occurring on the venue's big screen, such as trivia, polls, or shout-outs.

**Public transportation**. An NFC device can be used to access a public transportation system, be it train, bus, or subway. Again, scanning into the system can invoke an application. This program can provide the user with a real-time schedule, customized to their current stop, and can alert the user when their destination nears.

## 2.2 Security of NFC Transactions

Security threats in current uses of NFC are well understood from similar applications in areas like content distribution (DRM), web browsing, and networking. Here we discuss techniques and principles to provide security in NFC-based applications.

### 2.2.1 *Preventing unauthorized ticket sharing*

In the case of electronically presented service "tickets", such as in public transportation or sports events without assigned seating, we have to ensure that users can not share their benefits with other parties. Consider the case of Shawn who has a ticket to watch the San Jose Sharks. Shawn decides to share his ticket with a friend: he beams the contents of the ticket over to Sara's smartphone, and now both of them can present a valid token at the entrance.

The means of dealing with unauthorized sharing depend heavily on the level of protection desired. At one end of the spectrum, a centralized database can keep track of used tickets at the venue, and a ticket becomes invalid once presented. Either Shawn or Sara can get in, but not both of them. Optionally, the ticket can be made valid again in case the owner exits the venue, to allow re-entry. Note that if transfer of ownership must be supported, then centralized tracking of ownership has to be in place from the time of ticket issue, all the way to the time of use.

A less centralized solution involves tickets that are tied to a specific person: at the time of ticket generation, the issuing authority uses a private key to sign the ticket along with a photo of the authorized owner. The benefits of this approach extend to long-term tickets that can be used multiple times (such as commuter rail permits or ski passes). Finally, in situations where long-term permits are not visually checked (such as in high-traffic areas like the subway), data mining can be used to verify legitimate use and flag suspicious cases for examination or even revocation.

### 2.2.2 *Securing Contextual Application Invocation*

The smartphone reading passive content over NFC will be a dominant mode of interaction. Several security principles underpin such operations both for the purpose of contextual application invocation. These principles derive from the analogy to browsing on the Internet and following links: we learn to be careful when navigating to unknown domains, and modern browsers offer much assistance in helping us make the correct decisions.

The content scanned must be assumed insecure and should not be trusted in any critical or expensive actions. Specifically, data received over NFC should not lead to arbitrary behind-the-scenes activities, such as sending SMS or performing a phone call [10]. Rather, every scan should either lead to no "side effects", or such effects have to be explicitly described to the user prior to taking the action to give her an opportunity to verify the action taken (e.g. confirm the outgoing call or SMS, or verify that the domain being browsed is correct before navigating to it [3]). This of course extends to loading and running applications which require any privileges (access to private data, Internet access, and so on). Although visual cues can be provided to a user that the scanned tag has not been tampered with, ultimately trust in the content must start with a signature from a known entity.

### 2.2.3 *Man-in-the-Middle Attacks*

With NFC, we must watch out for the possibility for an attacker, a third party with an active tag, to inject itself in the conversation and modify it to his advantage possibly even without being noticed. While peer certificates can go a long way towards excluding third parties from an exchange, they will never be the complete answer: certificates can be obtained fraudulently, or perhaps with an apparent owner which appears to be legitimate, but is not (such as using a slightly misspelled version of the legitimate owner). Because of this, it is imperative that interactions be designed with multiple safeguards: verification based on cryptography, as well as user verification and common sense (e.g. when confirming a payment, there should not be two or more simultaneous payment requests from different payees, Figure 1; or, when payment is confirmed but the service is still unavailable, assume fraudulent use—the payment went to the wrong destination—so the user should investigate).

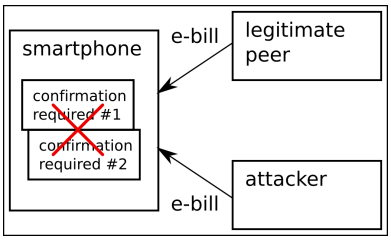


Figure 1: Security-aware interaction design. Two simultaneous e-bills (a very rare occurrence) are flagged and assumed malicious by the payment application.

### 2.2.4 *Preventing relay attacks*

In a relay attack the authentication protocol is bridged, such that authentication no longer requires physical proximity [4, 5]. Users transacting unique low-cost objects (such as people presenting movie tickets at the entrance) are particularly vulnerable to relay attacks. On the one hand, the low value of the transaction makes an interaction-free approach more acceptable. On the other hand, if the object owner is willing to publicly share the object, then she becomes vulnerable to malicious relaying of the ticket and involuntarily granting entry to an attacker. While relay attacks can be prevented by distance bounding [13], the technology is still in its infancy: a simpler approach could be to give ticket owners a choice between security (user confirmation required to use the ticket) and convenience (the ticket is presented automatically). This behavior could adjust based on context: the ticket management application can decide whether it is safe to present a token without asking the user—based on the device location and past history of fraud at that location.

# 3. PEER-GENERATED NFC TOKENS

Programmable NFC can be used in place of physical tokens that we use in our daily life. Besides not having to keep track of little pieces of paper, there are other benefits when these NFC tokens are better integrated into our digital systems. In the following, we will describe the applications, summarize the requirements, describe the platform, and discuss the security and usability issues.

## 3.1 Applications

**Receipts, physical objects represented by virtual tokens: dry-cleaning, valet parking, luggage, coat check**. Receipts or tokens are often handed to the customers as they bring in their dry-cleaning to the cleaners, leave their car with the valet, leave the luggage with the hotel bell hop, or check their coat at a museum. Instead of pieces of papers, NFC can be used to give the customers a receipt, which they can use to claim their items upon their return. In this way, customers do not have to worry about misplacing the receipts. More importantly, customers can submit these tokens remotely over the network before they arrive in person so that these objects can be retrieved ahead of time. Here it is important that the tokens are not forgeable.

These tokens can also be used as signatures from grantee to grantor. Users can sign for packages by tapping their phone to a delivery person's portable kiosk, adding improved verifiability to the system. With a suitable system of security, we can also notarize digital content with the tap of a device.

**Trading virtual tokens for physical objects: coupons and awards**. Here, users can hand in their virtual awards earned in a game for physical awards. Happy Feet is an example of a mobile health application[14] that encourages users to exercise by handing them badges for miles logged; these badges can be turned in at local restaurants sponsoring the badges using NFC. If these badges are just coupons, rather than free products, then it is necessary to ensure that these badges are consumed and cannot be re-used.

**Granted identities: loyalty programs**. Stores and restaurants may want to increase repeat customers by instituting a loyalty program that grants customers a discount after some amount of purchases. Using NFC, a store can easily give their customers a unique ID and later record their purchases. By having it integrated into the database, the store can collect customer profiles which can be extremely valuable. The ease of NFC transactions and the elimination of the hassle of keeping a card can greatly increase the adoption of loyalty programs.

## 3.2 Platform for Peer-Generated Tokens

Users cannot be expected to install separate mobile applications for each service provider they interact with. We propose the notion of a rich token manager for handing tokens. The token manager maps a service (a particular restaurant, dry-cleaning, etc.) to a token. Given a service identifier, a user will typically have at most one matching token in his database. Each token has a globally unique identifier generated by the grantor, such as a random 128-bit number. In addition, the token incorporates a creation timestamp and the creator's service identifier—the grantor certificate (Figure 2). The token will not have personal identifying information associated with it by default, although it may be reissued with additional data attached as the grantor and grantee interact over time. All token contents are signed by the grantor, and the signature is verified by the receiving party.

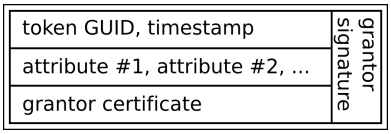


Figure 2: The structure of a peer-generated token.

The user experience is as follows. First, the user unlocks her phone. She then touches it to the grantor's station, be it kiosk, mobile phone, or otherwise. The phone learns of the grantor by its known service name and checks the phone's database to see if the user has an existing token. If so, the token is presented to the grantor, as long as the user has allowed automatic submission. Otherwise, she is prompted to approve the action. Finally, if the user does not have an existing token, the grantor can create one; the new token will be unique and can be coupled with other personal information at the user's discretion.

At the protocol level, the following steps take place: first, the two communicating parties perform mutual authentication (Figure 3). This step is similar to SSL session setup using client-side certificates: at the end, both sides have verified each other's stated identity (note that this identity is not guaranteed to be related to any real-world entity—it is only used as a reference point in future interactions). In the second step, one side—typically the service provider—requests its peer's token. At that point the token holder needs to locate the matching token, and decide whether it will present it: the decision might involve a number of contextual clues such as location, type of the token, and amount of personal information associated. Third, the token is transmitted back to the requester, or otherwise the response indicates that no token is available. Finally, as an optional fourth step the requester can issue a fresh token.

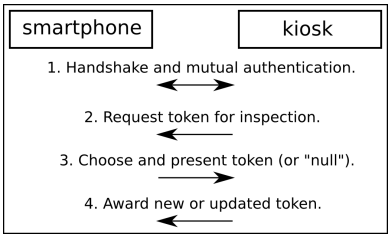


Figure 3: High-level flow of a peer-to-peer token transaction.

For most private peer-to-peer transactions, such as receiving a claim check for physical objects and receiving those objects after presenting a claim check, interaction over NFC is similar to that using physical tokens. The claim check for a car is the proof of receipt token signed by the valet desk. Claiming the car back involves the user's phone signing a proof of return token, which references the proof of receipt. In fact, electronic token handling offers opportunities to enhance the security of interactions: for example the proof of receipt can include the vehicle owner's photo, making sure somebody who stole the phone can not retrieve the car.

The technique of incorporating owner photos (or one-way

hash functions of photos) in tokens also helps when it is desirable to prevent multiple users from claiming the same service, such as in the case of digital coupons—in the virtual world, coupons can be issued individually such that only someone matching the photo referenced in the coupon can claim them.

# 4. P2P APPLICATIONS WITH JUNCTION

NFC is useful to introduce two peers so they can interact online. By allowing peers to exchange a session-specific secret, we can enable all forms of peer-based interactions, without having to be monitored by third-party servers. We have created Junction [8], a platform to support such interactions.

NFC allows two devices to interact simply by placing them together. However, the nearness requirements for NFC would not lend itself well for comfortable interaction in a multi-party application session. Instead, we use NFC as a tool for bootstrapping multi-party applications, with the application session running over another channel.

Imagine Alice wants to play a multiplayer dice game called Bluff with her friend Bob, which is played across two or more phones. Alice opens her phone and browses to her Bluff application. The application instructs her to tap the phones to start a game. Bob takes out his phone, unlocks it, and they touch their devices together. Bob hears a beep from his phone, indicating the NFC transaction worked. His phone asks him if he'd like to join the game of Bluff, which he does. Had Bob not had the Bluff application installed, his phone would prompt him to download and install it. After the installation, another tap of the devices launches the game.

In our example, only one device needs to explicitly launch the application. The other phone is given enough contextual information to locate the appropriate program and join the existing session. To avoid unwanted applications from launching on a device, we require the phone to be unlocked prior to the NFC transaction, and also prompt the user when we detect a joinable session.

We have created Junction to foster the creation of such multi-party applications. Junction includes client-side libraries for application development, as well as infrastructure supporting their connectivity at runtime. Junction applications do not have a central server, instead using a *switchboard* service that simply routes messages between devices, but does no application-specific computation.

Junction development is session oriented. A session supports multiple participants, with an activity uniquely represented with a URI such as:

```
junction://sb.openjunction.org/un1qu3?key=s3cr3t
```

This URI expresses four things. The scheme "junction" indicates to a host platform that the URI should be recognized as a Junction session. "sb.openjunction.org" is the switchboard that is hosting the particular activity session, with session identifier "un1qu3". The URI also contains a key ("s3cr3t") that is used to encrypt communication. Because this URI is never seen by the switchboard, messages can be encrypted between peers without the switchboard knowing their contents. The Junction protocol also includes a mechanism for looking up the activity's details. Most importantly, an activity can have supporting code on a number of different platforms, and the protocol allows a device to locate the codebase it requires.

Since NFC is not yet commonplace on smartphones, we have been using QR codes to exchange Junction URIs. The user experience is more cumbersome—to join a session, a user must launch the QR code scanning application and take a photo, a process that can take ten seconds or more. With NFC, the exchange of session information is nearly instant.

Because of the simple URI representation of sessions, integrating Junction with NFC is simple. One device emits a tag, with its content being the session URI. The other device scans the phone and handles the URI, which is opened with the Junction application installed on their phone, which in turn launches the appropriate peer-to-peer application. Our Junction applications that have been using QR codes need little to no modification to support NFC.

We have built several applications on top of the Junction platform, including:

- Bluff, a dice game for two to eight players (Figure 4).
- weHold'Em, a version of Texas Hold'Em poker played between Android phones and a TV.
- weMeet, exchanging contact information between two phones.
- wePix, share photos between mobile phones and a TV.
- weTunes and weTube, share music and videos between Android phones and a TV.
- weDraw, a collaborative whiteboard between Android phones, iPhones and iPads, and web browsers.

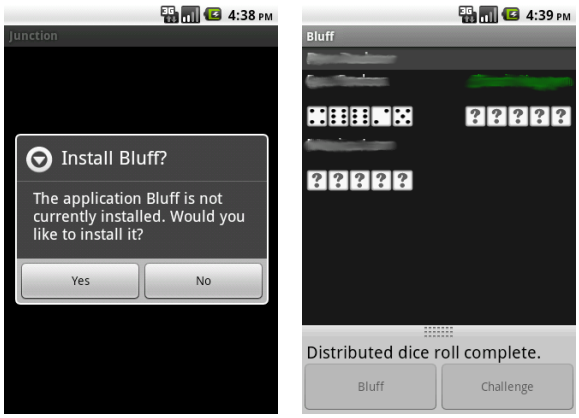


Figure 4: An Android device launching a game over NFC.

# 5. DEALING WITH DEVICE LOSS

With the phone holding increasing amounts of private, as well as financial data, loss of the device is a top concern. Password authentication has been the de facto standard for securing user data, yet passwords do not work well on smartphones: small or touch-based keyboards and frequent unlock events make entering a password every time a nuisance. The alternative is to build a hardware authenticator, in the form of a ring, wristband, or key, which can unlock the user's phone without requiring any interaction. A stolen device will be impossible to unlock without the matching authenticator.

Even though hardware authenticators have existed for a long time, they have failed to become ubiquitous because of their cost as well as complexity of deployment. One current example is the RSA SecurID, a small keyfob device which provides

a stream of unique numbers that the user can type along with her password when authenticating to remote services. The server verifies that the supplied number—unguessable by an attacker—matches what is expected. The cost of one authenticator is more than $10 per year, with the added expense and hassle of installing a central authentication server. With wide adoption of NFC, and the low cost to manufacture passive or semi-passive tokens, we envision that for the first time hardware authentication can become pervasive.

In a basic scenario, hardware authenticators can serve as keys for unlocking smartphones. A contactless, passive (no battery) keyfob can perform a challenge-response protocol with the smartphone over NFC. The authentication happens on boot as well as screen unlock events. After the authentication, the keyfob provides to the smartphone a symmetic cryptographic key necessary to unlock data storage on the phone—this key is only stored in volatile memory on the phone, and erased on screen lock or shutdown. Phone loss or theft are now reduced to an inconvenience: without having physical possession of the authenticator, an attacker can no longer access any private data on the phone. Furthermore, without the authentiactor the phone can lock up and become unusable, discouraging theft in the first place.

The NFC authenticators we envision in the future will be able to assume multiple identities, performing challenge-response authentication with different types of peers—electronic devices, online services, and physical-world infrastructure. Some credentials can work automatically: devices that the user carries (such as the smart phone itself) could be unlocked by the mere presence of the NFC token at the time of use. In other cases, the push of a button may be required to prevent relay attacks. Such attacks are a possibility when users unlock assets that are usually away, such as vehicles or buildings. Hardware authenticators must have dedicated buttons to confirm the unlocking of such assets to prevent car theft or home burglary while the owner is away.

# 6. RELATED WORK

Much research has been done to bring interactions between mobile phones and physical spaces. Broll et al. explore a platform for running generic web services on phones, invoked by NFC or other proximity cue. [1] The focus is on supporting web services on phones, and in our paper we explore other context-aware platforms for mobile interactions.

The MIT Mobile Experience Laboratory demonstrates several real-world uses of NFC in phones, including making payments and pairing devices for peer-to-peer games. [9] Ghìron et al. explore a system for virtual ticketing on an NFC-enabled phone. [7] We expand on these ideas to associate a contextually-driven platform to the transactions, supporting a new variety of applications.

With NFCSocial, Fressancourt et al. explore the use of NFC to update presence information in a social network. [6] NFC-Social leverages the ability of an NFC transaction to invoke an application on a phone, and apply it to social networking check-ins. We expand on the idea to support in situ check-ins, taking location-aware cues from other NFC transactions such as payments.

Junction uses NFC to launch programs across multiple phones. Similarly, Bump Technologies has built an API to support communication between two devices after they've "bumped" together. [2] Their approach requires a matching algorithm running in the cloud, and all subsequent data flows through a central server managed by Bump. We believe the many applications built with Bump demonstrate the utility of NFC as a method for bootstrapping cross-device applications, and that widespread adoption of NFC would replace the need for this cloud-based interaction. Also, NFC supports a better user experience, since one device can scan another without first launching a special application.

# 7. CONCLUSIONS

In this paper, we have presented our vision of how smart phone applications will change when NFC becomes commonplace. NFC will allow what we term contextual application invocations. Applications can be invoked as a side-effect (attachment) of another transaction that provides it meaningful context. Applications can also be launched to exchange tokens, with our phones responding to the context of the token grantor. Finally, one phone may provide context to another to create a junction between them, allowing them to partake in a cross-device activity. We have implemented the Junction platform and written several applications for it, demonstrating the usefulness of programmable NFC on smart phones.

# 8. REFERENCES


[1] G. Broll, S. Siorpaes, E. Rukzio, M. Paolucci, J. Hamard, M. Wagner, and A. Schmidt. Supporting mobile service usage through physical mobile interaction. In *In Proceedings of PerCom 2007, White Plains*, pages 262–271. IEEE Computer Society, 2007.

[2] Bump technologies. http://bu.mp.

[3] R. Dhamija. The battle against phishing: Dynamic security skins. In *In SOUPS 2005: Proceedings of the 2005 symposium on Usable privacy and security*, pages 77–88. ACM Press, 2005.

[4] A. Francillon, B. Danev, and S. Capkun. Relay attacks on passive keyless entry and start systems in modern cars. Cryptology ePrint Archive, Report 2010/332, 2010. `http://eprint.iacr.org/`.

[5] L. Francis, G. P. Hancke, K. Mayes, and K. Markantonakis. Practical NFC Peer-to-Peer Relay Attack using Mobile Phones. In *Workshop on RFID Security – RFIDSec'10*, Istanbul, Turkey, June 2010.

[6] A. Fressancourt, C. Herault, and E. Ptak. Nfcsocial: Social networking in mobility through ims and nfc. *Near Field Communication, International Workshop on*, 0:24–29, 2009.

[7] S. L. Ghiron, S. Sposato, C. M. Medaglia, and A. Moroni. Nfc ticketing: A prototype and usability test of an nfc-based virtual ticketing application. *Near Field Communication, International Workshop on*, 0:45–50, 2009.

[8] Junction. `http://openjunction.org/`.

[9] MIT Mobile Experience Lab. A day at mit with near-field communication. http://techtv.mit.edu/videos/1369-a-day-at-mit-with-near-field-communication.

[10] C. Mulliner. Vulnerability analysis and attacks on nfc-enabled mobile phones. *Availability, Reliability and Security, International Conference on*, 0:695–700, 2009.

[11] NFC Forum. Smart poster record type definition technical specification. 2006.

[12] NFC Forum. Generic control record type definition technical specification. 2007.

[13] K. B. Rasmussen and S. Čapkun. Realization of rf distance bounding. In *Proceedings of the USENIX Security Symposium*, 2010.

[14] H. F. D. Team. Happy feet: The social way to walk, jog, run, cycle, or even... ski., 2010. `http://happyfeet.heroku.com/`.