Verifiable Credit Based Transfers in Wireless Ad Hoc Networks

Bogdan Carbunar, Brett Lindsley, Michael Pearce and Venu Vasudevan
Motorola Labs
Pervasive Platforms and Architectures Lab
Schaumburg,IL 60196 USA
{carbunar,brett.lindsley,michael.pearce,cvv012}@motorola.com

Abstract

Encouraging cooperation between users of mobile devices operating in ad hoc mode is a difficult task mostly because the scarce battery and bandwidth resources of devices suggest that selfish behavior may be most beneficial. The insecure usage of credits to reward cooperation can easily provide an incentive for cheating, thus, on the long term only leading to selfishness. In this paper we propose several secure credit based mechanisms enforcing fairness in a hybrid wireless content retrieval system operating both in cellular and ad hoc connectivity modes. Our solution consists of mechanisms for securely and privately discovering desired content on neighboring devices, simultaneously exchanging credit and content shares in a verifiable manner and for generating and expiring non-forgeable credits. We present experimental results of a partial prototype of our system implemented on MPx and E680i cellular phones and HP iPaq hx4700 PDAs, along with extensive simulation results showing that our solution significantly reduces the effectiveness of selfish behavior, making it an unattractive strategy.

1 Introduction

There is an explosive demand for portable media products that provide mobile users with access to their personal/commercial media "on the go". Products such as ipod, iRadio and ROKR show how the traditional concepts of jukebox, radio programming and DVR are being enhanced in the mobile media context to provide greater variety, portability and choice. The media and storage capabilities of cellphones combined with their global penetration, make them a promising "converged" media device that combines both a networked media player and a communications device. If done right, such a device can provide the user with transparent access to a large, distributed media library of both stored (e.g. media files on your PC) as well as dynamically recorded content (e.g. tv programs recorded on your home DVR), and a compelling playback experience. A networked player goes beyond portable media players such as the ipod video in leveraging the network to providing timely access to dynamic content (such as news, shows, or sport events).

Building a cost-effective networked media player that supports dynamic content markets presents a challenge - that of being able to pro-actively and cheaply pre-cache media so as to provide users the illusion of a large "virtual" media library on a device that has a large (order of 1 GB) but not infinite storage capacity. Naive usage of the cellular network can rack up large cellular bills and drain the battery, thus saddling the user with a large wireless bill while degrading the communication function of the device. A more elaborate solution could take advantage of the existing alternate, cheaper 802.11b [4] based communication capabilities of devices (present on Motorola MPx or CN620, Nokia 6136 or 9500 Communicator, Siemens SX66 or Fujitsu Siemens LOOX T800 series cellphones to mention just a few) in order to complement cellular service whenever Wi-Fi hotspots are available.

While hybrid use of cellular and Wi-Fi networks can reduce the operational expenses of participants, user mobility coupled with insufficient access point deployment provide significant space for improvement. An interesting attempt at filling this gap, requiring no infrastructure changes, consists in using communities of iRadio [2] type service users, where participants use their wireless 802.11b interfaces to search for and transfer desired content from each other, in cases where access points are unavailable. However, the functionality of proximity based wireless peer-to-peer communities relies on the willingness of users to cooperate, that is, to consume resources such as battery and bandwidth. In such instances, individual reasoning suggests selfishness as the most effective strategy for maximizing personal satisfaction, at the expense of global welfare. Selfish behavior

1-4244-0910-1/07/$20.00

may not only reduce the levels of satisfaction achieved by participants but more importantly, reduce the trust of cooperating participants in the system and ultimately affect their willingness to cooperate.

A popular mechanism for encouraging cooperation consists in rewarding resource consumption with credits. Participants share their content with others only when offered credits in return and earned credits can later be used to enable similar transactions benefiting them. By regulating transactions, credits discourage selfishness, however, the use of insecure mechanisms for managing credits may generate malicious behavior, such as repeatedly spending the same credits, fraudulent minting of credits or cheating during transactions.

In this paper we study mechanisms for detecting malicious behavior and identifying attackers in the context of the Cache-and-Carry (CnC) system [8] developed in our lab (see Section 3 for a detailed description). CnC is designed to operate on Wi-Fi equipped cellphones and enables their users to efficiently retrieve desired content from their home computers through a multitude of connections (USB, cellular, Wi-Fi hotspots). We extend CnC to support ad hoc transfers of content from nearby devices in exchange for credits. For this we propose a secure decentralized mechanism that prevents cheating during transfers of content and credits. Selfish users have little reason to consume their battery power by sending erroneous content. However, the network may contain purely malicious users that do not have battery concerns and whose sole purpose is to cheat.

We propose an offline protocol for simultaneously transferring content and credits, where the participant receiving content blocks can verify not only that the blocks are indeed part of content signed by the other party, but also that they are part of the content expected. The other participant can verify that the received credit shares are part of a valid credit message signed by the other party and at the completion of the protocol can also efficiently reconstruct the credit message, which can then easily be transferred to other participants. Our solution uses Merkle trees to efficiently authenticate content, error correcting (erasure) codes to reconstruct corrupted content blocks and a publicly verifiable secret sharing scheme to authenticate and reconstruct signed credits. Moreover, we provide mechanisms for securely and privately discovering content on neighboring devices, tracking the usage of credits and detecting attacks such as forgery and double spending, all employing common cryptographic primitives. Our solution takes advantage of the occasional free of charge centralized operation mode of devices (wireless hotspots for instance) to bootstrap trust and verify the correctness of past transactions. We implement and test the computation and communication efficiency of the discovery and simple transfer component of the presented protocol on Motorola MPx cellphones and HP iPaq hx4700 PDAs and also extensively simulate the performance of the credit based mechanisms.

The paper is organized as follows. Section 2 discusses related work. Section 3 presents the architecture of the system, Section 4 details our solution and Section 5 describes the defenses provided against known attacks. Section 6 presents our experimental results and Section 7 draws the conclusions.

2 Related Work

Incentives. Yang and Garcia-Molina [24] proposed PPay, a secure and scalable payment scheme designed for peer-to-peer networks. The solution allows participants to purchase coins from a broker and the coins can be transferred between participants in a scalable manner. A coin encodes the identities of both its owner (the participant that bought it initially) and its holder (the participant that is currently able to spend it). The transfer of a coin requires the involvement of the coin's owner. Wei et al. [14] propose a payment protocol that provides the additional feature of anonymity. Both protocols address only the side of the transfer involving coins. While not providing anonymity, our protocol ensures that the other part of the transfer, involving the content, also takes place securely and correctly.

Vishnumurty et al. [22] propose KARMA, which uses global credits (karma) associated with each participant in the system to motivate cooperation in transferring content from other participants. The karma of each participant is stored on a set of other participants (banks) chosen using a distributed hash table (DHT) mechanism. Similar to our work, this solution also relies on the certified mail scheme [9], however, the content transferred cannot be authenticated before the completion of the protocol, when it is too late. Moreover, the secure exchange mechanism proposed requires for each protocol run an additional communication burden between the participating parties and their respective banks.

Golle et al. [13] use a game theoretic approach to model peer-to-peer systems and study the effects of payment mechanisms (money, micropayments, points) on reaching an equilibrium where uploads are rewarded and downloads are penalized. Feldman et al. [10, 11] study mechanisms for encouraging cooperation and punishing free-riders (selfish users) and whitewashers (change of identity) in peer-to-peer systems. In [10] they propose to use the generalized prisoner's dilemma to model the system and use reciprocation as the basis for several incentive mechanisms. In [11] they model the behavior of users based on a local parameter emulating the willingness of users to cooperate. A user decides its contribution dynamically, based on the relationship between the value of the local parameter and the cost of cooperation. While game theoretic mechanisms seem to en-

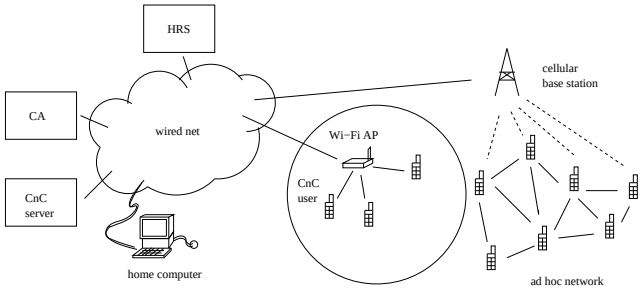

Figure 1. System Model. When connected to a Wi-Fi access point a cellphone connects to the home computer through the HRS server to bypass existing firewalls. When disconnected, but in the vicinity of other devices running the same application, cellphones establish an ad hoc network and locally search for desired content.

courage the overall cooperation of the system participants, none of these works address the issue of purely malicious behavior and its effects on the cooperation of victims.

Fair Electronic Exchange. Several solutions for the fair exchange problem exist, either online, involving a trusted third party in each transaction [25, 7] or offline. The solutions requiring the existence of a trusted third party cannot be used in the environment assumed in this paper, since participants want to keep the protocol local. Schneier and Riordan [19] propose online and offline versions of a protocol where the trusted third party is a public publishing location. Ateniese and Nita-Rotaru [6] propose an offline certified e-mail system using verifiable encryption of digital signatures. We make the observation that the participants in these protocols cannot verify the correctness of the transferred content. Asokan et al. [5] propose a protocol allowing users to exchange digital signatures in a fair way and adapt the protocol to exchange encrypted data.

3 System Model

Our work is built on the Cache-and-Carry (CnC) system [8], which was designed to improve the availability of personal content on memory constrained mobile devices (cellphones). The CnC architecture consists of home computers, storing the personal content of users and mobile devices carried by users and equipped with both cellular and wireless 802.11b interfaces. The home computers store a superset of the content items desired by users. Each content item $\mathtt{c}$ has associated a unique identifier, $\mathtt{Id(c)}$, accessible to any application that knows $\mathtt{c}$. Each mobile device initially synchronizes with the home computer over the home network to obtain a comprehensive list of identifiers of desired content, called the *wish list*. Later, when content consumed by the user is garbage collected, a scheduler running on the device decides which item from the wish list needs to be downloaded first. Based on its location, the mobile device has several different connection modes to the home computer, including USB, Wi-Fi access point mode or cellular. The desired item is then downloaded from the optimum connection. In the CnC design home connectivity is preferred, followed by Wi-Fi managed mode. The cellular connectivity is used only when the previous connections are unavailable and a "wanted-by" timestamp of the content to be fetched approaches its deadline.

In this paper we extend the basic CnC design with the ad hoc functionality of 802.11b wireless cards, allowing users to retrieve content from other co-located mobile devices, in exchange for credits. In the new design, mobile devices are bootstrapped with the identities and public key certificates of two trusted entities, a certification authority (CA) and a CnC server (CnCS). The CA is used to establish trust in the network and each mobile device stores its own public key certificate, signed by the CA. Each mobile device user $\mathtt{A}$ has a cellular account, which uniquely identifies it during its operation [1, 15]. We denote the unique identifier of $\mathtt{A}$ with $\mathtt{Id(A)}$.

As shown in the next section, the CnCS performs various administrative tasks, such as secure credit management. For each content item $\mathtt{c}$, a Merkle tree [17] having blocks of the item as leaves is built. The value stored in the root of the Merkle tree, $\mathtt{mroot(c)}$, is stored on CnCS along with $\mathtt{Id(c)}$. Moreover, for each item $\mathtt{c}$ in its want list, a mobile device stores both the $\mathtt{Id(c)}$ and the $\mathtt{mroot(c)}$ value.

In the following, we use $\mathtt{H(M)}$ to represent the cryptographic hash of message $\mathtt{M}$, $\mathtt{S_X(M)}$ to denote message $\mathtt{M}$ signed by participant $\mathtt{X}$, $\mathtt{E_X(M)}$ to represent message $\mathtt{M}$ encrypted with the public key of $\mathtt{X}$ and $\mathtt{Cert(X)}$ to denote the public key certificate of $\mathtt{X}$ signed by the CA.

4 Credit Based Content Ecosystem

In this section we describe the mechanisms proposed to encourage cooperation and ensure verifiable fair transfers in the ad hoc operation mode of CnC. The three main components of our solution are the *content handling*, *credit management* and *verifiable simultaneous transfer* component. The credit management component consists of mechanisms for generating and transferring credits (described in Section 4.1) and for expiring unused credits (described in Section 4.6). The content handling component addresses the issues of content storage (see Section 4.2) and content discovery (Sec-

tion 4.3). Finally, the secure transactions component encloses mechanisms for simultaneous transfer of content and credits, verification of partially received information and recovery of lost or tampered with information (Section 4.4).

4.1 Credit Minting and Transfers

All credits are issued by CnCS, either initially, when devices join the CnC system or later, upon request and payment. For illustrative purposes, assume that device A purchases credits from CnCS. CnCS issues credits by sending A the following message, one for each unit of credit

$$\mathtt{cr(A)} = \mathtt{Id(A)}, \mathtt{SN}, \mathtt{S_{CnCS}(H(M_A))},$$

where SN is a serial number uniquely identifying the credit unit. We use $\mathtt{M_A}$ to denote $(\mathtt{Id(A)}, \mathtt{SN})$. A can then transfer this credit to device B by appending B's identifier in cleartext to the message and also its own signature

$$\mathtt{cr(B)} = \mathtt{Id(A)}, \mathtt{Id(B)}, \mathtt{SN}, \mathtt{S_{CnCS}(H(M_A))}, \mathtt{S_A(H(M_B))},$$

where $\mathtt{M_B}$ denotes $(\mathtt{Id(A)}, \mathtt{Id(B)}, \mathtt{SN})$. That is, the credits accumulate the unique identifiers of the devices traversed as well as their signatures. When B receives a credit message from A it verifies only the last signature in the list, the one belonging to A. The credit message is not valid unless the signature verifies. That is, B is interested only in A's honest behavior and not the correct behavior of all the devices traversed by the credits. In order for B to transfer this credit unit to C, it similarly has to add C's identifier as well as append its own signature of all the identifiers of devices traversed by the credit message.

In the following, we make the simplifying assumption that a user is charged one credit unit for each content item transferred from another user. The protocol can be easily extended to transfer multiple units of credit, by handling each signed credit message separately. Secure auction mechanisms [12, 23, 21] can be used, allowing each content storer to set its own price for sharing a requested content item.

4.2 Content Storage

When a content item identified by c is initially generated, CnCS splits the content into 2n blocks of equal length $\mathtt{b_1}, \mathtt{b_2}, .., \mathtt{b_{2n}}$ and constructs a Merkle tree [17] having the blocks as leaves. When presented with a request containing $\mathtt{Id(c)}$, CnCS makes public the signed value stored in the root of c's Merkle tree, $\mathtt{mroot(c)}$, using the message

$$\mathtt{Id(c)}, \mathtt{mroot_{Id(c)}}, \mathtt{S_{CnCS}(H(Id(c), mroot_{Id(c)}))},$$

Thus, the $\mathtt{mroot(c)}$ value can be either statically downloaded and stored along with content c on user home computers and then transferred in want lists on mobile devices, or requested later from CnCS by mobile devices in need of fresh content, not yet stored in their want lists.

In the CnC system, a user stores the bulk of the content on her home computer. Before the user begins to download content from the home computer to her mobile device, the home computer performs the following simple and fast computation. It first generates a key $\mathtt{K} \in \{0,1\}^*$ uniformly at random. K is later stored also on the user's mobile device. Then, for each content item c, it generates key $\mathtt{K_c} = \mathtt{H(K, Id(c))}$ which it uses to seed a pseudo-random number generator, G. Using G, it generates q pseudo-random bit positions in content c, $\mathtt{p_1^c}, .., \mathtt{p_q^c}$. When the user's mobile device downloads the want list, along with the identifier of each item c it also stores $\mathtt{H(c(p_1^c), .., c(p_q^c))}$, where $\mathtt{c(p)}$ denotes the bit of content c at position p. We make the observation that for the same user but different content items, the sets of positions $\mathtt{p_1}, .., \mathtt{p_q}$ are different. Moreover, for different users and the same content item the sets of positions are different. The bits at the selected positions are used later (see Section 4.3) to force other devices prove their knowledge of content c.

4.3 Content Discovery

We describe a privacy-preserving protocol allowing users to determine the presence of content items on co-located devices, without leaking the identity of the desired items to devices not storing it. The protocol works in the following way. When device A cannot find Wi-Fi access points to connect to its home computer and download desired items, it switches the wireless card to operate in ad hoc mode, spawns a new thread and listens on a predefined port pn for incoming queries. In parallel, it reads the first content identifier, $\mathtt{Id(c)}$, from its want list, generates a fresh pseudo-random number $\mathtt{N_A}$ to act as challenge and constructs the following message

$$\mathtt{Cert(A)}, \mathtt{Q} = \{\mathtt{N_A}, \mathtt{Id(c)}, \mathtt{K_c}, \mathtt{q}\}, \mathtt{S_A(H(Q))}$$

where $\mathtt{K_c} = \mathtt{H(K, Id(c))}$ is the random number associated with content c (see Section 4.2) and q is the number of query bits. A sends this message periodically as a broadcast on port number pn. We make the observation that while A could optimize the broadcast process by including more than one item identifier in the above query, we prefer to keep it simple for clarity purposes. Moreover, after A transfers content item c from one of its neighbors, it removes $\mathtt{Id(c)}$ from the want list.

When a device B receives the above broadcast message, it discards it if it has already processed the same $(\mathtt{Id(A)}, \mathtt{N_A})$ pair. Otherwise, B checks for the presence of c in its local content store. If the search is successful B uses $\mathtt{K_c}$ to seed a pseudo-random number generator and compute q bit positions $\mathtt{p_1^c}, .., \mathtt{p_q^c}$. It then generates a fresh pseudo-random

number $\mathtt{N_B}$ as its own challenge and produces the message

$$\mathtt{Cert(B)}, \mathtt{N_B}, \mathtt{N_A}, \mathtt{Id(c)}, \mathtt{E_A}(\mathtt{H}(\mathtt{c}(\mathtt{p_1^c}), .., \mathtt{c}(\mathtt{p_q^c}), \mathtt{N_B}, \mathtt{N_A})),$$

$$\mathtt{S_B}(\mathtt{H}(\mathtt{N_B}, \mathtt{N_A}, \mathtt{Id(c)})),$$

and sends it back to A. We use the notation $\mathtt{c}(\mathtt{p_1^c})$ to denote the $\mathtt{p_1^c}$th bit of content c. Note that only the neighbors of A that store c can produce the bits at the queried positions $(\mathtt{p_1^c}, .., \mathtt{p_q^c})$. Without knowing the key K belonging to the home computer of the owner of device A, the chance of guessing all q query bits is $1/2^q$. The $\mathtt{H}(\mathtt{c}(\mathtt{p_1^c}), .., \mathtt{c}(\mathtt{p_q^c})), \mathtt{N_A}, \mathtt{N_B}$ field is sent encrypted in order to prevent malicious neighbors, not knowing c, from learning the value of the bits queried and then replaying it as if knowing c. Note that a neighbor M, who does not store content c, could try pretending it has it, by querying one of its neighbors for the hash of the bits requested by A. However, this message would be overheard by A, that would then interrupt the protocol.

When A receives the above reply it first authenticates it by verifying the signature of the CA on B's certificate, by checking that the third field of the message is indeed its own nonce, then by encrypting the sixth field of the message with the public key stored in $\mathtt{Cert(B)}$ and comparing the result with the hash of the concatenation of the second, third and fourth fields ($\mathtt{N_A}$, $\mathtt{N_B}$ and $\mathtt{Id(c)}$). Then, A decrypts the fifth field and verifies that it equals the hash stored in the want list corresponding to $\mathtt{Id(c)}$ (see Section 4.2), concatenated with the second and third fields of the message ($\mathtt{N_A}$ and $\mathtt{N_B}$). If any of these verifications fails, A stops the protocol. Otherwise, A is convinced that the origin of this reply is B and that B knows c with probability $1 - 1/2^q$. Then, A generates the message

$$\mathtt{N_B}, \mathtt{S_A}(\mathtt{H}(\mathtt{N_B})),$$

and sends it to B. When B receives this messages, it similarly verifies that its origin is A, using the public key certificate of A, $\mathtt{Cert(A)}$, received in the previous message. The fresh nonces $\mathtt{N_A}$ and $\mathtt{N_B}$ are used in this protocol to prove knowledge of the private keys corresponding to the public keys contained in the certificates exchanged. We make the observation that while the broadcast messages of A are sent over UDP, potentially requiring multiple explicit retransmissions, the remaining messages are unicast over TCP, ensuring their correct reception. At the completion of this step A and B are convinced of each other's identity and may proceed with the exchange of content and credits.

In the following, to save space, we assume without explicitly mentioning, that all the messages sent between A and B are signed, since A and B have already exchanged their public key certificates.

4.4 Verifiable Simultaneous Transfer

Participant A knows the value $\mathtt{mroot(c)}$ stored in the root of the Merkle tree associated with the desired content c. The value might have been uploaded on the want list from the home computer but also dynamically retrieved from CnCS, that publishes it as specified in Section 4.2. A initiates the transfer by sending B part of the credit message, $\mathtt{Id(A)}, \mathtt{Id(B)}, \mathtt{SN}, \mathtt{S_{CnCS}}(\mathtt{H}(\mathtt{M_A}))$, leaving out the last part, its own signature. The protocol completes when A has the desired content and, continuing the example in Section 4.1, B has A's credit signature $\mathtt{S_A}(\mathtt{H}(\mathtt{Id(A)}, \mathtt{Id(B)}, \mathtt{SN}))$.

Following the transmission of the above message, A and B begin a simultaneous exchange of content and credits, using the solution of Even et al. [9]. B splits content c in 2n equal sized blocks, builds their Merkle tree, generates n pairs of keys, encrypts each block with a different key and sends all the resulting data, except the keys, to A. Simultaneously, A generates n copies of a credit message for B, splits each into two halves, generates n key pairs, encrypts each of the 2n resulting blocks with a different key and sends the encrypted blocks to B.

A and B use an oblivious transfer protocol (OT) to exchange exactly one key in each key pair, without any of them knowing which key the other has received. A uses the Merkle tree received in the previous step and the $\mathtt{mroot(c)}$ value it stores, in order to verify the authenticity of the recovered blocks. If all the blocks verify, A and B exchange bit by bit all their keys. During each step, both A and B verify that half of the bits belong indeed to the keys exchanged during the previous OT step. At the end of this step, A retrieves all the blocks, concatenates them and obtains content c. B is able to retrieve the left and right halves of at least one pair of signed credits, but is able to use only one such pair at a later time. In Section 5 we show how double spending of credits is detected and the perpetrator caught.

Publicly Verifiable Credit Sharing. In the above protocol, since A splits its signature, B can prove to other parties the knowledge of A's signature but it cannot reconstruct it, thus unnecessarily doubling the size of the credit message. We solve this issue through the use of a publicly verifiable secret sharing scheme PVSS [20], that allows B not only to verify any decrypted share of A's signature after the OT protocol but also, at the end of the transfer, combine the two shares of any signed credit instances received, into A's valid (and publicly verifiable) signature.

To achieve this, in the above protocol, before sending the credit message, A splits each of the n signed credit instances $\mathtt{S_A}(\mathtt{H}(\mathtt{Id(A)}, \mathtt{Id(B)}, \mathtt{SN}, \mathtt{i_{1..n}}))$ for B into two shares, such that each signed credit instance can be reconstructed only if both shares are available. A then pre-commits to each credit instance and its shares as described in [20]. We make

the observation that instead of using two different public keys, A uses only the public key of B to precommit to the shares. The precommitments are signed with A's private key. However, B cannot retrieve the shares from the precommitments. A annotates one share with L and the other share with the R symbol. A follows then the protocol above, that is, encrypts each share of the secret with a different key. For each such share A also sends a proof of correctness, as described in [20]. After the execution of the OT subprotocol for the simultaneous exchange protocol, B can retrieve exactly one share of each secret and can verify its correctness using the corresponding proof. At the end of the protocol, B retrieves the other share of each secret and can reconstruct each secret $S_A(H(Id(A), Id(B), SN, i_{1..n}))$.

Block Recovery. In the above protocol, the chances of cheating for A and B are unbalanced. B has only to correctly retrieve both shares of the same signed credit instance in order to successfully complete the protocol, whereas A cannot reconstruct the content without recovering all 2n blocks. For example, if B cheats by corrupting a single block out of the 2n blocks, its chance of not being caught in time is 1/2, whereas A will not be able to reconstruct the content. We solve this issue through the use of error correcting codes such as the ones proposed in [18, 16]. Instead of dividing the encrypted content c into 2n sequential blocks such that all the blocks are required to construct the content, the CnCS divides the content into 2n larger blocks, such that only $2n - k$ blocks are required to reconstruct the content, where $0 < k < n$ is a small parameter. The CnCS constructs the Merkle tree and mroot value using the 2n error correcting blocks. Note that k has to be strictly smaller than n, otherwise A can reconstruct the content immediately following the oblivious transfer of the encryption keys.

4.5 Alternatives

The solution presented in Section 4.4 is computation and communication intensive. We propose several simple alternatives to remedy this problem. The solutions rely on breaking the content on blocks of predefined length (for instance 256KB) and on changing the payment infrastructure to require one credit per transferred block. The first alternative consists in extending the solution of Section 4.4 to work on a per block basis instead of the whole content. In the second alternative, the involved participants exchange one credit for one block in a serial fashion. That is, either the credit or block transfer takes place first. The block is encrypted with a key precomputed by the two participants. The message carrying the content block contains also the block's authentication information, from the Merkle tree [17]. We remind the reader that this information consists of the values stored on the Merkle tree in the siblings of the nodes on the path from the block's leaf to the root of the tree. If one of the participants cheats, the next transfer does not take place. In a third, hybrid alternative, both participants maintain a reputation of the other participant. The protocol starts the transfer of a content item using the first solution, that is, of simultaneously exchanging one block for one credit. Successful transfers lead to an incremental increase of the reputations of the participants. Once the reputation reaches a predefined threshold the participants switch to the cleartext transfer strategy. A failure leads to a drastic decrease in reputation (halving it for instance) and switches the transfer strategy back to simultaneous transfer.

4.6 Credit Expiration

Too many credits in the system can prevent cooperation, since a device with enough credits will only want to use them to get content and will have no reason to outsource content. In order to control the amount of credit units in the system, we propose to use the following mechanisms. Upon creation, credits are timestamped and given a time-to-live (TTL) and expiration date parameters. Credits can only travel over a number of hops equal to the TTL parameter and the TTL is decreased at each hop. Once the TTL of a credit reaches 0, the current application storing it (A) has to send the credit to the CnCS (over the cellular or WLAN link). The CnCS verifies the authenticity of the credit (see Section 5) and generates a new valid credit for A with fresh TTL and expiration date values. Credits whose expiration date is reached become invalid, however, applications can first spend the credits closer to expiration or can contact CnCS to renew the expiration date.

5 Attack Resilience

The goal of the credit based ecosystem proposed above is to ensure a fair exchange environment for mobile devices operating in ad hoc mode. We describe now several attacks both against participants in the system and against the mechanisms proposed and then show the defenses offered.

Double Spending. Malicious colluding devices may try to spend the same credit more than once. We prevent this by allowing credits to traverse up to TTL devices before being checked back with CnCS. Moreover, CnCS locally stores a table, mapping for each sequence number SN, associated to a credit, the path of devices traversed by the credit before being reported to CnCS.

Let us extend the example from Section 4.1, considering a TTL value of 3 and that the credits from A are later traded by B to a third device C. When CnCS receives a report for a credit message originating at A, from device C, it first verifies the validity of all the signatures contained in the

message, against the public keys of device identifiers listed in cleartext in the credit message (CnCS locally stores certificates of all participating devices). If any signature does not verify, CnCS contacts all the devices listed in the credit message and queries them with the serial number SN from the credit message. Each device is required to send back its own view of the queried SN value. For instance, device A needs to send CnCS both the signature of the credit received from CnCS, $S_{CnCS}(H(Id(A), SN))$ and the signature of the same credit given to B, $S_A(H(Id(A), Id(B), SN))$.

Using this procedure over at most TTL iterations, CnCS is able to find the correct set of devices traversed by the credit message SN. A cannot be framed, since it is the only device that can generate a valid signature $S_A(H(Id(A), Id(B), SN))$. Moreover, if A generates an invalid signature, it will be detected either by B or by CnCS. While more expensive, this procedure is seldom necessary.

After this verification, the server looks in its local table for the serial number SN contained in the credit message. If the entry is empty, CnCS stores the ordered identifiers of all devices traversed by the credit, under the heading SN. It then generates a new signed credit for C $cr(C) = Id(C), SN', S_{CnCS}(H(Id(C), SN'))$, where SN′ is a newly generated unique serial number.

If the entry is not empty, a double spending attack has been discovered. CnCS identifies the attacker by retrieving from its local table the identities of all the devices that have previously signed the credit unit and compares them against the ones contained in the newly received credit message. The comparison is performed starting with the second signature, since the first signature is its own. The first match denotes the device that has duplicated the credits. If no match exists, the culprit is the first device listed in the second credit message, that cannot produce a valid credit signature received from another device. The CnCS can then contact the cellular provider of the culprit, which can then punish it by charging it enough to make this attack economically unfeasible or by closing its account if repeated violations occur.

Forgery. When a device B receives credits from a device A, it only verifies the authenticity of the last signature (A's), instead of verifying all the signatures of devices traversed by the credits. This reduces both the verification time for B and the length of messages carrying credits. This is because to verify all the signatures, the credit message should also contain the public key certificates of all the intermediate devices making a credit message unnecessarily long. We make the observation that when B receives A's signature of the credits, it has already received the public key certificate of A (see Section 4.3).

Since only the last signature on the credit message is verified, a malicious host M could generate new credits, that is, forge the signature of CnCS, sign them and distribute them. While M can succeed in transferring content for fake credits, when the credit message reaches the CnCS, the server can easily detect that it has never generated the serial number contained in the credits. Moreover, the identity of M can be revealed since M's signature is the first to occur on the credits.

Cheating During Transfers. If after the OT protocol one of the participants detects that at least one of the 2n shares received from the other does not verify, it can stop the protocol immediately. At this stage no participant can construct the information desired. If during the second transfer, A tampers with any of the second shares of its credit signature, the cheating will be detected by B at the end of the protocol. However, B needs only two shares of the same credit instance in order to reconstruct A's signature. Similarly, if B corrupts up to k content block during the second transfer, A will still be able to reconstruct the entire content, using erasure codes.

Stopping in the Middle of a Transfer. During the proposed protocol, A could attempt to stop participating after receiving the keys enabling it to decrypt half of the blocks from B. A could then initiate the same protocol with other devices storing c. If each queried device gives A half of the blocks of c in clear, after $\log n$ half-executed protocols, A can have the entire content c, with high probability, without actually giving credits to any other device. However, the requirement of consuming $\log n$ times more battery power can act as a counter incentive for this attack.

6 Experimental Results

In this section we experimentally analyze our credit based ecosystem. In Section 6.1 we measure the overhead of the secure content discovery part of our secure transfer protocol, using a prototype implementation. In Section 6.2 we study the impact of the credit based sharing strategy in encouraging cooperation between CnC applications in a larger scale ad hoc environment.

6.1 Security Overhead

We investigate the security overhead introduced by the credit based exchange protocol presented in Section 4. For the transfer of content and credits we used the second alternative of Section 4.5, where the transfer takes place in clear but authenticity of content is verified using Merkle tree information. The protocol was implemented on Motorola MPx cellular phones (Texas Instruments OMAP 733 at 200 MHz processor and 32 MB of RAM) and HP iPAQ hx4700

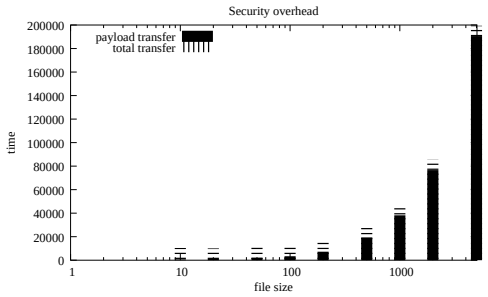

Figure 2. Authentication overhead of the credit based exchange protocol. For small items the overhead is high (more than 90% for 10KB items) but it is small for large items (less than 5% for a 5MB file).

(Intel PXA270 at 624 MHz processor and 64 MB of RAM). Both devices have integrated 802.11b network cards. The prototype implementation is written in C# for Microsoft's .Net Compact Framework and uses security primitives provided by the Bouncy Castle [3] library.

In our experiment we transfer files of size between 10KB and 5MB and measure the time taken by the secure discovery of the content versus the time taken by the entire protocol. For each file size, we perform the transfer 5 times and represent only the average values. Figure 2 shows the results of our experiment. While for small files the overhead is significant (94% for a 10KB file) for relatively large files it becomes negligible (4.41% for a 5MB file).

We are currently porting our C# on Compact Framework implementation to C/C++ on Linux based E680i phones (312MHz Intel XScale Bulverde (PXA270) processor, 32MB RAM and Fujitsu Wi-Fi card). Initial experiments show that the content discovery phase and a simple transfer of a 2.5MB file takes 7 seconds between a E680i phone and a Dell Latitude E600 laptop and 12 seconds between two E680i phones, a significant improvement over the previous implementation. The difference between 12 and 7 seconds is due to the overhead of writing to the phone's flash card.

6.2 Strategy Evaluation

We evaluate the credit based sharing strategy by randomly placing 500 devices in a $900 \times 900\ m^2$ area, where each device has a transmission range of 115 m. We divide time into epochs such that a device can perform a single transfer during each epoch, acting either as content client or server. Each device has an initial battery level of 50 units, where a single battery unit is consumed by a participant, acting either as server or as client, during a transfer. We assume a total population of 5000 content items, where each item is uniquely identified. Each device stores initially 100 items randomly chosen from the content population and a want list of 400 desired content items, ordered in decreasing order of relevance. In the following experiments the content of the want list and its order are randomly chosen. During each epoch, a device sends to its neighbors a broadcast query containing identifying information for the top 100 items in its want list. Our simulation ends when the network reaches a stable state. A stable state occurs at the end of a time epoch during which no transfer takes place. When the network reaches stable state, no device stores and is willing to share an item that other device wants and both devices have enough battery to perform the transfer. This ensures that in subsequent time epochs no other transfer will take place.

We measure the performance of sharing strategies using three metrics, *transfer rate*, *satisfaction level* and *battery level*. The transfer rate measures the number of transfers performed during each time epoch. The satisfaction and battery levels denote the number of items acquired by devices and their battery level at a given time.

Effects of Selfishness. In the first experiment, 50% randomly chosen devices exhibit a selfish behavior whereas the remaining 50% are altruistic. A selfish device will act only as a client, but only when one of its neighbors stores an item desired and agrees to cooperate. An altruistic device will choose to act as a server whenever it can satisfy a query from one of its neighbors and it receives the query of its neighbors before an answer to its own queries. Figure 3(a) shows the evolution of transfer rates in time, both for selfish and for altruistic devices. At the beginning of the experiment, the number of altruistic devices able to acquire content is half the number of selfish devices that successfully complete transfers. This is because, on average, half of the altruistic devices will interact with selfish devices and half will cooperate among themselves. In time, the number of transfers of altruistic devices remains almost constant, with a quick decrease to 0 when most devices run out of battery. However, the number of transfers of selfish devices declines in time, at the 50th time epoch being almost the same as the transfers of altruistic devices. This decrease can be explained by selfish devices gradually downloading all desired items stored by their altruistic neighbors.

Figure 3(b) displays the distribution of satisfaction when the system reaches stable state. The mean number of items acquired by selfish devices is 21.81, with a standard error of 0.39, whereas the mean number of items acquired by altruistic devices is 13.35 with a standard error of 0.17. Thus, selfish devices get more than 63% more items than cooperating devices. Figure 3(c) shows in logarithmic scale the distribution of remaining battery power of selfish and cooperating devices at stable state. 210 out of 250 cooperating

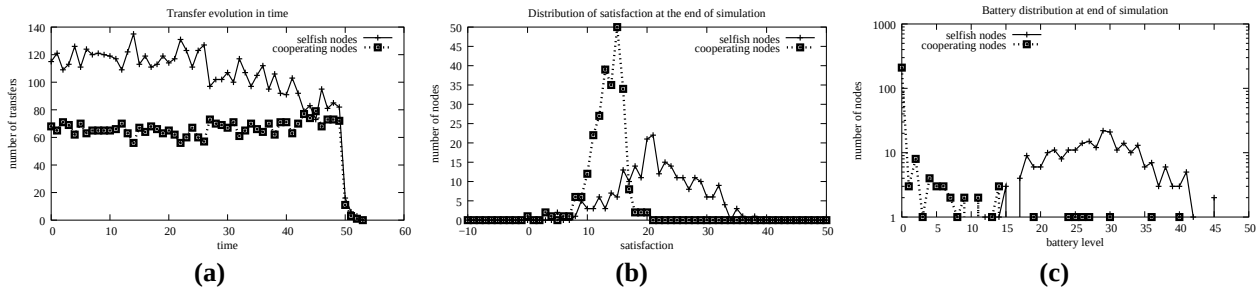

Figure 3. Performance of selfish and cooperating participants in an environment containing 50% selfish and 50% altruistic devices. Altruistic devices download 63% less content than selfish ones and are left with almost no battery. The average battery left in selfish devices is above 28 units, out of the initial 50 units.

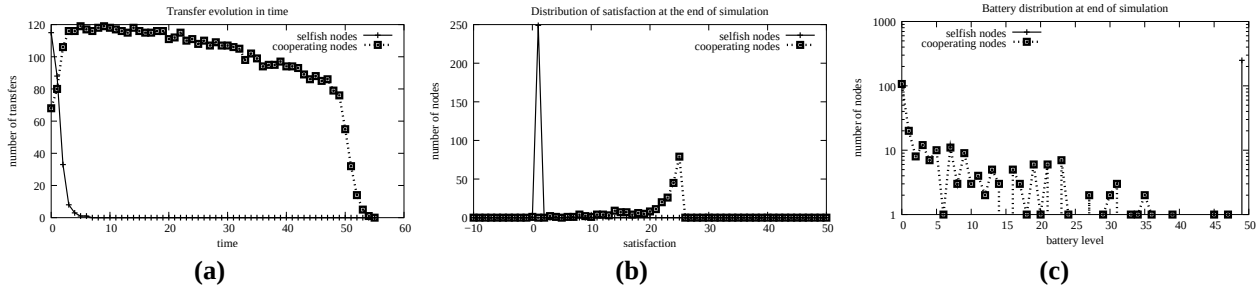

Figure 4. Performance of selfish and credit based cooperating devices in an environment containing 50% selfish 50% credit based cooperating devices. The average number of items downloaded by cooperating devices is above 20, whereas selfish devices only get 1 item. Moreover, the average battery left in cooperating devices is above 6 units, much improved over altruistic behavior.

devices are left without any battery and the remaining ones have very little battery left. In contrast, no selfish devices are left without battery. The mean battery power left on selfish devices is 28.18 units, with a standard error of 0.39. Thus, in this scenario, selfishness pays off. Selfish devices get both more items and are left with more battery at the end of the protocol.

In the following we experiment again with a 50% selfish population. However, the other half of the population is cooperating as long as it receives credits for its cooperation. We assume that each device, including selfish ones, has initially only 1 credit, allowing the device to acquire exactly one item from another device. However, once a device consumes its credits, any other device will refuse to share items with it. Thus, a device without credits can either become a server, share its items and earn credits, or choose to be selfish and starve. A cooperating device that has at least one credit, can choose to either act as a client or as server. Figure 4(a) shows the evolution in time of the number of transfers initiated by selfish and cooperating devices. The selfish devices manage to transfer items only during the first 6 time epochs but are isolated from then on both by their selfish and their cooperating neighbors. The cooperating devices are performing much better this time, though. Until the 50th epoch, marking the battery depletion of most devices, more than 80 cooperating devices manage to download an item in each epoch.

The distribution of satisfaction when the system reaches stable state is shown in Figure 4(b). While each selfish device gets exactly one item during the entire simulation, 189 cooperating devices get more than 20 items each, out of which 79 get 25 items each. The mean number of items downloaded by a cooperating device is 21.16 with a standard error of 0.32. Figure 4(c) depicts the distribution of battery at the end of the simulation, where the number of devices is shown in logarithmic scale. The selfish devices all have 49 units of battery left, since each has participated in exactly one transfer. The mean battery level of cooperating devices is 6.68 with a standard error of 0.61, which is a significant improvement over the previous experiment. In

conclusion, when devices take into consideration the global credit levels of neighbors, before deciding to transfer content, their level of satisfaction and also remaining battery are much improved over an altruistic strategy. In contrast, selfish devices conserve their battery, however, are able to acquire only a few items, using their initial amount of credits.

7 Conclusions

In this paper we study several security issues associated with the use of credit based systems to enable cooperation in environments where devices use their cellular and ad-hoc communication capabilities to effectively transfer desired content from their home computer or neighboring devices. We propose an efficient offline mechanism for simultaneously exchanging content and credits that allows the participants involved to verify at each step the validity of the partially exchanged content. We provide mechanisms for securely generating, managing and tracking the correct spending of credits in an offline manner. Our solution does not require the permanent availability of a trusted server mediating transactions. We prove through extensive simulations that our credit based exchange mechanisms are effective in protecting cooperating devices from selfish devices.

References

[1] Generic authentication architecture (gaa). `http://www.3gpp.org/ftp/Specs/html-info/33220.htm`.

[2] *iRadio*. URL `http://broadband.motorola.com/iradio/`.

[3] *The Legion of the Bouncy Castle*. URL `http://www.bouncycastle.org/`.

[4] *IEEE Std 802.11b-1999*. 1999. URL `http://standards.ieee.org/`.

[5] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.

[6] G. Ateniese and C. Nita-Rotaru. Stateless-recipient certified e-mail system based on verifiable encryption. In *CT-RSA '02: Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology*, pages 182–199, 2002.

[7] A. Bahreman and J. D. Tygar. Certified electronic mail. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, pages 3–19, 1994.

[8] R. Chaudhri, M. Pearce, and J. Almaula. Cache and Carry: Seamless Content Consumption in an Intermittently Connected World. In *Proceedings of IEEE Consummer Communications and Networking Conference (CCNC)*, 2007.

[9] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[10] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce*, pages 102–111, 2004.

[11] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and whitewashing in peer-to-peer systems. In *PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 228–236, 2004.

[12] M. Franklin and M. Reiter. The design and implementation of a secure auction service. In *Proceedings of IEEE Conference on Security and Privacy*, 1995.

[13] P. Golle, K. Leyton-Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks, 2001.

[14] A. J. S. Kai Wei, Yih-Farn Chen and B. Vo. WhoPay: a Scalable and Anonymous Payment System for Peer-to-Peer Environments. Technical Report UCB/CSD-05-1386, EECS Department, University of California, Berkeley, 2005.

[15] P. Laitinen, P. Ginzboorg, N. Asokan, S. Holtmanns, and V. Niemi. Extending cellular authentication as a service. In *IEE 3rd Annual Forum on Secure Mobile Communications*, 2005.

[16] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. In *ACM Symposium on Theory of Computing (STOC)*, 2001.

[17] R. C. Merkle. Protocols for public key cryptosystem. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 122–134, 1980.

[18] M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 36(2):335–348, 1989.

[19] B. Schneier and J. Riordan. A certified e-mail protocol. In *ACSAC '98: Proceedings of the 14th Annual Computer Security Applications Conference*, page 347, 1998.

[20] B. Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 148–164, 1999.

[21] F. Stajano and R. Anderson. The cocaine auction protocol: On the power of anonymous broadcast. In *Third International Workshop on Information Hiding*, 1999.

[22] V. Vishnimurthy, S. Chandrakumar, , and E. Gun Sirer. Karma: A secure economic framework for p2p resource sharing. In *Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[23] Y. Watanabe and H. Imai. Reducing the round complexity of a sealed-bid auction protocol with an off-line ttp. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 80–86, 2000.

[24] B. Yang and H. Garcia-Molina. PPay: micropayments for peer-to-peer systems. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications sec urity*, pages 300–310, 2003.

[25] J. Zhou and D. Gollmann. Certified electronic mail. In *ESORICS '96: Proceedings of the 4th European Symposium on Research in Computer Security*, pages 160–171, 1996.