NetCash: A design for practical electronic currency on the Internet

Gennady Medvinsky B. Clifford Neuman

Information Sciences Institute
University of Southern California

Abstract

NetCash is a framework that supports realtime electronic payments with provision of anonymity over an unsecure network. It is designed to enable new types of services on the Internet which have not been practical to date because of the absence of a secure, scalable, potentially anonymous payment method.

NetCash strikes a balance between unconditionally anonymous electronic currency, and signed instruments analogous to checks that are more scalable but identify the principals in a transaction. It does this by providing the framework within which proposed electronic currency protocols can be integrated with the scalable, but non-anonymous, electronic banking infrastructure that has been proposed for routine transactions.

1 Introduction

As the world becomes more connected, the number and variety of network resources and services requiring monetary payments will grow rapidly. For example, access to online documents might require payment of royalties. Many offline services that formerly relied on cash now use electronic payment methods. More recently, protocols have been proposed [5] to support online payment for such services over open networks. While these protocols are suitable for the vast majority of transactions, most do not protect the identities of the parties to a transaction.

Concern for privacy dictates that it should be possible to protect the identity of the parties to a transaction. This is important to prevent the accumulation of information about the habits of individuals, e.g., the documents they read, or the items they purchase. It is also important to protect parties that receive payment in certain situations, such as rewards. Many protocols have been proposed for anonymous transactions, among them those by Chaum [2]. These protocols typically require a central bank that is involved in all transactions.

In this paper, we present a framework for electronic transactions that combines the benefits of anonymous transactions with the scalability of non-anonymous online payment protocols. The paper begins with a discussion of possible requirements for electronic payment systems, followed by a discussion of related work. We then present a scalable framework for anonymous transactions, discuss the benefits of the framework, and describe how it can be applied to electronic currency protocols. The paper concludes with a discussion of the scope and limitations of the framework.

2 Requirements for electronic currency

Among the desirable properties for an electronic currency system are: security, anonymity, scalability, acceptability, offline operation, transferability, and hardware independence. Some of these requirements are also described in [6].

Security: Forging paper currency is difficult. Unfortunately, electronic currency is just data and is easily copied. Copying or double spending of currency should be prevented or detected. Ideally, the illegal creation, copying, and reuse of electronic cash should be unconditionally or computationally impossible. Some systems rely instead on post-fact detection and punishment of double spending [2].

Anonymity: The identity of an individual using electronic currency should be protected; it should not be possible to monitor an individual's spending patterns, nor determine one's source of income. An individual is traceable in traditional transaction systems such as checks and credit cards. Some protocols are unconditionally untraceable, where an individual's spending can not be determined even if all parties collude [1, 2]. For some transactions, weaker forms of anonymity may be appropriate, e.g. traceability can be made difficult enough that the cost of obtaining such information outweighs the benefit.

Scalability: A system is scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance. The existence of a central server through which transactions must be processed limits the scale of the system. The mechanisms used to detect double spending also affects scalability. Most proposed e-cash protocols assume that the currency server will record all coins that have been previously spent and check this list when verifying a transaction [2, 6, 7]. This database will grow over time, increasing the cost to detect double spending. Even if the life of a coin is bounded, there is no upper bound on the amount of storage required since the storage requirement depends on the rate at which coins are used, rather than on the number of coins in circulation.

Acceptability: Most e-cash proposals use a single bank [2, 6, 7]. In practice, multiple banks are needed for scalability, and because not all users will be customers of a single bank. In such an environment, it is important that currency minted by one bank be accepted by others. Without such acceptability, electronic currency could only be used between parties that share a common bank. When currency minted by one bank is accepted by others, reconciliation between banks should occur automatically. To our knowledge, NetCash is the first system that satisfies this requirement.

Off-line operation: The ability for two parties to make a safe transaction without instantaneously contacting the authority that issued the currency is desirable.

Transferability: The ability of the recipient of electronic currency to spend the currency with a third party without first contacting the currency server is desirable. Such transferability can improve anonymity, but it complicates the mechanism that assures security.

Hardware independence: To prevent double spending during offline operation, some e-cash protocols rely on tamper-proof hardware [4]. A drawback to this approach is that new technology might allow the compromise of such hardware, leaving users vulnerable to double spending.

3 Related work

There have been numerous recent proposals for protocols to support unconditionally untraceable, electronic currency [6, 7]. Many of these proposals are variants of and improvements upon proposals by Chaum [2, 3]. Although these protocols address many of the the requirements from section 2, unconditional anonymity is achieved at the expense of scalability, and acceptability is unaddressed.

NetCash provides scalability and acceptability with weaker anonymity and only a limited form of offline-operation. We believe that for many transactions this is sufficient. Where unconditional anonymity or completely offline operation is required, our framework can be extended to integrate exchanges from other protocols.

Protocols have been proposed that support scalable distributed accounting without anonymity [5]. These protocols provide an accounting infrastructure within which funds can be transferred between clients and servers. Because these protocols do not provide anonymity, they are not by themselves sufficient for our purposes in this paper. They will, however, be used to reconcile balances across currency servers, and to allow users to withdraw and deposit money into existing accounts.

4 Framework

NetCash is designed to support realtime electronic payments with varying transaction anonymity characteristics to geographically dispersed clients in multiple administrative domains. The primary contribution of NetCash is as a framework for integrating anonymous electronic currency into the global banking and accounting infrastructure. Section 5 defines a practical electronic currency protocol that provides weaker anonymity than the unconditional anonymity provided by Chaum [2]. The framework is useful even where unconditional anonymity is required since the protocols implementing Chaum's currency can replace the basic building blocks of the protocol described in section 5, while leaving the basic framework intact.

The NetCash infrastructure is based on independently managed, distributed currency servers that provide a point of exchange between anonymous electronic currency and non-anonymous instruments such as electronic checks. In the framework, checks based on the global accounting infrastructure [5] tie together currency servers in different administrative domains, into a financial federation where currency minted by different servers is accepted.

An organization wishing to set up and manage a currency server obtains insurance for the new currency from an agency similar to federal deposit and insurance corporation; the currency is backed by account balances registered to the currency server in the non-anonymous accounting infrastructure. We will refer to the insuring agency as the federal insurance corporation (FIC). To add a new currency server, an authentication service is used to establish a secure connection between the currency server and FIC. The currency server creates a public key pair and sends the public key to FIC over the secure channel (the corresponding private key is used for signing coins). In return FIC issues a certificate of insurance for producing and managing the currency. Figure 1 shows a certificate of insurance. It includes a unique ID to identify a particular currency server named in the certificate, the public key of the currency server along with the date of issue and an expiration date of the certificate. All the information is sealed with the private key of FIC. Based on this certificate different currency servers and financial institutions will accept the currency of a given server as legal tender. The consequences of a compromise of K_{FIC}^{-1} are severe.

$$\{\text{Certif\_id, CS\_name, } K_{CS}\text{, issue\_date, exp\_date}\}K_{FIC}^{-1}$$

Figure 1: A certificate for minting currency

$$\{\text{CS\_name, s\_addr, exp\_date, serial\_num, coin\_val}\}K_{CS}^{-1}, \text{Certif\_id}$$

Figure 2: Electronic coin

It is up to the client to select a currency server. A reasonable choice could be based on geographical proximity and the amount of trust the client places in the currency server. A currency server provides the following services to its clients: coin verification (detection of double spending), coin exchange for untraceability, purchasing coins with checks, cashing in coins for checks. The latter two services as well as verification of coins minted by other servers relies on the accounting infrastructure described in [5] and is not further described in this paper. Below, we describe the basic function provided by the currency server to facilitate coin verification and potentially anonymous coin exchanges.

4.1 Functionality and structure of NetCash components

A coin in our protocol (see figure 2) includes among other information a serial number signed with the currency servers private key. This information uniquely identifies the coin to the currency server that issued it. The currency server keeps a list of serial numbers for all outstanding coins[1]. When a participant in a monetary transaction sends a coin for verification, the currency server checks the coin's serial number against the outstanding list. If the serial number is found, the coin is valid (has not been spent before). The serial number is deleted from the list, and a new coin with a different serial number is issued to the client and the new serial number added to the list. If a coin is tendered for which the serial number is not found, an attempt at double spending has been detected and the exchange is refused.

A currency server is implemented as a collection of servers connected on a network. This set of servers has a collective name valid on the Internet. Initially, each server is allowed to create a number of coins based on a policy set by the agency insuring the currency. Each server will manage coins with a range of values.

The structure of an electronic coin is shown in figure 2. The monetary value of the coin is specified in the coin_val field. An internet address is part of the coin, allowing the coin to be sent directly to the server keeping track of it. If the currency server is not reachable at the address in a coin, the name of the currency server (CS_name) is used to find the address by querying a directory server. Time stamps in the coins limit the state that must be maintained by each currency server.

All information in a coin is sealed with the private key K_{CS}^{-1} of the currency server. A client wanting to decrypt the coin can use the Certif_id, which provides a mapping to an appropriate certificate, thus obtaining the public key K_{CS}. The validity of the coin is proven upon successful decryption

[1] Depending on the characteristics of currency used, this list might be represented as a bit vector or as a list of serial numbers.

of the coin, but the fact that it has not been double spent is not assured until the coin is exchanged for a new coin directly with the currency server.

5 NetCash exchange protocols

In this section, we define protocols for monetary exchanges with provision of anonymity. We will concentrate on protocols providing practical anonymity. In our protocol description, we make the assumption that clients use different currency servers, each of which mints its own currency backed by account balances in the non-anonymous accounting infrastructure, but registered in the name of the currency server itself, not its clients. For the sake of clarity in our protocols, we refer to such non-anonymous transactions using this accounting infrastructure as the transfer of a check. The mechanics of the non-anonymous transfers will not be discussed here but can instead be found in [5].

5.1 Notation

The participants in our protocols are: clients, merchants, and currency servers. The first two are represented as A and B, the currency servers are denoted as CS. Banks are simply merchants in our protocol and are also represented by A and B. X stands for any participant.

The term ‘transaction’ is used here to mean a monetary transaction between participants. A payment happens in one direction from A to B. Encryption is represented by curly braces {}. A public key is represented as the letter K with a subscript naming the owner of the key. A subscript ending with letter N indicates a newly generated key not advertised anywhere in association with the principal's identity. A private key is the inverse of the public key and is represented as K_{key}^{-1}. Keys for a symmetric encryption system are represented by the letters SK with a subscript.

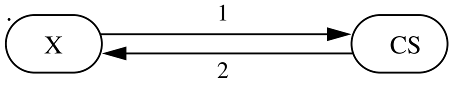
5.2 Basic building blocks

In this section, we describe basic exchanges that serve as building blocks for later protocols. As discussed in section 4, currency servers issue coins that may be used by their clients while preserving their anonymity. Currency servers provide the point of entry to the accounting infrastructure, accepting coins from other currency servers as well as other financial instruments (checks for example).

Anonymity of a client is preserved in a pure currency exchange with the CS for several reasons: the CS exchanges coins with a client by providing new coins having different serial numbers and does not keep records pairing the coins accepted with those issued. The exchange occurs anonymously; the CS does not know who is trading in the coins. It might be the client itself, or an anonymous merchant that just accepted the coins and needs to exchange them to be sure that they are not double spent. Finally, the client selects the CS to be used and is likely to choose one it trusts.

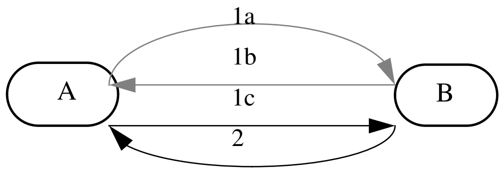
5.2.1 Exchange with the currency server

The basic exchange shown in figure 3, provides the following services: coin verification (detection of double spending), coin exchange for untraceability, purchasing coins with checks, and cashing in coins for checks. Only the latter two operations expose the identity of X. Initially, X is assumed to know the currency server's public key K_{CS}, cached from previous transactions or obtained directly from CS embedded in a certificate (see figure 1). In step 1, X sends a check or a coin (collectively referred to as an instrument), SK_X a newly chosen secret key and an indication of the transaction to be performed (e.g., whether it wants new coins or a check,



1. {instrument, SK_X, transaction}K_{CS}
2. {instrument}SK_X

Figure 3: Exchange with the currency server



1a. K_{AN}
1b. { K_{BN} }K_{AN}
1c. {coins, SK_{AN1}, K_{ses}, S_id }K_{BN}, {Certif_id, K_{CS}, issue_date, expiration_date}K_{FIC}^{-1}
2. {{amount, T_id, date }K_{BN}^{-1} }SK_{AN1}

Figure 4: Simple payment, optional steps: 1a & 1b.

and if a check, the name of the party to which it should be payable), all sealed with the currency server's public key.

If the instrument provided is a coin issued by the currency server itself, the coin is checked for double spending by verifying whether the record associated with the coin exists. If the instrument is a coin issued by another currency server, the local currency server contacts the remote currency server to convert the coin, accepting in return a check payable to the local currency server, which is then cleared through the global accounting infrastructure. If the instrument is a check, the local currency server clears it, depositing the proceeds in its own account.

In the second step, the server returns the desired instrument, either newly issued coins, or a check made payable to the individual named in the transaction. Encryption with SK_X, proves the identity of the CS and prevents the contents of the message from exposure to an attacker.

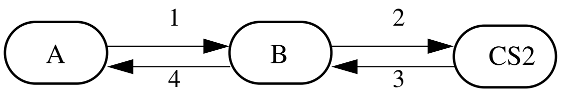
5.2.2 Simple payor-payee exchange

Figure 4 shows a simple payment protocol where A remains anonymous. B has the option to remain anonymous with additional provisions described below. Upon completion of the protocol, B is not protected against double spending and A is not guaranteed a valid receipt.

Initially A is assumed to posses B's address. Messages 1a and 1b are used to obtain B's public key, either one that identifies B, or one generated on the fly if B is to remain anonymous. If A already knows B's public key these messages may be dropped. In step 1c, A sends the coins[2], the identifier of the desired service S_id along with two keys SK_{AN1} and K_{ses}. B uses K_{CS} to verify that a certified currency server minted the coins. In order to pair the coins with a connection, B retains the session key K_{ses}; at the time the service is to be provided, B verifies that A knows the session key. In the last step, B returns a receipt signed with its private key and encrypted with SK_{AN1}, thus preventing the contents of the message from exposure to an attacker. The receipt includes amount paid, date and a unique identifier T_id that will be used along with the session key to obtain the service.

Note if steps 1a and 1b are used to obtain an anonymous public key, the protocol can withstand passive attacks in the

[2]The insurance certificate for the coins can be obtained in one of the following ways: directly from the currency server, sent with the coins as shown in figure 4, or retrieved from a directory service.



1. {coins, SK_{AN1}, K_{ses}, S_id }K_B
2. {coins, SK_{BN}, transaction}K_{CS}
3. {new_coins}SK_{BN}
4. {{amount, T_id, date }K_B^{-1} }SK_{AN1}

Figure 5: Prevention of double spending.

sense that the privacy of the transaction is maintained, but is vulnerable to an active attack where an anonymous attacker impersonates the anonymous service provider.

5.3 Combining the building blocks

Below, we define protocols preserving the payee's and payor's anonymity using combinations of building blocks to provide guarantees against double spending, no receipt, or invalid receipt. To avoid redundancy, we omit the detailed description of steps within a given block; refer to section 5.2 for this information.

5.3.1 Exchange with protection from double spending

We combine the two protocol modules in figures 3 and 4 as shown in figure 5 to protect B from double spending by an anonymous payor A. A is assumed to know B's public key, but the anonymity of B can be protected by adding message 1a and 1b from figure 4. After receiving coins from A, B verifies the coins with the currency server. If the coins haven't been spent already, then B issues a receipt to A. The shortcoming of this protocol is that it only provides protection from fraud by the payor. B could simply spend A's coin without providing a valid receipt. The protocol presented in the next section solves this problem.

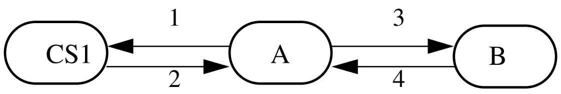
5.3.2 Exchange with both parties protected from fraud

We would like to extend the model presented in section 5.3.1 to eliminate B's ability to cheat. The protocol presented in this section preserves A's anonymity, protects B from double spending, and guarantees A a valid receipt or its money back. To make such guarantees possible we extend the definition of a coin. A coin can be customized for a given principal, i.e., it can only be used by that principal for a certain window of time. Thus, the payor A can use the currency server to obtain a coin triplet $<C_B,C_A,C_X>$. Each coin in the triplet has the same serial number and coin value.

During the first window, C_B is the only coin that can be used with the currency server. In the next window, only C_A can be used, and in the last window the C_X coin is used exclusively. This can be implemented by embedding window boundaries (time stamps) into each coin in the triplet.

The first two coins, intended for B and A respectively have keys embedded in them. If either party wants to use its coin in a transaction with CS1, it must prove knowledge of the embedded key. For example B's public key K_B is encrypted in C_B, during the transaction with CS1 it must prove knowledge of K_B^{-1}. The third coin C_X, does not have a key encrypted in it and can be used by anyone. Additional information embedded in C_A is B's public key, this is done to reduce the amount of state information needed to be maintained by the server in order to issue A a receipt. Also, an additional bit needs to be associated with a coin serial number in the CS1's database to keep track of whether A or B spent the coin.

In the transaction with B, A will keep coins C_X and C_A and pass C_B to B. If B does not give a receipt to A, A



1. { coins, SK_{AN1}, K_B, $date_B$, $date_A$, amount}K_{CS1}
2. {$< C_B,C_A,C_X>$, $<,,C_X>$}SK_{AN1}
3. { C_B, SK_{AN2}, K_{ses}, S_id }K_B
4. {{amount, T_id, date }K_B^{-1} }SK_{AN2}

Figure 6: Protection from fraud.

can query the currency server and check whether B spent the coin. If B spent the coin, the currency server will issue A a receipt specifying the coin value and B's public key. Otherwise, A can obtain a refund during the window in which C_A is valid. B should keep track of C_B until it expires in case A attempts to double spend C_B with B. C_X is provided for additional flexibility in monetary transactions when A does not ultimately spend the coin with B. Figure 6 shows the steps of the enhanced protocol. In step 1, A sends coins to its currency server to obtain a coin triplet[3] ($date_A$ and $date_B$ denote expiration dates for A's and B's window of operation). The currency server creates a coin triplet and embeds the information in the coins as described above. CS1 returns the triplet, along with possible change $<,,C_X>$ if the amount specified was less than the total value of the coins sent in step 1. In step 3, A passes C_B to B. B must convert the coin while it's valid, during the first interval. In the next step B returns a valid receipt to A. In case it doesn't, during the second time interval, A sends C_A to CS1. CS1 then checks whether the coin was spent in the first window of time. If it was, CS1 returns a receipt specifying B's key and the value of the coin all signed with CS1's private key. In case the coin was not spent, CS1 will issue a new coin to A.

It should be noted even though B is a client of CS2, it can still accept coins minted by other authorities because of the accounting infrastructure on which NetCash is based.

The anonymity of the payee can be achieved by combining steps 1a and 1b of figure 4 with the protocol presented in this section. In the resulting protocol, the reciept provided to the payor is not very useful since the payee is anonymous. The details of the protocol are left as an exercise to the reader.

5.4 Off-line protocols

In an offline transaction, it is desirable to prevent double spending while preserving the anonymity of the participating parties. Transactions conducted in the offline mode where neither party contacts the currency server during the exchange can be supported in NetCash by several means.

The protocol shown in figure 6 can be used as follows: If A knows ahead of time that it is going to conduct business with B, steps 1 & 2 can be done in advance. At a later time, A & B go through an exchange, using steps 3 & 4, where upon completion, double spending is prevented and payor's anonymity is maintained. A drawback of this protocol is the payor has to know in advance with which particular party a transaction will be performed.

Another approach to offline transactions is to use the protocol shown in figure 4 in conjunction with tamper-proof electronic wallets. Double spending is prevented by properties of the hardware. The problems with this approach was described in section 2.

Currently, we are looking into incorporating Chaum's post-fact punishment scheme[2] into NetCash. Double spend-

[3] In case different coin denominations are desired, A could specify several amounts, and obtain a number of triplets each having a particular value.

ing makes it statistically possible to determine the identity of a dishonest client. The drawback with this approach is that post-fact punishment may be unacceptable to financial institutions due to the complications in tracking and punishing potential violators.

6 Discussion

NetCash combines the benefits of anonymous transactions with the scalability of non-anonymous online payment protocols. It is secure, scalable, valid across administrative domains, and provides some assurance of anonymity for the parties to a transaction. In this section, we discuss the benefits and drawbacks of NetCash, revisiting some of the requirements from section 2.

Where it is possible for at least one party to interact with a currency server at some point during a transaction, NetCash is secure. Double spending is either detected at the time the recipient verifies or exchanges coins with the currency server, or the coins can only be spent by the recipient during an initial time window, allowing the recipient to cash them in before they can be double spent.

Because independent currency servers exist NetCash is more scalable than other e-cash proposals. When coins are exchanged with remote currency servers, the balances of the currency servers (the backing of the currency) are adjusted through the scalable, but non-anonymous, accounting infrastructure proposed in [5]. The anonymity of the client is not jeopardized because only the currency servers themselves are identified in the non-anonymous transaction.

The anonymity provided by NetCash is weaker than the unconditional anonymity provided by Chaum. In particular, at the point that a client purchases coins from a currency server by check, or cashes in coins, it is possible for the currency server to record which coins have been issued to a particular client. It is expected that currency servers will not do so, and it is likely that the agreement with clients will specifically preclude it. Additionally, the client can choose its own currency server, and will choose one that it feels it can trust.

Once coins have been purchased, they can continue to circulate without identifying the intermediaries. Although the currency server is involved each time a coin changes hands, and could conceivably track which coins are exchanged for others though prohibited from doing so, it will not know the identity of the intermediaries until one of the parties chooses to identify itself when converting in coins. The longer the chain of intermediaries, the less information that is available about who made purchases where.

Although coins may be transferred in our scheme without interaction with the currency server, when coins are used in this manner, no assurances exist that a coin has not been double spent. Thus, among a group of individuals that trust one another (or each others tamper-proof hardware), coin transfer is possible. Parties to a transaction would need to eventually verify and exchange their coins to limit their vulnerability to double spending.

Our approach supports partially offline operation, where the parties are offline during the final exchange; secure operations do require that at least one party interact with a currency server at some point during a transaction.

Where unconditional anonymity or completely offline operation is required, our framework can be extended to support exchanges from Chaum's protocol or from other electronic currency mechanisms. Such exchanges could be applied to only those transactions that require them, while still providing scalability, acceptability, and interoperability across mechanisms.

7 Conclusion

This paper presents a framework for electronic transactions that combines the benefits of anonymous transactions with the scalability of non-anonymous online payment protocols. Our framework is secure, scalable, acceptable across administrative domains, and provides some assurance of anonymity for the parties a transaction. Our approach supports partially offline operation, where the parties are offline during the final exchange; secure operations do require that at least one party interact with a currency server at some point during a transaction.

Where unconditional anonymity or completely offline operation is required our framework can be extended to support exchanges from other electronic currency mechanisms for those transactions that require them, while still providing scalability, acceptability, and interoperability across mechanisms.

Acknowledgments

We would like to give a special thanks to Yacov Yacobi for his insightful comments regarding e-cash. We would also like to thank Celeste Anderson, Deborah Estrin, Katia Obraczka, Barry Perkins, Jon Postel, Stuart Stubblebine, and Peter Will for discussion and comments on drafts of this paper. A great thanks to Shai Herzog for coming up with the name NetCash.

References

[1] D. Chaum, B. Boer, E. Heyst, S. Mjolsnes, and A. Steenbeek. Efficient off-line electronic checks. In *Proceedings of Eurocrypt '89*, 1989.

[2] D. Chaum, A. Fiat, and N. Naor. Untraceable electronic cash. In *Proceedings of Crypto '88*, 1988.

[3] Chaum D. Security without identification: Transaction systems to make big brother obsolete. *Communication of the ACM*, 28(10), October 1985.

[4] S. Even, O. Goldreich, and Y. Yacobi. Electronic wallet. In *Proceedings of Crypto '83*, 1983.

[5] B. Clifford Neuman. Proxy-based authorization and accounting for distributed systems. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, May 1993.

[6] T. Okamoto and K. Ohta. Universal electronic cash. In *Proceedings of Crypto '91*, 1991.

[7] B. Pfitzmann and M. Waidner. How to break and repair a 'provably secure' untraceable payment system. In *Proceedings of Crypto '91*, 1991.

This research was supported in part by the Advanced Research Projects Agency under NASA Cooperative Agreement NCC-2-539. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any of the funding agencies. Figures and descriptions in this paper were provided by the authors and are used with permission. The authors may be reached at USC/ISI, 4676 Admiralty Way, Marina del Rey, CA 90292-6695, USA. Telephone +1 (310) 822-1511, email ari@isi.edu, bcn@isi.edu.