Open Network Handles
—
Rough Design Prototype
(DRAFT 0.3)

Michael J. O'Donnell

13 August, 2002

Abstract

I propose a system of *Open Network Handles* to provide permanent primitive network handles promiscuously to all who request them. Handles provide an intermediate level of service between IP numbers and domain names. While assignment of IP numbers is constrained by routing considerations, the owner of a handle may reassign it to different addresses over time for mobility or changes in configuration of resources. Unlike domain names, handles carry no significance in natural language, so they should not have high commercial value, nor should they attract disputes based on assertions of rights in significant names.

This document describes a prototype design for a network handle system, based on public-key cryptography.

0.1 First, I work out a design for the desirable system of the future, with a flat handle structure. I think that the design is pretty good, except for some remaining uncertainty about handle reassignment. I believe that details can be filled in to the sketch using the current DNS as a model for discovery, and public-key systems (PGP, GPG, etc.) as models for authentication. Later, I will consider hierarchical handles, and also interim plans to move toward the desirable future system.

0.2 I added a sketch of the hierarchical structure. While the basic flat design seems to be close to a very natural optimal point, the hierarchical structure seems less clearly constrained. In particular, I am not sure of the logistical and efficiency considerations in the different effects of the hierarchy on discovery and authentication.

0.3 I added a discussion of security for updates through auditing.

1 Basic Definitions and Principles

I laid out the definitions of *handle*, *address*, and *handle system* in "Open Network Handles—Definitions and Specifications." In this document, I refer to three sorts of agents:

Querier Any agent who uses a handle to communicate with the intention of communicating with its owner.

Owner The agent who holds authority over a handle.

Global System The abstract agent carrying out all operations other than those carried out by the querier and the owner. This is almost surely a highly distributed collective agent.

In particular, notice that the *global system* need not be centralized, and we may be able to design protocols so that it need not be highly trusted for many purposes. In a good design, handle owners and queriers may contract with trusted agents to participate in the global system and improve its performance on transactions that particularly interest them.

I refer to any communications or other operations that are not part of the system protocols as *out of band* operations.

Here are some key principles driving this prototype design:

- A handle system only provides a certain *continuity* of authority in a sequence of messages. It does not identify the agent wielding that authority, nor describe it, nor guarantee any sort of quality of the agent's behavior. The continuity of authority allows other communicating agents to accumulate information and confidence across a sequence of messages, but the origin of such information and confidence is always external to the handle system.

- Because a handle system only provides *continuity*, and not any guarantee of behavior, the integrity of the system is *not* degraded by any sort of bad behavior on the part of agents with respect to their own handles, nor by the failure of handle owners and queriers to follow its protocols. The integrity of the system is only degraded by interference in the use of a handle from agents other than the handle owner and/or from natural phenomena.

- The best design separates *discovery* of the address associated with a handle from *authentication* of that address as much as possible.

- The implementation of discovery must be efficient and reliable enough to provide a very high likelihood that each handle query is resolved correctly with a low latency, and a low rate of false resolutions. These are performance considerations, not logical correctness considerations. Discovery may be carried out with *best effort*, and need not be highly trusted.

- The implementation of authentication must be highly trusted. As much as possible, it should depend only on direct communication between querier and owner, and not on the global system.

- At some point, a design probably should also distinguish between agents taking responsibility for tracking the assignment of a particular handle, and agents caching assignments for local efficiency reasons. This distinction can clearly apply to discovery. Does it also apply usefully to authentication?

- The best design provides for operations by the global system to resolve handle queries into addresses, and only those additional operations required for the effective and efficient resolution of queries.

- A good implementation will almost certainly be highly distributed. In a highly distributed system, all messages should be as meaningful as possible in isolation, without the context provided by other messages. All messages should be treated as carrying presumably good, but not perfectly reliable, information, and should be amenable to authentication. Positive information (typically, the resolution of a handle to an address) may carry a certificate from the party with direct authority to determine that information. But negative information (typically, the unavailability of a satisfactory resolvent for a given handle) may only be certified voluntarily by those participants in the global system willing to do so.

- At present, public-key cryptography provides the only sensible paradigm for end-to-end authentication, so I will describe the design in terms of that paradigm. Other authentication methods can probably be substituted when and if they are invented.

- Whatever methods the system uses for authentication and other security concerns, the design should assume that these methods will occasionally fail. As much as possible, the consequences of failure should be local, and the agents concerned should be able to take steps to defend against those consequences. Defensive steps will mostly be out of band, but the system should provide the information that agents need in order to determine and execute a reasonable defense.

In this document, I am only describing the core operations of a handle system. A handle system is only useful when surrounded by other systems that interact with it (referred to here as *out-of-band* systems). The promulgator of a handle system may wish to implement some of these out-of-band systems in order to generate a satisfying level of use. For example, name-resolution systems such as DNS already exist, and they are crucial to the initial acquisition of handles. The promulgator of an early handle system may wish to offer a service that resolves abbreviated handles, which might be hash codes for actual handles, to facilitate the out-of-band acquisition of handles through human-mediated communications.

2 Querier's Protocol

In order to use the handle system, a querier must first acquire out of band a handle for discovery, an identifier for an authentication technique, and a public key for authentication using that technique. In general, we might allow less secure special cases (clearly marked as such), including completely unauthenticated handles, handles authenticated through a trusted authority (probably by password), and handles authenticated out of band (probably just a small number of handles important to the global system itself). In the last case, the global system will surely need some sort of primitive initial addressability to get started. It might be handy to implement that initial addressability through a small number of wired-in handles that are treated uniformly with other handles in query resolution.

If the querier retrieves the public key from some external lookup service, that is part of the out-of-band acquisition. Notice that only permanent caching of the public key by the querier can possibly provide authentication that does not depend on trusting the global system. The handle and the key may be the same, or the system protocols may prescribe some relation between them.

Here is the basic abstract structure of query resolution, from the querier's point of view:

1. The querier sends the global system a query containing the handle.

2. The global system normally responds with a resolvent address. In some cases, the global system may notify the querier that it cannot resolve the handle, or that the handle is reported to be compromised, or that the handle has been permanently reassigned.

3. The querier communicates with the putative handle owner through the resolvent address.

4. If the querier is unsatisfied with the authenticity of the response from the resolvent address, he may challenge the putative handle owner to certify the response with the private key corresponding to the public key that the querier holds.

5. If the querier is unsatisfied with the authenticity of the resolvent address or other information from the global system, he may challenge the global system to provide a stored certificate, which came originally from the owner of the handle. This certificate is authorized entirely by the handle owner. No additional certification by the global system is available, nor desirable.

6. If the querier remains unsatisfied, he repeats the query, possibly with a description of the unsatisfactory response (e.g., the querier might include the time stamp, and ask for a more current resolvent).

7. If the global system fails to provide a satisfactory response, the querier may challenge some agent(s) participating in the global system to provide

certificate(s) that there is no better resolvent (e.g., no resolvent with a more current time stamp) available or no other information (compromise, reassignment). This certificate carries no authority from the handle owner, only from those agents in the global system who are willing to provide it.

The list above gives a sequential order of operations, except that some operations may be conflated for efficiency (e.g., the global system may provide a certificate without being asked), and individual operations may be skipped as long as the other operations depending on its results are skipped as well. I think that these possible variations are easy to work out. The current DNS already provides an excellent distributed implementation of query resolution (the discovery portion of the work, essentially step 2).

3 Owner's Protocol

An agent becomes a handle owner by generating a random pair of private/public cryptographic keys, keeping the private key secret, and doing anything necessary to associate the public key with a handle (if the public key is the same as the handle, or determines the handle in an unambiguous way, this may be a null operation). The probability of different agents randomly generating the same key may be kept acceptable low.

Here are the operations that a handle owner may perform in pretty much any sequence:

- Send a time-stamped and certified revocable announcement of a handle/address assignment to the global system. Depending on details, this announcement may contain the public key in addition to the handle.

- Send a (possibly time-stamped, time-stamp possibly certified by a trusted timer) certified irrevocable announcement that the handle authority is reassigned to a new handle to the global system. This announcement may carry additional certification by a trusted party other than the handle owner who attests to some out-of-band authentication of the operation. The owner should only reassign a given handle to one other handle.

- Send a (possibly time-stamped, time-stamp possibly certified by a trusted timer) certified irrevocable announcement that the handle is compromised to the global system.

- Send an announcement (possibly time-stamped, possibly irrevocable) to the global system of an address and/or policy for notification of all further update operations, so that the handle owner and perhaps others may audit attacks on the handle. I discuss this sort of announcement a bit more in Section 5.

- In some (*extremely* rare in a successful implementation) circumstances, the global system may reply to a handle-owner's announcement with a response indicating that it lacks the resources to accommodate the announcement. Since a lack of resources might even lead to the inability to make such a response, handle owners should verify important announcements by querying the global system.

The time-stamp on announcement of a handle/address assignment need not be authenticated by anyone other than the handle owner, nor need it be accurate, nor need it use any global system of time. It might just be a sequential number. It is only used by the global system to choose between otherwise conflicting announcements. The time-stamps on irrevocable reassignments and notifications of compromised key should use a global system of time, since they may be compared with announcements generated by attackers.

The generality of addresses is open. They include at least IP addresses, possibly a null address, possibly revocable redirection to another handle, possibly time-dependent addresses to deal with mobility and/or predicted expiration, ... Addresses accommodated by the system should only include information that is essential for efficient system operation and/or for establishing communication with a putative handle owner. All information that may be as effectively and efficiently communicated directly from the putative handle owner to the querier should be left to such direct communication. Essentially all revocable information for the use of the global system in resolving queries may be bundled into the address abstraction for the purposes of the current discussion.

Handle reassignment may be used to transfer a handle, and also to upgrade to new cryptographic or other technology supporting authenticity. Since it is irrevocable, the time stamp is not crucial to its unchallenged use. But the time stamp, particularly if certified by a trusted timer, and the additional certification if any by an authority who attests to out-of-band authentication, may provide confidence in a transfer after the original handle is compromised.

A handle should be marked as compromised only when there is a reason to suspect that the private key for that *particular* handle has been cracked, spied, or accidentally revealed. The obsolescence of a whole method of authentication should be tracked separately. The use of announcements of compromised handles does not add any additional attack path, since an attacker who invalidates a handle by marking it compromised has already compromised it in fact.

The possible optional value of time stamps on announcements of compromise and reassignment arises when a reassigned handle is marked as compromised. The time stamps, particularly if they are certified by a trusted timer, may help queriers decide whether to accept the reassignment, or whether to regard the reassignment as invalidated by the compromise. The time stamp on a reassignment may also be useful when following reassignment of a handle whose authentication method is obsolete.

It seems clear that reassignment is not a perfect implementation of upgrade and transfer (due to the ineluctable obsolescence of cryptographic technology, and arbitrarily long time intervals between a given querier's uses of a handle), and may require additional out-of-band support. But it also appears that the reliability of any implementation of upgrade and transfer can only be improved by reliance on a trusted authority. I can't see any way to make upgrade and transfer reliable based only on communication between querier and owner. I think that the reliance on a trusted authority might as well be left open for now, and treated as an out-of-band operation.

4 Global System's Protocol

The global system should preserve the address assigned to each handle by the correctly certified announcement with the latest time stamp. If there are conflicting announcements with the same time stamp, we have to decide whether to make an arbitrary choice, revert to a default (presumably null), mark the handle as compromised. I favor the arbitrary choice for now. Announcements that are superseded or that expire based on their own addresses may be dropped, since they cannot affect query resolution. Announcements with incorrect certification may also be dropped (perhaps with notification to the handle owner for a security audit). If obsolete and/or incorrectly certified announcements are available in an archive, they may occasionally be useful for out-of-band auditing of a possible compromise.

The global system should preserve reassignments and announcements of compromise indefinitely, except that announcements associated with obsolete authentication techniques may be dropped after widespread announcement and time for upgrade by reassignment. Perhaps the global system should attach its own trusted time stamp to irrevocable announcements if they don't already carry one, but this would be the first case where we required trust in the global system.

In response to a query, the global system should return the address associated to the given handle by the correctly certified announcement with latest time stamp, or the reassignment if there is one. If the handle is marked compromised, the global system should report that in addition. If the querier challenges the resolvent, the global system should return the owner-certified announcement itself. If the querier challenges the lack or insufficiency (e.g., insufficient freshness) of a resolvent, an individual agent participating in the global system may return a statement certified by herself that no resolvent, or no better resolvent, is available.

An owner who follows the protocol will only reassign the handle to one other handle. If there are conflicting reassignments with trusted time stamps, the global system should prefer the *earlier* reassignment, with an unstamped reassignment taking last place. Should the global system give owner-initiated time stamps precedence over other trusted time stamps? Should it allow certifications with a trusted authority to call for notification in case of conflict? When reassignment is used for transfer, these issues become a bit delicate. For A to transfer a handle to B, B should if possible get a certificate from a trusted agent in the global system that it knows no current reassignment; A should announce an assignment, with a time stamp and third-party trusted certification acceptable to B; B should use a test query to ensure that this certified time-stamped reassignment is in the global system. Agents in the global system may be treading dangerously close to legal liability here, and we may have to trade off some a priori certainty in a transfer and leave remedies to legal action by B based on contract law. If possible, we should make minimal *requirements* on the good citizenship of participants in the global system, but allow owners and queriers to contract with particular participants for a higher level of service. In any case, B has excellent ability to *test* A's compliance by submitting test queries.

To what extent should the global system support and even initiate backwards traces of handle reassignment? For example when a reassigned handle is marked as compromised, should that fact be reported in response to a query to the target of the reassignment? Notice that the target of a compromised reassignment is not compromised itself as a handle, since hijacking of a handle before the first use is irrelevant. Only the use of the sequence of old and new handles as a single abstract handle is compromised. But the compromise of the old handle might be discovered after queriers have incorporated the reassignment. Perhaps the global system should report when a handle is the target of a compromised reassignment, and leave it to the querier how much auditing of reassignment trails to perform. We might want the global system to provide a backward reassignment lookup function as an aid in auditing.

5 Security for Updates Through Auditing

As much as possible, the responsibility for security should reside with the owner of a handle. Each owner has the best notion of the importance of security for her own handles, and the best incentive to either implement it herself or contract with a trusted provider. Authenticity and other security issues in owner-querier communications are already entirely the responsibility of those two agents.

But the global system may need to include some features to accommodate security in updates, to allow owners to defend their own handles against hijacking. Even though a querier may take independent steps to authenticate communication after a hijacked handle directs him to the hijacker, the hijacking has damaged the querier and the rightful handle owner by preventing correct discovery. If the querier tolerates the hijacking, then the handle owner may be further damaged by losing a desirable contact, perhaps a customer. A successful handle system should accommodate reasonable defenses against such hijacking.

The first line of defense against handle hijacking is the authentication of updates. It is important to allow a reasonably secure authentication method, presumably based on public-key cryptography, from the start. It is also important to leave the type of update authentication open, so we may plug in new methods as they arise in the future. The basic structure for incorporating authentication of update announcements is already in the protocol described above.

The only additional defense that I can think of is auditing. Handle owners may already perform audits by periodically querying the system, but we should probably offer notifications initiated by the global system, both to provide information that might be lost by the time of the query, and to reduce the message load. The global system should allow a handle owner to specify an address for notification of each update attempt. This provides the owner with an audit trail of update attempts to help detect and defend against attacks. The owner may perform audits herself, or may direct notification to a trusted contractor. Participants in the global system may also archive audit trails, and provide them to handle owners on request. The details of the global protocols for auditing require some more thought, but I am confident that we can find a good working design. Here are some considerations that occur to me:

- Owners may want to allow auditing with weaker authentication than updates, or even with no authentication. An owner who feels little or no privacy interest in the audit trail may wish to gain security by making that trail public. In particular, a public audit trail may provide a defense against an attacker who changes the audit notification.

- Some parameters of notification should probably be optionally irrevocable. Making notification irrevocable is a defense against an attacker who changes the notification. Irrevocability trades off flexibility for assurance. But irrevocable qualities may still be revoked in effect by reassignment. Should the system allow irrevocable outlawing of reassignment? Should it allow a handle owner to specify that certain irrevocable qualities are irrevocably inherited under reassignment? The latter feature might provide a means to attack a handle by reassigning another handle to it, so it requires careful thought to make sure that such reassignment with inherited qualities is accepted by the owner of the target handle. Without inherited qualities, no such acceptance is needed, since the mere use of a handle guarantees nothing at all about the querier. Perhaps the right approach is to allow reassignment only to a target handle that already has the appropriate irrevocable qualities.

- Notification may also be used to strengthen basic security. The global system should probably support an optional enforced delay, and optional required confirmation, before executing an update. This is particularly important for an irrevocable update.

6 Hierarchical Extension

The design description above treats handle space as a flat space, with no inherent structure, and no implicit internal structure other than reassignment. In practice, we would probably like a hierarchical system of responsibility for discovery, removing some query load from the participants in the global system. We could develop a hierarchy completely post hoc: each handle may lead to a subsidiary handle system, completely independent of the global one. But there probably should be protocols that allow the most transparent possible operation through all the levels of hierarchy, so that most or all user-level operations can ignore the hierarchy and treat handle space as flat again. To take full advantage of this hierarchy, some or all handles at some levels of hierarchy should be unauthenticated, since the authentication of the local system covers them.

DNS includes a very successful design and implementation of hierarchical handles, which is used administratively in handle assignment, and technically in discovery of IP numbers. But it is not immediately obvious how the authentication component of the open network handle system interacts with the hierarchy. A hierarchical system of handles essentially allows us to manipulate certain subspaces of the entire handle space differently.

To derive a good hierarchical handle design, consider the different relationships that an agent might maintain with a particular subspace of handles:

- An agent may *claim authority* over a subspace of handles;
- An agent may *take responsibility* for supporting discovery of IP numbers for handles in a subspace;
- An agent may *voluntarily aid* in discovery of IP numbers for handles in a subspace.

The current DNS system deals with voluntary aid in discovery very nicely through the local caching of handle/IP pairs. Caching does not require any lasting relationship between the agent supporting the cache and the particular handle subspace. I see no reason to change this. Notice that an agent who claims authority over a subspace has a natural incentive to support discovery within that subspace. So, I propose to allow agents to *own* handle subspaces for which the agent both claims authority, and takes responsibility for supporting discovery. It seems that, whenever these functions need to be separated, the separation may be achieved by having the subspace owner contract with another agent for discovery support.

In a hierarchical handle space, initial ownership is inherited down the hierarchy. The owner of each handle automatically becomes the initial owner of the hierarchy below that handle. For the most part, that means that the owner of a global handle controls and supports all of the hierarchy below. But, the owner may assign subhierarchies to subagents, employees, contractors, or other agents, implementing that assignment by her own methods. The global system need take no notice of such independent local assignment of authority and/or responsibility.

Authentication of an actual connection takes place after discovery, through direct communication between querier and owner, so authentication keys are needed at the bottom of the hierarchy for this purpose. But, authentication keys are also used to establish authority over the root of a subhierarchy. So, the hierarchical structure on handles seems to call for one additional type of authentication associated with a handle: authentication *inherited* from the parent. The inherited-authentication type only affects the authentication of direct communication between querier and owner. The owner of a subhierarchy is the one to impose authentication on updates to its own subhandles, and to sign answers to queries about those subhandles that reach it.

So, in a hierarchical handle space, we may almost always resolve a query on <*subhandle*>.<*global-handle*> by forwarding it to the owner of <*global-handle*>. That owner has full authority and responsibility to determine the resolvent of the subhandle, and to support discovery of that resolvent. As in the current DNS, other handle servers will normally cache resolvents of subhandles, and optimize future discovery by avoiding repeated appeal to the owner of <*global-handle*>. This caching depends on the transparency of the hierarchy. A handle owner is at liberty to implement any sort of service, including another handle system, opaquely at the resolvent address, but the implementor of an opaque subsystem forfeits the benefit of local caching by other agents who cooperate with the global system, and the ability to reassign local handles through the global system.

The storage of subhandle resolvents by agents outside of the global handle owner is almost always just a voluntary act to improve system performance, and such resolvents may be deleted at will. But a reassignment of a subhandle to an outside agent requires global support. At least one globally trusted agent should take responsibility for preserving each such reassignment. This is mildly annoying, but it requires at most a minor change to the existing bind software for DNS, and global support for subhandle reassignment avoids one incentive for acquiring multiple global handles, and encourages agents to take local responsibility for local handle subhierarchies. If reassignment becomes important, this is a particularly likely area for contractors to support particular reassignments for a fee. Even if the global system supports reassignments at no charge, such contractors may be able to improve efficiency and reliability.

7 Transitional Systems

The sections above are intended to describe the design of a worthwhile network handle system for long-term use into the indefinite future. I think that most of the implementation details for discovery are already worked out by the current DNS, and for authentication by cryptographic systems such as PGP and GPG.

In the long term, software derived from the current bind package should probably be adapted and optimized for the particular loads experienced by the handle system. In the short term, handle discovery can be implemented directly on the current DNS. It just needs a sponsor who owns a domain `sponsor.org` and is willing to support subdomains of the form

<*textual-handle-number*>`.handles.sponsor.org`

and an associated database with any information that doesn't fit in current DNS tables. This DNS-based implementation makes the use of handles completely transparent to users who follow links in Web browsers and send email to addresses stored in address books.

Rather than implementing an explicitly different initial structure for the handle system, and trying to update it in place to the long-term structure, I propose to implement exactly the long-term logical structure, and surround it with out-of-band services to ease the transition. Each of these out-of-band services should be understandable in one of two forms:

1. Provision of an external service to perform some operation that is logically part of the querier's or owner's protocol. Agents who use such external services must be aware that the integrity of the results may now depend on the reliability of the external service.

2. Provision of services out of band. These services are completely outside the handle system protocols, but make it easier for some agents (particularly naive agents) to use the protocols productively.

To ease handle ownership by naive agents, we might provide a proxy service that generates key pairs, and posts announcements for those naive agents based on a primitive password authentication. Such a proxy might save the private key for possible later surrender to the proper owner (with acknowledgment of the security risk in doing so). Or, if the proxy is appropriately integrated into the system, it might discard the private keys (which could provide slightly less security degradation), and resolve password protected handles with a clear marking that they are only password protected by the system. If private keys are discarded, owners may still upgrade to new secured handles by reassignment. This is an example of the first sort of external service.

To ease out-of-band handle acquisition by naive users, we might want a parallel set of subdomains of the form

$<$*handle-abbreviation*$>$`.abbreviations.sponsor.org`

linking each (relatively short) handle abbreviation to the corresponding (very annoyingly long) handle. Handle abbreviations should be assigned automatically from handle values, probably by a hash function. They should be comparable to telephone numbers in ease of copying and remembering, but meaningless commercially. Perhaps the right length is 12 decimal digits, nestling between 10-digit telephone numbers and 16-digit credit card numbers. In particular, abbreviations should be generated by a method that has a very low probability of colliding with a trade name. Instead of looking like numbers, they might use character strings from a good meaningless password generator. Users should be encouraged, particularly by the automatic behavior of application software, to look up an abbreviation externally only once, storing the associated handle in a link or bookmark or address book entry instead of the abbreviation. This is an example of the second sort of out-of-band service.

Naive queriers are at liberty to trust the resolvents returned by the global system, with no independent authentication through the public key. Such an additional reliance on the system (logically another example of the first sort of external service) doesn't require any change to the system behavior or any additional implementation of an external service. It is implemented purely by the naive querier through its own exercise of options in the handle system protocols. There's only a small constraint here: the global system must return an address in plaintext to save a naive querier completely from having to apply the public key.

It is very possible that many queriers will continue indefinitely to trust many handle resolvents without end-to-end authentication. Incorporation of automatic authentication in software may raise the rate of performance of authentication, but the precise behavior of future queriers is impossible to predict. In some cases, queriers may feel that they risk very little damage through misdirection, since the transactions that they perform are safe because of some out of band considerations. Notice that handle owners may be damaged by misdirection even when queriers are not, since they may have lost valuable contacts, such as customers. But we have reason to hope that the availability of end-to-end authentication improves the practical reliability of discovery a lot, even when the authentication is very seldom exercised. Notice that a similar thing happens with hand-signed transactions on paper. Hand signatures are very seldom verified in any serious way. But their presence, and the potential for verification, serve as a deterrent to fraudulent behavior in at least two ways. First, the potential for verification exposes a fraud to some probability of discovery. Second, the provision of a signature signals the signer's seriousness, and increases the socially imposed punishment for a discovered fraud.

8 Acknowledgment

The essence of the public-key handle method given here was proposed by an anonymous post to the *ICANN Watch* discussion. If possible, I will credit the author later. The structure of the transitional implementation through DNS came from Bob Frankston's article, "DNS: A Safe Haven." In a conversation with Nick Russo, he brought up the example that made me realize the value of reassignment of handles from lower levels of the hierarchy. I'll work up a proper bibliography later.