# FIA Investigator Meeting

May, 2011, Berkeley Ca

Meeting summary

Version 3.1 of March 30, 2012

The focus of this meeting was on aspects of system architecture relating to security, trustworthy operation, and related issues such as privacy.

This report was prepared from the slides used during the presentations and from an audio transcript of the discussions that followed. Instead of reporting the comments of individual speakers, the material has been re-organized around issues and topics. The goal of this reorganization was to provide a better structure to the report, and to deal with the fact that the nature of the recording did not permit the correct attribution of comments to speakers in all cases. However, in a few select places where a quote was obviously associated with a specific speaker, it has been given attribution.

This report was prepared by David Clark, who takes responsibility for its actual content and organization.

## Defining security

As a way to help presenters prepare their remarks and to make clear the focus of the meeting, the planning committee for the meeting prepared a list of possible questions that each project might choose to answer, questions that tried to capture the set of issues that fit under the heading of "security".

1) How does your design deal with attacks directed at the network itself? Examples: attacks on routing protocols, management protocols, physical attacks (e.g. coordinated fiber cuts). Supply chain issues.
   Does your design assume that the network is built out of separately administered regions, some of which may be malicious or uncooperative?
   Does your design tolerate rogue ISPs and corrupted network elements? What degree of trust is expected among the regions and how is this developed and maintained?

   1a) To what extent does your design emphasize resilience/availability of the network in the face of attack, as opposed to resistance to attack?

2) How does your design deal with attacks on communication among end-points? Examples: unwanted disclosure and modification of content (person in the middle attacks). Redirection attacks (rerouting to rogue end-points).
   Does your design include a framework for communicating elements to

identify each other? Does your design deal with issues of trust (or lack of trust) among communicating parties?
Does your design place controls on (or require authorization for) various sorts of end-point actions across the net?
Does your design deal with selective/targeted loss of availability/performance?
What is your point of view about traffic analysis? Does your design store user or transactional information?

3) How does your design attempt to mitigate attacks on end-nodes?
Examples: delivery of malware as part of communication, direct attacks on vulnerabilities presented via the network interface, espionage and data exfiltration.

3a) Does your design include methods to deal with multi-stage attacks (using intermediate nodes previously corrupted)? For example, do you intend to address prevention, diagnosis, and/or deterrence of multi-stage attacks?

3b) Does your design attempt to mitigate the consequences of end-node faults and vulnerabilities?
Examples: requirements for application design. Restrictions on patterns of communication.

4) Does your design deal with information assurance?
Possible goals: information integrity, authenticity (provenance), detection of stale versions. Integrity of search.

5) How does your design deal with DDoS attacks? (Such attacks could be seen as attacks on the network and attacks on the end-node, but probably worthy of a distinct category.)

6) Does your design address the sorts of higher-level considerations listed below? (These could be seen as covered by the list above, but should be noted as named issues.)

6a) National security: lawful intercept, traffic analysis.
6b) Privacy; anonymous action. How does your design relate to privacy-enhancing or anonymizing tools? Does your design raise (or mitigate) issues related to correlation across stored information?
6c) Control of delivery of illegal or forbidden content.
6d) Accountability; deterrence. Does your design intend to make jurisdictional boundaries explicit?
6e) DRM and content management.

7) What new sorts of vulnerabilities does your approach create? What are the "new" security problems you will have to resolve? Can you identify points of

(security) failure with severe implications?

8) Has your group agreed on a threat model? What is your point of view?

## The format of the meeting

The project participants were joined by two teams of observers: an invited set of experts in security, and members of the Values in Design Council.

The security experts:
Alan Kirby
Susan Landau
Jim Fenton
Drew Dean
Earl Boebert
Lee Badger

The participants from the Values in Design Council:
Paul Ohm
Deirdre Mulligan
Chris Hoofnagle
Jacob Gaboury
Batya Friedman
Geoffrey Bowker

Each project gave a presentation on the security aspects of their design, which was then followed by an hour of discussion of the approach. Following those four presentations/discussions, the two groups of observers were invited to give overall reactions and thoughts.

## Project presentations/discussions

### Extensible Internet Architecture (XIA)

The relevant aspects of the XIA architecture are as follows.

- The rich addressing/forwarding mechanisms in XIA allow a packet to carry several forms of addresses at once. For example, they can carry the content id (CID) of a desired piece of content, but as well the ID of a service hosting that content (a SID), or the host where the service is located (a HID) or an administrative domain in which the CID is known (an AD). This richness is described as expressing *intent*, and the other addresses allow various forms of *fallback forwarding* and the like. In detail, these various IDs are structured as a DAG in the packet header, which the router parses to find a suitable ID on which to act.

- The various IDs, collectively called XIDs, must be *self-certifying*. For example, they may be the hash of a public key, or the hash of the content to which they refer. These ID mechanisms allow the end-points (e.g. the applications or perhaps code acting on behalf of the actual human users) to confirm that the action they attempted has completed correctly: that they connected to the host they intended, got the content they intended, and so on. Put otherwise, these mechanisms transform a wide range of attacks into detected failures.
- The architecture of XIA will attempt to provide tools to isolate the source of the attack/malfunction that caused the failure. While these were not elaborated in the presentation, the XIA team assumes that they will sit above the architecture—diagnostic tools that can be run as needed.
- XIA gives the end-point options for control to bypass or route around these points of failure. These include the detailed routing choices in SCION, the diversity of options for binding human-oriented names to XIDs, classes of XIDs that support replication and caching, and the like.
- The SCION mechanisms provide a structure to break up the overall network into what are called *trust domains*, which allow a variety of end-point controls over routing.

Requiring that XIDs be self-certifying is part of the mechanisms that allow confirmation that the intended action has actually occurred. However, this step is only part of the scheme. XIDs provide a means to confirm that the correct action occurred *once the end points have the correct XIDs*. As a first step, most network operations start with "higher level" names that describe what is desired, such as URLs, email addresses, and the like.

- Since different applications may involve different sorts of high-level names, the XIA architecture does not define how these names should be converted to XIDs in a trustworthy way. The XIA architecture gives mandates to application/support service designers as to what the application and its supporting services must do, but does not dictate a mandatory way of doing it.

The class of mechanisms that an application uses to provide these binding is called *trust management mechanisms*, and XIA provides an API (a "narrow waist") for an application to invoke these.

## Discussion

In the discussion that followed the XIA presentation, the following issues came up.

**Security implications of intents.** While *intents* may lead to more efficient network utilization, and contribute to the goal of availability by finding nearby copies of content, finding an available service provider (a form of anycast), and the like, they may have serious consequences with respect to privacy and traffic analysis. A visible CID in the header seems to provide a very fine-grained revelation of what a specific user is doing, and the source HID in the packet can tie the action back to a user.

One response to this is that the source (and the application acting on behalf of the source) has the option of using a persistent HID or a short-lived one. Hosts can have more than one HID. If the source domain can route on an *ephemeral SID* (sort of like the random source port in the current Internet), then there need be no source HID in the packet, which can partially obscure the identity of the sending host, and also allow process migration. Another answer is that the full expressive power of intents need not always be used: the packet could have only an SID, with (for example) a CID encrypted inside the packet so that only the end-node can see it. However, these choices may be defined by what the application designer does, not the actual user at the moment of execution.

There was a question as to whether the DAG addresses could permit certain sorts of abuse, such as making a packet seem to come from the wrong location. It was claimed that one problem with source routes are the security vulnerabilities they create, and DAGs seem like source addresses. One answer is that DAGs are less general than source addresses, and as well some sort of policy restriction may need to be imposed on the sorts of DAG addresses that can be used in specific contexts. This seems to imply a complex bit of analysis as to what sorts of DAGs represent risks in particular contexts.

**Who is the user?** Several of the slides in the XIA presentation referred to "users". In some cases, this may refer to the actual user at the time of execution, and in other cases refer to the "user of XIA" who is the application designer. Clarity of distinction between the two is important, as (for example) in the discussion above of which "user" has control over how much is revealed in the packets. If the project really wants to look at "users", they must look at real users: kids with games, the third world, the people at the edges.

**Key management.** Several forms of XIDs, in particular HIDs and SIDs, use public key/private key pairs as the mechanism for self-certification. This approach implies a number of issues specific to the design. If a host is penetrated and the private keys are stolen, there needs to be some form of revocation. It is possible that the trust management mechanisms are part of this (e.g. binding may time out), but this needs careful consideration. It was also pointed out that the "dog ate my private key" problem is critical and hard to solve. The more copies of the private key are around (both to protect from total loss and allow replication of services) the more chances of it being stolen. The dependence on key pairs as the foundation of XIA security may imply some very daunting operational requirements. See below.

**Classes of principals.** In XIA, elements identified by XIDs are *principals*. There was a question as to whether content, service and host were the only ones, and what was required to make a new one. The team responded that the only reason to add a new class of principal at the XIA level is that it allows the network to provide a service (e.g. routing, caching) at that level. Otherwise, the function should just as well be done at the service layer. The term "overlay" was used, but the relationship between overlay and service was not clear. The team speculated that new classes of principals would be invented infrequently, since the XIA infrastructure would have to evolve for the new class of ID to be useful. Two possible examples of future classes were DTNs and multicast. The team invited the listeners to think of possible new classes to challenge their thinking. SCION routes can be integrated into XIA as a new type of XID.

There was a question as to why the host is a class of principal, since it seemed as if services were the real building block. There were several answers: HIDs will be useful for low-level issues such as management. HIDs provide the migration path from today's host-centric Internet. Finally, once a stateful connection to a service has been established, a HID can be used to select the particular version of the service that has the state. However, this could perhaps be done with an ephemeral SID on the server side, which would enable the process migration mentioned elsewhere.

**Traffic analysis.** In the list of security objectives for XIA, privacy was mentioned, but there was no specific mention of traffic analysis. Several of the invited discussants strongly made the point that traffic analysis (looking at headers, not content) is a very powerful tool for intelligence. Patterns of interaction reveal a great deal, and can be a means to break *operations security* or *opsec*, and discover that "something new and different" is going on. As a means to model what the adversary can do, one should not begin with a catalog of mechanisms, but a catalog of the potential adversaries that can observe aspects of the traffic. One should list what they can see individually, and then ask what is revealed if they can share knowledge from different parts of the network.

One of the team members observed that some of the XIDs can be transient, including the HID of the host retrieving content. However, it would seem that CIDs, which are a hash of the content, must by design be persistent, which would facilitate the long-term tracking of the dissemination of a known piece of content over XIA.

**Privacy and accountability.** Some of these same issues came up in discussions about privacy and accountability. The XIA team commented that one of their design questions is the extent to which this balance should be in the infrastructure vs. in the applications. There was a comment from the room that user behavior may be different if users are identified and held accountable, and the team said that Sara Kiesler was researching this question. However, this did not resolve the question of XIA vs. higher-level mechanisms. The team commented that in their view, it would be desirable if XIA could be configured to provide a different balance (at the XIA level) in different deployment contexts. This is a harder challenge than a fixed outcome.

**Trust domains and scopes.** There was considerable confusion as to what a trust domain was. Given the different uses of the word *trust* in the discussion, a number of different objectives could be presumed. Mechanically, within the SCION system, trust domains seem to be routing regions that are collections of administrative domains (ADs). Within a TD, an end node can express controls/preferences over how routes are formed between the end-node and a *path construction beacon* (PCB), which is a point within the *core* of the TD where ISPs (sort of like tier 1 ISPs today) patch routes from end-nodes together and compute routes to a PCB in another TD as necessary. This seemed to leave a number of questions unanswered.

- What does this have to do with trust? Are we "trusting" that the specified route will be followed between the end-node and the PCB?
- Does this design imply that the end-node must trust its access ISP? There are many real-world situations where the users must use an ISP they do not trust.
- TDs were described as matching legal jurisdictions (as an example). Why is this the right modularity? Does the design depend on legal sanctions to discipline the operators to behave in a trustworthy way? Why would there be bigger or smaller TDs?
- There was a PKI associated with TDs, but the role of the PKI was not clear.

In the subsequent discussion, it was made clear that in the XIA design the TDs are only concerned with the issue of path availability, and no other aspect of trust. Within a TD the users have control over AD-level routing (among the valid routes they have been offered), and if traffic is between a source and destination in the same TD, the end-points can be assured the traffic did not leave the AD. But it is not clear what assurances are available when the traffic leaves the TD. This concern was raised in the context of increased use of "cloud-based applications". Many countries are too small to sustain an internal cloud infrastructure, and many popular applications (such as Facebook and Youtube) are almost global in scope. This fact raised doubts that most traffic will remain inside a TD, which in turn raised doubts about the power of the TD mechanisms to improve overall operational security/availability/control.

One of the roles of a TD is to isolate routing and other control messages. Spurious routing messages from outside the TD cannot disrupt the internal routing of the TD. One way to explain this that in the current Internet we have two levels of routing, IGP and EGP, where the EGP should not be able to disrupt the IGP, and in SCION we have three, the AS IGP, the TD and the global, and at each level, the outside should not be able to disrupt the inside. This raises a number of questions, such as whether ISPs inside different trust regions can directly peer.

XIA permits the use of mechanisms such as TOR, and the SCION work includes a TOR-like mechanism. This can be used to tunnel out of a region where there is no trust. However, the interplay between lack of trust, the use of *intents* and traffic analysis seems complex and nuanced, and may require careful thought as part of the design. Again, good security in practice may require a high-level of attention to operational issues, which ties back to the question of "who is the user?" and what level of sophistication is required of each class of user.

There was considerable doubt about the use of "scopes" (parts of the net in which more or less is revealed) to protect the user from traffic analysis (and complete profiling) if the analysis can be carried out over any extended time period. And given the diverse patterns of trust (I trust my ISP but not yours, or the other way round), the interplay between the use of something like TOR to hide information and the desire to reveal intents only in trusted regions seems more complex than the XIA mechanisms could realize.

The idea that the application designer has control over how these tools are mixed (e.g. how scoping/tunneling is done) raised concerns. There is no reason to assume that the application designer always represents the interests of the user, or that the user can understand the consequences of complex behaviors that are embedded in the application. Something more is needed as part of a complete story.

**Performance:** Since this meeting was about security, performance came up only indirectly. But in the discussion about the performance of long distance connections, it was asserted that tier 1 ISPs carry most of the long distance traffic, and on this basis, SCION paths were not materially longer than BGP paths. However, the current trend in the Intenet is toward private peering among access and content, which raises the question of how SCION deals with paths that cross TDs but are private arrangements among the ADs in the two TDs.

On the other hand, the XIA team noted that data is migrating to follow the user, and the CIDs may allow local retrieval of content. However, there is a general question as to whether the efficiency of this routing scheme is dependent on various assumptions about traffic patterns.

**Verification:** What approach will be taken to reason about the correctness of the system and its properties. It was agreed that long, formal proofs would not be effective. But is there some sort of semi-formal process, perhaps some sort of diagram or representation that can help. The team noted that as a practical matter, verification can take as much time as design, and the scope of the project limits what can be done.

With respect to trust, one suggestion was to construct some sort of matrix of principals and actors, and describe the trust relationship between them. What would the consequences be of violating this trust?

**Evolution:** Following on the discussion of XIDs and the possibility of adding new ones over time, the question was raised as to whether the encryption algorithms can be changed over time. If one is found to be flawed, or is rendered vulnerable by Moore's law, how much of a problem is this for the architecture?

**Regulation and transparency:** The sophistication of the addressing and the range of trust management mechanisms seems to give many more actors options for control, perhaps very nuanced control, over what happens in the system. The debates today over network neutrality are about imposing external restrictions on what ISPs can do, in a context that is much simpler than XIA. For example, in XIA will there need to be potential regulation over which content is cached? As a starting point, XIA should provide means to make the control actions take by different actors visible or transparent. Visibility is one of the means to discipline actors.

## Summary—XIA

**Control of XIA addresses:** in several different parts of the discussion, the question came up as to which actors would have control over what is in an address: whether there is a CID, for example, or a HID. It seemed as if some of the answers were inconsistent. One answer was that the application designer had control. Another answer was that the "user at the source" would have control, perhaps to control whether a cached copy of content was acceptable or whether the request has to go to the service. It was stated that different addresses (including options for tunnels and selective revelation of parts of the address) might vary based on the trust that the "user" placed in various parts of the network. Since this happens at execution time, this is not the application designer but the user or an agent helping the user. Finally, of course, what is in an address is determined by what is returned from the "name lookup function". It would be useful to write down in one place all the factors that have to be taken into account in fixing the address being used, and making sure that all these uses are consistent.

**Vulnerabilities:** There were two aspects of the system that seemed as if they might represent points of vulnerability: key management and the trust management mechanisms that map human names to XIDs. Key management is tricky and complex, and given that the project said that they were not addressing the issues of host security (which seems reasonable) it is not clear how this system can be secure in practice. If it is to be anything but a toy system, the key management tools must be designed as a part of the system, and a team participant that plays the role of the adversary must be part of the design process from the beginning. The tools that map from name to XID are outside the architecture, and by design there will be multiple of them. If they are vulnerable, the attention at the XIA level will be fruitless. Is there some sort of "end-to-end" check that can validate the human-meaningful name as well as the XIDs?

## Nebula

With respect to security, the Nebula approach is based on the presumption of a clean division of responsibility between network and end-points. The end-points are responsible for confirming that the connections have reached the correct places, and that integrity and confidentiality are preserved (e.g. by encryption). The responsibilities of the network are highly reliable availability, predictable service, and assuring that the requirements (policies) of all the relevant actors are taken into account as traffic is routed across the network. The relevant actors include networks themselves, and as well higher-level service elements.

An example, as developed in the group discussion, will help to illustrate what is meant by that last responsibility. Nebula, like essentially all architecture proposals, assumes that the network is made up of regions that are separately built and operated, often by private-sector providers. These providers will have policies concerning which sorts of traffic (e.g. for which classes of senders and receivers) they will carry, and so on. Today, the only tools that can be used to express these policies are the various options within BGP, which may be very limiting. In addition, the application may want to control the routing of traffic between higher-level service elements. A simple example might be a "packet scrubber" that tries to detect and remove malicious traffic, or a higher-level processing element such as a virus detector in email. A service might wish to assert that it will only receive packets that have first gone through the scrubber, even though the scrubber is not directly adjacent to the service. In Nebula, the network itself can enforce this sort of routing policy.

To do this, the Nebula architecture has two relevant parts: a data plane (NDP) that can enforce arbitrary policies, and a control plane (NVENT) that can compute these policies. While the control plane is still under development, there is a specific proposal for the data plane, and a claim that it can enforce arbitrary policies with respect to valid routes through sequences of actors.

Nebula is not a pure datagram network—to send a packet the NVENT policy mechanisms must first compute and return to the sender a string of information that will authorize the data plane to forward the packet. NDP is "deny by default"; without this information to put in the packet, the packet will not be forwarded. What is returned by NVENT is a "proof of consent" (POC), which is a cryptographically signed sequence of values that (for each step in the processing/forwarding) encode all the previous steps that must have processed the packet before this actor receives it. Clever use of crypto and XOR allow this to be coded in a way that is linear in the number of actors. As the packet is forwarded, each actor that processes the packet computes, using a similar set of methods, a "proof of path" (POP), which allows each subsequent actor to confirm that the previous step actually did process the packet. Thus, by comparing at each stage the POP at that point with the POC that was computed by NVENT for that stage, the NDP can confirm, for each actor in the path, that the packet has already passed through all the required previous actors.

Nebula can improve reliability by allowing the sender to ask NVENT for multiple routes (e.g. multiple POCs), so that if one fails another one can be tried. Or for improved performance several could be used at once. So one of the consequences of the rich set of policy options that NVENT can handle is a much richer set of routing options than the single option computed today by BGP.

### Discussion

**Proof of path.** In the discussion, there was a lot of attention spent on clarifying exactly what the POP actually confirmed. The Nebula team made clear that all it was intended to confirm is that the correct set of previous actors has signed the POP. It cannot confirm the actual path: if some ISP sends the packet via a second ISP that was not in the POC, but which forwards it anyway, or if the packet is copied, and so on, the POP cannot detect or prevent this.

**Mechanism.** One set of questions in the discussion concerned the actual mechanisms by which keys are managed and used. However, the discussion did not allow time to fully explore this issue.

**Integration of cloud and network.** Another set of questions explored the assertion that a distinctive aspect of Nebula was that the cloud and the network are working together. It was not clear from the talk exactly what this implies. The highly redundant paths that NCore can provide between the data center and the router are an obvious example, but it was not clear how this synergy manifested in NDP and NVENT. One example that the Nebula team gave was that the rich policy options of NVENT could allow an application to obtain a service with a higher degree of performance, reliability, availability and so on, which would allow the application to move data in ways it would not otherwise consider reasonable. So the potential richness of the NVENT policies can change the way applications are designed.

**Consequences of rich tools for policy control.** Several of the questions revolved around the consequences of the rich control implied by NVENT. The optimistic consequence of NVENT is that the end-user can find paths that meet his needs for performance, trust and the like. The pessimistic consequence is that the options for control give network operators and third parties rich tools to control, selectively block, observe and regulate what happens on the network. The general question is how can a control plane such as NVENT, which was not described in any detail, be designed so that the desired parties have sufficient control. The specific example in the question related to the case of the insulin pump: how can "the system" confirm that HIPPA is observed, the service of sufficient reliability is actually provided, and so on.

A related question had to do with surveillance, including in the control plane. What is revealed there, and is every aspect of the service request revealed to all the actors along the path? One questioner asked if this did not make traffic analysis much easier. In general, what can be learned from observation, both in the data plane and in the control plane? The team responded that highly decentralized systems can make traffic analysis harder, but cannot prevent it, especially since today we see powerful actors such as Google with monitors on 80% of the Internet.

With respect to censorship and its mitigation, the Nebula team described a vision in which a large number of web sites could set aside a fraction of their resources to allow relay access to other sites. This approach would allow someone trying to bypass censorship to "blend into the crowd" by appearing to be going to an innocent site. The controls in NVENT might allow the web site to distinguish between its traffic and traffic it is relaying, but this in turn raises questions about which other agents in NVENT could see that distinction and recognize this as a bypass.

**Who is the user?** The role of the user came up in the answer to several questions. In some cases, it seemed to imply the application designer. In other cases (as with different users with different concerns about the privacy of their medical records) it was the actual end user. This tension raised two concerns. The first is whether the typical end-user could translate between high-level expressions of preference (e.g. more or less privacy) and the resulting inputs to NVENT. The other is what the interplay was between the application designer, the end-user, and other actors that might want to have control on behalf of the user. There was also a question as to how an end-user could look at the address (e.g. the sequence of POCs, labels and the like) and understand what they meant—what service they actually described. This system seems to raise some substantial user-interface issues. Finally, it was observed that the complexity of dealing with the options in NVENT might tempt most application designers to take the default option, and end up with routing equivalent to BGP.

**Patterns of communication.** While the discussion tended to focus on the packet level of the architecture, the question was raised as to whether, and in what ways, this system might change the higher-level pattern of communication. As Batya Friedman said: "Any design makes some things easy, some things hard, and some things impossible. Is there a way to look at this system and decide what behaviors are in those three categories?" This question relates, among other things, to the discussion above about censorship and bypass.

**Cloud policy—jurisdiction, antitrust, and the like.** This question related to data being stored in the cloud, and what policies might be enforced to regulate data movement. For example, could the POC/POP be used to determine what jurisdictions the data had visited? Does the network facilitate data portability, or allow companies the secretly exchange information and collude in the cloud? The discussion suggested that while NVENT is designed to help with moving data, policies related to the proper treatment of data were to a larger extent must be at a higher level.

**Overhead:** One question concerned the cost of the big headers, especially for small packets and TCP acks. The team suggested that many of the packets today are large (e.g. video) and the system need not use TCP, but perhaps an alternative with fewer acks.

**Additional questions:** Since paths seem to be pre-computed in NVENT, how does this affect mobility of end-nodes? Does Nebula contain tools to validate that paths are working as committed? Perhaps a user could create several paths and try them to see which is most trustworthy.

## MobilityFirst

The MobilityFirst architecture is motivated by the desire to deal with issues raised by mobile end-nodes—in particular movement of devices from one network access point to another, and transient outages when devices become unreachable. In this architecture, naming/addressing is done at two levels. At a higher level, there are a set of alternative Naming Certification Services (NSs) that map from some host, service, sensor, content or context (a context is a set of things that match some criteria) to a flat ID, a GUID. (A GUID also contains a Service ID, or SID, which is similar to the TOS field in IP in that it contains well-known numbers mapped to service descriptions such as unicast/multicast/anycast, delayed delivery, privacy-preserving, content query, etc. The SID is used by routers to decide how to process the packet.)

At a lower level, there is a service, the Global Name Resolution Service or GNRS, that maps from a GUID to its current location, which is a Network Address, or NA. A network address has a network part NA:N and a port part NA:P.

The network data unit is not a packet, but a larger data unit (sort of like an Application Data Unit) called a PDU. When a PDU is sent, both the destination GUID and the destination NA are included in the header. This allows rapid forwarding based on the NA:N, but also allows elements in the network to deal with mobility and redirection by making dynamic queries of the GNRS as the PDU is moving through the network.

Both the GUID and the NA:N are public keys (or hashes of public keys), so the namespace of NAs is flat. Their design assumption is that there might be as many NA:N values as there are routing table entries today. When the GNRS is queried to look up the NA associated with the GUID, what is returned is actually a "locator", which is an assertion signed by the owner of the GUID that the NA is valid. This allows for the prevention of certain forms of attack, such as a "backscatter" DDoS attack, in which a rogue endpoint R sends SYN packets to machines all around the network with the source address (R,NA), where NA is the target of the attack. All these machines respond to the SYN with a packet to NA, which has to process and drop them. Allowing the end-point verify a signed locator for NA prevents this sort of misbehavior. Further, since the locators are signed, and thus "self-validating", they can be cached and retrieved from anywhere, with the assurance that they are not forged.

The actual process of transport breaks the PDU up into packets, which are transported using a protocol perhaps like TCP (but tuned to the failure and loss modes of wireless systems). PDUs are forwarded in a staged manner, and may be reassembled from the component packets at intermediate points. If a subsequent network is impaired or the destination is unavailable, the PDU can be stored until forwarding is possible.

MobilityFirst, like XIA, tries to avoid the problems that have arisen in the current Internet with the single root of the DNS, by allowing multiple Name Certification Services to co-exist.

## Discussion

**Defining "User":** As with XIA, there were a number of questions about what was meant by "user", and what was expected of the user. In response to a question as to the role of the user, the MobilityFirst team pointed out that they cannot verify the physical identity of the user operating a device—if the device has the credentials to confirm its GUID, they will take it as valid. Users can have more than one device, and devices can have more than one interface: the GUID is in common, the NAs differ. So the architecture tries to give a clean identity/location split.

Another conversation centered on the role of the user as giving trust guidance and choices. Today, users have many trust choices made for them (e.g. the list of CAs in the browser), and the MobilityFirst team would like to avoid this by giving the user more explicit choice. However, some of the participants argued that the typical user would not know what to do with this choice. This answer triggered a range of discussion, as to how to provide guidance to the user in exercising these trust-related choices in an effective way. For example, different groups of users might settle on a common Naming Certification Service for their common tasks, based on a collective trust decision. See the discussion of choice in the final discussion section.

**Multiple GNRS:** There was some confusion in the discussion about the design of the GNRS. The MobilityFirst team said that there could be multiple such services, not just one, but this left people confused as to how a GUID updated its GUID-NA binding. Did it have to update it in all the GNRS services, or just one? If not all of them, then how would a source end-node or a router in the network know which GNRS to use to look up the current NA. The answer seemed to be that there was a baseline GNRS, but there could be other services that cached or replicated the information.

**Higher-level services in the routers:** According to the MobilityFirst white paper, their design includes the option of downloading third-party code into their routers—active networking. This was described as an option that had not yet been explored, but which was not precluded. Routers might run trusted code, with some sort of flag in the header to indicate that the packet should be deflected to that code. Services might include anonymous routing. However, it was not clear from the discussion whether caching (e.g. dealing with transient wireless disruptions) was a part of the core architecture or a trusted third-party service.

**Storage in the net:** There was some confusion about storage of content in the network. The architecture included a means to hold content during periods of wireless impairment. It seems that the expectation is that these events would be brief—short enough that end-to-end confirmation of delivery, challenge response protocols and the like are still viable tools. This reduces the need to trust the intermediate storage nodes, as one must if (as with email) there is no delivery confirmation. On the other hand, there were discussions of systems like CDNs and Torrents, and it was not clear if these were being contemplated as part of the same storage capability.

**Traffic analysis and revelation of intent:** As with all of these systems, the increased expressive power of the header, with the GUID as well as the NA in the packet, raised concerns about the revelation of rich transaction information. The fact that packets are reassembled into a larger data unit and (on occasion) stored in the network seemed to make the concerns much worse. The MobilityFirst team responded that GUIDs are opaque, and there is no easy way to tell what you are seeing. There was some dissatisfaction with this answer, since the observer may be tracking the flow of known pieces of content which have been previously identified and associated with their GUIDs. (Consider the story of the FBI tracking child pornography discussed elsewhere, where they find the content, and then passively observe everyone who retrieves it before acting. In this system, because they have found the content, they know its GUID. )

## Named Data Network (NDN)

The NDN architecture is distinctly different from the current approaches to addressing and forwarding. In NDN, there are two sorts of packets, *interests* and *content*. An *interest* packet is sent to request some named content, and the *content* packet returns that named content. In neither case is there a "host" address in the packet, only the name of the desired information.

An *interest* packet contains the name of the content being requested. A *content* packet contains the name of the content, the data itself, and a signature, which confirms the contents, as described below.

Names of content are hierarchical, and begin with the authoritative owner of the content, followed by the name of the specific content. Any owner/creator of content has a public/private key pair, and uses the private key to produce the signature. Thus, anyone with the public key of the owner can verify the *content* packet: in particular the integrity of the content and the binding to the name.

A distributed mechanism will allow nodes to build up a catalog of public keys for different owners, a *key certification graph.* In this way, routers can learn public keys, which will allow them to validate packets (as appropriate) as they forward them. (In particular, forged *content* packets that claim to match an *interest* can be detected in the net, not just at the final end-point.)

The name of data also describes its location. When information moves to a new location, there is a variation of a *content* packet called a *link* that encapsulates a content packet and is signed by the operator of the current location. This signature allows anyone with the public key of the current location to verify that the location named in the packet is the actual sender of this packet.

It is assumed that on top of this mechanism there will be a variety of search tools, content providers and the like that, among other things, provide for translation between other sorts of names and queries and the specific names used by NDN.

A key technical aspect of NDN is that when an *interest* packet is routed toward the location of the content, a copy of the interest is kept at each router. In NDN, there is thus per-packet state in each router along the path followed by the interest. The copy of the interest records the router port from which the interest came, as well as the name of the content being requested. The interest packet itself does not carry any "source address" from where the interest originated: this information is recorded in the per-packet state in all the routers along the path back to the original requestor.

When a router forwards a *content* packet, it has the option of keeping a cached copy for some period of time. This cached copy can be used to satisfy a request, rather than having to fetch the content from the original location.

**Security.** In this architecture, integrity and confidentiality of content can be assured by the use of encryption. The major differences that follow from this novel design relate to availability, DDoS, filtering, traffic analysis and privacy.

**Availability.** The ability to cache copies of popular content means that they can be retrieved even if the original location is down or the path is missing. In principle, anyone with a copy of the content can respond to an interest, so clever protocols for disseminating an interest can allow very robust retrieval. In particular, because the per-packet state in the routers eliminates the risk of a looping interest packet, these packets can be sent out using multiple paths

**DDoS.** Certain sorts of DDoS attacks cannot be carried out in this architecture. A node cannot be flooded with unwanted content packets, since one cannot address a content packet to a specific destination. The correct destination of a content packet is stored in the per-packet state laid down by an interest packet in the routers, so a content packet will only go where an interest packet has created the state. A node, however, can be flooded with interest packets. A server cannot be overloaded by repeated requests for specific content, since the content being requested will be cached in the network, so that most of the requests will never reach the server. Certain attacks will still reach the server, such as interests that request non-cacheable content or content that does not exist.

**Filtering.** Since interests and content packets do reveal the name of the content, it is easy for a router to drop packets referring to specific content. Since the name is not a hash but a somewhat meaningful name, this sort of filtering takes less effort than if it were necessary to first identify the content in some higher-level name space and then track down the matching content identifier.

**Traffic analysis and privacy.** In the discussion about NDN, there was a long consideration of these issues. Because the interest packets carry, in the clear, the name of the content (not as some sort of flat hash of a key but as a structure that indicated the source as well as the actual item), there is a lot revealed about what is being released. On the other hand, since the interest and content packets do not carry any identity of who requested the content, there is a different form of obscuration going on. If one can observe the very first router that gets the interest, one can see exactly who requested the content, but as the observer moves away from the requestor and toward the source, the interest (and its stored record) only indicate the direction from which the interest came. This different balance of features, with different implications for different sorts of monitoring, triggered a lot of discussion and seems to call for careful analysis of options for attack.

Because the interest packet does not carry the identity of the requestor, that information is hidden by default from the content provider. If the content provider requires that information, it will have to be transferred as part of a higher-level end-to-end protocol.

A user, to reduce the risks of traffic analysis and filtering, can encapsulate an interest inside another interest, which just describes a non-specific destination. Such an interest is similar to sending an interest to a service rather than to a content object. Obviously, such an interest cannot be cached, so this approach will require that the interest be sent all the way to the point where the encapsulation is undone. Alternatively, parts of the interest can be encrypted using the public key of a router along the path. Assuming that the routing protocol carries the packet to that router, the next segment of the interest can be revealed.

**Protecting the network infrastructure.** The NDN project includes a design for a secure OSPF, where routers are assigned keys in a hierarchy rooted in the domain, and sign routing messages.

**Security of *interest* packets.** For data retrieval, the interest packet contains little except the name of the desired data. However, there can be small amounts of additional information encoded in the name in an interest packet, potentially including authorization information or *control* information. In other words, interest packets can be used to convey control information. In these cases, the authenticity and authority of the interest packet needs to be verified, and the contents protected from change. A key management scheme and the ability to sign an interest packet are tools to make this secure.

## Comparisons

### Similarities

In general, all these schemes share a feature that distinguishes them from the current Internet: a two-step rather than one-step name-to-address resolution scheme. In the Internet, a high level name (e.g. a URL) is mapped to an IP address by the DNS. This happens in one step. The IP address is being used as both an identifier for the end-point and its location. All of these schemes have a separate identity and location schemes, and separate mechanisms for mapping from name to identity and from identity to location, except for NDN, which has effectively eliminated any concept of a location. Most of the schemes have a way to assign an identity to things other than physical end-nodes, including services and content.

In contrast to the current Internet, which uses IP addresses as a weak form of identity for end-nodes, all of these schemes implement the idea that the identifier of an end-point entity, whether a host, a service, a piece of content or the like should be "self-authenticating". Mechanically, this is done by making the identifier a hash of a public key (or some other feature of the entity). Assuming that the entity holds the private key, a challenge-response exchange can confirm to each end that the other end is as expected. This check prevents many sorts of attacks in the network, including DNS poisoning, packet mis-direction, and so on, from being successful.

However, detecting a failure or an attack is not enough to assure successful operation—all it can do is give a clean signal of failure. To provide successful operation in the face of these sorts of attacks, two more functions are required: first a means to detect where in the network the failure or attack is happening, and a means to avoid or "route around" this region. Many of these schemes contain mechanisms to try to isolate the region of a failure, and many of them give the end-point control over the route selection to some degree. This choice reflects a preference for a basic design principle of the current Internet: since the network is not aware of what applications are trying to do, the network cannot detect when a failure has occurred. Only the end-points of the communication, which are aware of the desired application semantics, can detect problems and attempt to restore service.

With respect to routing, all these schemes take the view that the network is build of regions (like the autonomous systems or ASs of today) that are separately managed and deployed. The architecture gives the end-points control over routing at the level of picking the series of ASs to be used, but give each AS control over its internal routing.  The control thus given to the end-node matches the granularity of the business relationships among operators.

While XIA seems to have a richer set of IDs (HID, SID, CID) compared to Mobility first with its GUID, the MobilityFirst GUID includes a component called the SID, which effectively qualifies the GUD with respect to how it is treated inside the network. It specifies different treatment, just as the different XIDs do. When the GUID is looked up, the GRNS must return the sort of information needed for the type of SID, for example an list of anycast addresses for the anycast SID. In NDN, the type of forwarding that applies to interests (e.g. possible flooding or multicast) is not specified by the name of the information being sought, but by the configuration of the router. In Nebula, different services can be invoked by the *label* associated with the POC at each stage of the forwarding.

## Important differences

NDN used structured names for information, which function as routable addresses. XIA puts several sorts of IDs in the header, including a flat ID of the service or content, and as a fallback, a network address at the level of a domain. It is assumed that the flat ID is routable inside the domain. MobilityFirst assumes that there is a global distributed table that can map from a flat ID to a network address NA that might have the granularity of a domain (AS) today. Like XIA, it puts both the flat ID and the domain-level routable NA in the header, but in a less flexible format than XIA, which can allow several options in the header to drive forwarding. Nebula uses a rich form of source address that specifies the route at the level of an AS.

NDN is distinctive, compared to the other three proposals, in that it puts per-packet state in the router, and explores what can be done in exchange. In particular, packets carry no source address—they lay down "bread crumbs" in the router as they flow toward the information, and the returning information cleans up the crumbs. So looking at a packet reveals what is being sought, but not who asked for it.

## Values in Design Council presentations/discussion

As part of the FIA process, NSF has funded Helen Nissenbaum from NYU to assemble a group of experts from the fields of social science and law, to participate in these group discussions and give reactions and insights. Five members of that group were at this meeting, and discussed a variety of topics.

### The concept of values in design.

In their introductory comments, they offered three options for thinking about the values in design movement. The first option, which is the basic message, is to be sensitive to the value implications of your designs. The second option is to assume you are being told to make the right choices. This is not actually the message, since your training in computer science does not especially qualify you to make these decisions. Ideally, they would be made by a larger cross-section of stakeholders, and what would be in place as part of the design process is a coupled "values design" process. The third option is to make sure there is enough flexibility in the system so that good options are not precluded.

Some of the FIA designers responded by saying that their design preference would indeed be to "add lots of knobs" to the system so that the values and policy could be adjusted later, but to that point there was also pushback from the VID group: there is no such thing as technology so flexible that it is value neutral at the core. Being sensitive to the residual embedded values is a critical part of the message. And if systems have "knobs", system designers should attempt to articulate the range of values that the system can capture.

Values in design does not mean one must achieve perfection. That criterion would paralyze any designer. We must be prepared to soften what we try to do so we have room to act.

One of the consequences of being sensitive to values implications is that it helps to discover tensions among values. Values do not usually exist in isolation; they are part of a context with contention among values. The idea above of a "values design" process that involved different stakeholders is to identify and air these contentions. However, there is an important distinction between values that are sorted out at design time (in a values design process) and those that are sorted out at "run time" by fighting over the knobs, and "messing with the architecture", examples of which might include firewalls in the current Internet.

Part of what must develop as part of the "values design" process is a common framework to discuss the points of contention, a sort of pidgin English that hooks the parties together. And the points of contention must also be expressed in the features of the system, as well as the language of the stakeholders. It is important to link the way values are expressed in words and expressed in the system.

### Choice and defaults

Another point of view is that if there are choices, especially at run time, it is important to ask who has control over those choices. One cannot leave these choices to the ultimate end-user: that person is not a skilled professional, and cannot be expected to set a large number of configuration parameters to “dial in” the values of choice. As well, there is a tremendous imbalance of power between individual users and the large actors that will control the platform and the applications. One way to deal with this is to try to set the choices to a default that respects the anticipated value preferences of the end-users rather than the more powerful actors, to“tilt the playing field” in favor of the users, but this point of view begs the question of how the architecture could possibly control the default setting of parameters that are intended to be adjusted at “run time”. But “opt-out” approaches that require explicit action by the end-users seem to lead to outcomes that do not align with preferences.

(An afterthought—perhaps the identification of explicit control points helps bring to the attention of regulators where they have the option of intervening, a fact that even they may not be easily able to find. The role of the regulator cannot be ignored here, since it was asserted that with respect to privacy, what we have today is a clear market failure.)

Since people cannot be expected to find every “value setting” in a system, or understand exactly what it does, it may be a defensible approach with respect to values design just to pick setting in cases that do not seem as critical to the outcome. Fixing them in the design, even if imperfectly, may be better than allowing actors with power to mess with them. Leaving all the tussle to run-time does not respect the actual power balance.

Because several of these systems have more scope for choice, as opposed to fixing one mode of operation, powerful actors may be able to seize control of these choices to shape the network to their interests.  The VID speakers pointed out that choice may not be the friend of the user, but instead the way that different powerful actors (e.g. the U.S. or China) can shape the Internet that their citizens have.

Because there is a real imbalance of power between the end-users and the range of powerful actors that design and operate the Internet and its applications and services, it was suggested that a design might have the goal of making some sorts of actions by the more powerful actors possible but hard. Examples included lawful intercept and traffic analysis. It was suggested that traffic analysis should be made “hard but possible” by design. Such a design might not depend on technical means to increase the work factor, but instead on the need for two or more actors to be required to cooperate in order to carry out the interception. Designs that require collaboration amongst two sorts of actors can protect against a change of heart in one.

## Security as a privileged value

Speakers pointed out that the particular topic of security seemed to be a privileged value, not just at this meeting where it was the assigned topic of discussion, but elsewhere as well. IETF RFCs must have a mandatory section that discusses security implications. Are there other values that should be similarly privileged, such as privacy?

In fact, a counter position was offered that at the level of architecture, security was not a value. Security is a property that emerges through the combination of mechanism, operational decisions, and the like. Few of the presentations positioned their mechanisms in a larger context of operational and deployment options.

## Defining security

At the same time that security seems to be a privileged value, it often went undefined in this meeting. Security was described as much by mechanism as by requirement, which necessitated the back-deriving of the intended value from the design spec. In the talks, there was a lot of discussion of self-signing identifiers, but much less discussion of the various implications of such a design, a fact that triggered a lot of questions from the observers at the meeting. An example of this was that it seemed that control of mis-direction (e.g. control of phishing) was a much higher priority than regulating traffic analysis. The SCION scheme seemed to have lots of potential value issues inside that had not been brought out in the discussion of mechanism.

## Case studies

Several speakers pointed out that discussion of values in the abstract is sometimes not effective; it can be too vague to illustrate the real tradeoffs. Several examples were offered in the context of this discussion.

- Law enforcement agencies regularly use traffic analysis (e.g. tracking of source and destination addresses) as a tool of investigation. A scheme that makes this harder (e.g. NDN) may trigger resistance from the government. Whether this sort of resistance would be effective was debated, but the point is clear—most value assertions, once they are made concrete, are contested. A specific approach here is that perhaps a system could be designed so that traffic analysis was possible but hard. A challenge to the research (security) community was whether it was possible to design mechanisms that made things not impossible but "robustly hard".
- TOR is currently an overlay that could not possibly scale to protect all the traffic in the Internet. One point of view was that if a service like this were around, it should be available to anyone—protection of this sort should not just be available to those who can pay. But making this universal might trigger the pushback just mentioned. So perhaps every project should have a graduate student look at what a TOR-like mechanism would look like in their architecture.
- Caching was a central idea in several of these schemes. Are there value issues in the decisions as to whether to cache, and how and when to clear the cache of older content?
- Since a future Internet should aspire to be a global infrastructure, how do we deal with variation in global values. Some values we hold may be somewhat

U.S.-centric, but human-rights norms should be universal, not seen as U.S. On the other hand, a perspective of cultural relativism suggests there are cases where we should be justified in designing systems that conform to norms different from ours. This aspect of the design space is tricky, especially when we take into account the tremendous imbalance of power among the different actors. Where and how could a conversation about these issues take place?

### Dynamics and evolution

One criticism of the designs (and their presentations) was that they seemed to define security as a static, design-time property, while real security management is an ongoing evolving process. Deirdre Mulligan mentioned her recent work with Fred Schneider that takes the point of view that one should talk about "managing insecurity", by analogy to the way we manage illness through the public health system. It was hard to jump from the mechanisms presented here, which seem to stress end-to-end validation of identity, with all the things one might want to do to manage insecurity, which might include such things as isolation of infected regions or implementation diversity and herd immunity.

Systems like this, if successful, might last for 50 years or more. What, over that time, will need to change? Different parts of the system may need to change with different time-constants. Are the rates of change among different parts lock-stepped? Can you imagine various sorts of catastrophes that might trigger (or demand) sudden change?

Norms and values can also change fast. Not that long ago, there were protests in Australia about a national identity card, but recently they were deployed without protest.

It may be helpful to have some future-facing case studies—the "science fiction" of the future Internet's success, as a lens to look at the design decisions. Find some that build in your values in the best way, and some that perhaps do not.

## Summary security comments

### Requirements and mechanism

All of the talks seemed heavy on mechanism and light on discussion of requirements and consequences. It was necessary to teach the mechanisms, since otherwise the listeners would have no idea what was going on, but it would be helpful to work out stories (perhaps "success scenarios") of how the mechanisms provide

### Red teams and the embedding of attackers

In several contexts, the security observers urged that the projects embed a "friendly attacker" in their design teams. As one person put it, a graduate student who has just designed a neat new mechanism is not the best person to find its flaws. Challenge yourselves to identify your soft spots and deep assumptions. The parts you like the most are likely to be the most troublesome, because you may not have scrutinized them because you like them.  NSF expressed a willingness to provide supplemental funds for a well-formulated plan of red-teaming and outside vulnerability assessment.

**Quote of the day:** "If you don't embed an adversary in the design of the critical mechanisms [key management and trust management], your system will have as much chance as a paper cat chasing an asbestos rat through hell." (Earl Boebert)

## Public keys

All these schemes use public-private key pairs to allow validation of transactions. NDN uses keys belonging to the content provider to sign the packet, so that anyone with the public key of the provider can confirm that the packet is not a forgery. Nebula uses keys belonging to each actor that has a policy with respect to the route to sign the Proof of Consent for the route.

Because of this emphasis on pubic keys and PKI, there was a lot of discussion about the problems of key management. In general, the security observers thought that key management would be a major source of vulnerability. A system penetration could lead to the theft of the associated private key, and the need to replicate that key at all replicas of the associated service or content increased the potential of the vulnerability. Several of the discussions stressed the issue of revocation of keys, and the problems of key continuity. Some of the architects stressed that the design of the key management system was an end-node issue, and thus outside the bounds of the architecture. This point of view actually seems to increase the vulnerability of the system, since the design of this part of the system is being left to people who may have less experience and skills than the system architects.

### Trust management and roots of trust

Essentially all these schemes try to avoid the problems that have arisen from the hierarchical nature of DNS, and its dominant role as a naming service, by allowing multiple naming services to co-exist. The goal is to avoid a single "root of trust". However, this approach raises its own set of issues, which in general have not yet been resolved.  Again, for all of these systems the observers questioned this approach, for two reasons.

The first was a doubt that there can actually be a "root-less" system. If there are multiple disjoint naming systems, then there has to be some convention that guides the potential user of a name to the correct service to look up its binding. One could put the name of the name service as a prefix to the name, but this just re-created the rook of the tree. One could try to avoid having actual "root servers", by distributing the information out of band, but one must still begin by finding the machine that hosts the relevant name service. In what name space is it named?

The second was a doubt that the users could actually cope with this range of choice, especially if the choices had to do with trust-related decisions. It seemed to many observers that the system and its users would settle on a default (just as DNS became a default—DNS is not mandated in the architecture).

In general, there were a lot of concerns about which elements in the network must be trusted, and to what extent. Some of the teams suggested that a user would naturally trust their access provider, but others asserted that based on what is happening today, access providers are not trustworthy at all, and should be limited to the most basic of actual packet forwarding. Other people suggested that people would trust their end-device, but again, given the user monitoring that has gone on with devices today (e.g. location tracking) this assumption was challenged as well.

A general suggestion from the security observers was that each project should do a complete audit of their "trust design". They should consider:

- The extent to which each element and class of element needs to be trusted.
- How that need for trust could be minimized.
- What happens if that trust is violated.
- How a violation of trust could be detected?
- Architectural methods to allow cross-checking and validation of trust assumptions.

One of the observers used this illustration. When you design a system that incorporates many actors, you should think through what happens if one of the actors changes his role or allegiance? Viktor Mayer-Schoenberger, in his book "Delete: The Virtue of Forgetting in the Digital Age" talks about the difference between the fate of the Jews in Holland and France. The French, with a cultural tradition of not fully trusting their government, kept many personal facts to themselves. The Dutch, as good citizens, were more forthcoming about things like census data. When the Nazis took over Holland, many more Jews were rounded up there than in France. The government turned from friend to adversary, and system designers should contemplate this sort of outcome with any powerful actor in the system.

## The definition of "user"

In many of the conversations about different systems, there was a degree of confusion between "user of the architecture", who is an application designer, and the human user at the end point of the operating system. When trust decisions and choices were "left to the user", it was not clear if this was the application designer, the actual human (e.g. in the example of users with different levels of concern about privacy), or some service that a user decided to trust (and delegate decision-making to) at run-time. The observers urged that the designers pay careful attention to this distinction, and be clear what they were thinking about when they said "user".

Beyond this point, there were several other comments about users.

- When you think about the human user, do not think of one user in isolation. Users collaborate to carry out a task, and they collaborate to sort out choices related to trust and other social aspects of the system. The system should contemplate the ides of the user in the aggregate, not in isolation. And do not assume that the community of users is homogeneous. Designers must not lose track of the diversity of real users.

- While there was a strong encouragement to distinguish the application and its designer from the human user, the same tools of analysis might be applicable to both. Asking the same questions of both, using the same sorts of analytical reasoning, tools and methods to study them, may open up some questions not otherwise considered.

## Privacy

Because several of these schemes imply greater revelation in the network of what is being done, there was a lot of discussion about privacy, and (at the packet level) traffic analysis or transactional data (who talks to whom).

- Those who have studied privacy believe that there is an intractable market failure around privacy. If one is concerned about privacy, one must bake in defaults that favor the user over the more powerful actors.
- The question of whether users actually care about privacy is a long-standing one, with answers that seem to vary over time, from culture to culture, and depending on how the question is asked.
- Leaving choice to the market to lead to poor choices. Market pressure could not give us safe food or safe medications. Why should market choice give us privacy, which is a pretty subtle item to specify or understand?
- One way to understand the design and selection in a space of choices is to model it as a multi-player, long-term, multi-round game. Ask what the incentive structure is, and how the actors fit into the game.
- As a value assertion, privacy should not be a luxury of the rich. It should be an essential service.  There is perhaps a tension between privacy as something you can get from providers that care, if you can obtain service from them, and a embedded preference for privacy in the architecture.

The work of Alessandro Acquisti at CMU was cited as a good source of information in this area.

With respect to the power of traffic analysis to learn what is going on without looking in the packets, the work of Vitaly Shmatikov (Texas) and Arvin Narayanan (Stanford) was recommended. They look at de-anonymizing of networks.  It was observed that one of the reasons that NSA went along with relaxed concerns over strong encryption is they found traffic analysis (transactional information) so rich.

Given the reality that powerful actors will have incentives to manipulate the system (e.g. to observe traffic, whether for law enforcement or behavioral profiling to support ad placement) we should assume that the network will exist in a legal framework. Rather than trying to solve the whole problem by technical means, perhaps we should co-design a model legal regime to describe the bounds of acceptable behavior, and demand of the technical design that it make it easier to separate and distinguish the two.

### Attacks on the network

There was relatively less attention paid in the various talk to protection of the network itself from attack, and more paid to protecting a communication from attack. This may be justified, since much is known today (if not deployed) about how to secure routing protocols from attack by malicious actors. However, these networks are more complex, with more components and classes of actors. It was suggested that each project should review the value of the various components of their system as attack targets.

## Back to the questions

While the project presentations did not directly answer the security questions that were included at the beginning of this report, many of the answers did emerge in the discussions:

1) *How does your design deal with attacks directed at the network itself?*
   There was partial attention to this topic, but perhaps less than some of the observers might have expected. All the systems assume that their network is build of regions that are not of equal trust. Rogue regions can exist, and must be quarantined. The discussion of secure routing protocols was mixed, with some suggesting that techniques derived from a variant of SBGP were a known approach, and some suggesting that the implied key management was a serious operational/policy problem. SCION (XIA) has regions of trust to try to isolate larger groups of regions for routing. NDN has a version of OSPF with signed routing information. In Nebula, routing and inter-region interaction are handled in the policy plane, NVENT, which is not yet fleshed out. There were concerns about the security implications of this complex layer. In MobilityFirst, there were concerns about the security of the highly distributed GNRS. There was not a lot of detail about global routing in MobilityFirst.
   There was no discussion of the security of management protocols, or other mechanisms that link regions.
   There was limited discussion of routing diversity as a means to bypass failed components, which could include physical failures and malicious failures, but there was only superficial discussion about techniques to isolate such failures.

   1a) *To what extent does your design emphasize resilience/availability of the network in the face of attack, as opposed to resistance to attack?*
   With respect to attacks on the routing protocols, the emphasis seemed to be on detection and isolation.

2) *How does your design deal with attacks on communication among end-points?*
   This question was central to most of the discussions. Integrity and confidentiality were in general assumed to be an end-to-end solution, but self-

certifying IDs were used to detect redirection attacks and other mis-directions. End nodes could verify the identity of each other, assuming that the higher-level binding of user-meaningful name to ID was not corrupted. In some of the schemes, propagation of public keys into the network would allow routers, and not just end-nodes, to detect forged packets.
There was a lot of attention to the issues of trust, and the need for the system to provide trustworthy versions of critical services such as name resolution.
The topic of traffic analysis received considerable discussion, as noted elsewhere in the report.

3) *How does your design attempt to mitigate attacks on end-nodes?*
This topic received limited consideration. NDN has the most distinctive character here, since the lack of host addresses preclude a large number of attacks that depend on addressing packets to the target. Attacks on end-point vulnerabilities are harder to contemplate in NDN, but a fresh analysis of attack patterns would be needed for that design because of its novelty. Nebula is "deny by default", so probes of machines are less likely. But in both cases, delivery of mal-ware is possible. XIA provides a more traditional source-destination delivery pattern, as does MobilityFirst.

3a) *Does your design include methods to deal with multi-stage attacks (using intermediate nodes previously corrupted)?*
There was no discussion of this issue.

3b) *Does your design attempt to mitigate the consequences of end-node faults and vulnerabilities?*
There was no discussion of this issue, aside from discussion of DDoS attacks.

4) *Does your design deal with information assurance?*
Content identifiers that are a hash of the contents provide a means for a receiver of information to verify its source and integrity. Higher-level issues (e.g. search or name resolution) were described as outside the scope of these proposals.

5) *How does your design deal with DDoS attacks?*
NDN, because it does not have host addresses, does not support a range of DDoS attacks, but residual options exist, as discussed in that part of the report. The signed locator in MobilityFirst that binds the GUID to the NA prevents attacks based on a falsified source address.

6) *Does your design address the sorts of higher-level considerations listed below?*
6a) *National security: lawful intercept, traffic analysis.*
This topic received a great deal of discussion, as described throughout the report. NDN makes one aspect of traffic analysis harder by removing host addresses. Most of the proposals, with the increased information in the headers, seem to increase the potential of traffic analysis. Systems such as Nebula that give the end-node some control over routing might be able to avoid regions

where these sorts of attacks were anticipated. The discussion from the floor suggested a design goal of making lawful intercept and traffic analysis possible but hard: none of the schemes discussed mechanism of that sophistication.
6b) *Privacy; anonymous action.*
Few of the systems had any tools to enhance privacy beyond end-to-end encryption of content.
6c) *Control of delivery of illegal or forbidden content.*
There was no discussion of this issue.
6d) *Accountability; deterrence.*
There was no discussion of this issue.
6e) *DRM and content management.*
There was no discussion of this issue.

7) *What new sorts of vulnerabilities does your approach create? What are the "new" security problems you will have to resolve? Can you identify points of (security) failure with severe implications?*
While there were specific points raised in the various presentations, none of the projects had a methodical answer to these questions, and the observers stressed the need to carry out this sort of assessment.

8) *Has your group agreed on a threat model?*
The groups did not give concise answers to this question, and again the observers stressed the need to do so. It was important to describe the societal context in which these systems were expected to function, and understand the level of threat they were expected to deal with.

## Reading list

Earl Boebert suggested the following books as good reading on aspects of thinking about security.

Kathryn Schultz , *Being Wrong: Adventures in the Margin of Error*. The science of "wrongology": why we ignore evidence and hang on to our errors. Or see: http://www.ted.com/talks/kathryn_schulz_on_being_wrong.html

Sam Adams, *War of Numbers: An Intelligence Memoir*. A revealing look at intelligence analysis, illustrating what can be learned from traffic analysis.

Sissela Bok, *Secrets: On the Ethics of Concealment and Revelation*.

## A final word

After the meeting, Earl Boebert, one of the outside observers, sent this summary of his thoughts. Rather than integrate this into the text, I am including it as he wrote it:

"Here's a consolidated list of my concerns, FWIW. They address the issue I called "deployability" at the workshop, that is, absence of showstoppers. I think this criterion applies to all projects, whether they are intended to deploy, prove a principle, or provide a point solution to a specific problem. The two showstoppers I have most commonly encountered are the Magic Crypto Fallacy and the Benign Environment Fallacy.

"The Magic Crypto Fallacy results when the system analysis and design begins with the assumption that all crypto boxes on the net are properly keyed. This dodges the essential and very hard design problem of establishing a chain of authentication from some human authority to a dumb robot with crypto in it. Any deployable system must address the emergency rekey problem, which, along with revocation, are the Achilles' heels of public key crypto. If I lose my private key **while the system is operating and under attack** then some really bad things happen. Issuing a new public/private pair is impractical. If I try, I can no longer see any encrypted secrets that I have cached or that may come my way between the time of loss and the time the **whole net** has been rekeyed with the new public part. I likewise cannot authenticate any messages in that interval. To rekey I have to reconstruct the chain of authentication completely ("Hi. This is President Obama. Bo ate my smartphone. Here's a new public key to use when when we talk. Have a nice day, and be sure to vote Democratic in 2012." Looks silly to you, but you're not a robot.) This form of emergency rekey is a subset of what in the old days was called OTAR, or Over The Air Rekey. It is analogous to the in-band signaling the Bell System tried in the 70's and we all know where that got them :-)

"So you have to escrow the private keys, which in the civilian sector opens up a whole ugly set of issues pertaining to who keeps the escrowed keys and under what circumstances can the Yoo Ess Gummint get their grasping hands on my private bits. You also have to figure out a highly reliable storage system, because a dropped bit in a movie is glitch but a dropped bit in a crypto key is a catastrophe.

"I should also note that crypto is a DoS attacker's dream. If the crypto system is not properly designed, I can interrupt/step on traffic in such a way that your crypto robot is so busy throwing away packets and resynching that it goes into the moral equivalent of a catatonic state.

"The Benign Environment Fallacy is the assumption that nothing inside the security perimeter (assuming that one can even be drawn in a network of mobile devices) has been compromised. Fuggedaboutit. Supply chain attacks are trivial. Mobile devices are lost and stolen. The mobile devices we are talking about are going to be made to commercial standards, which means they'll cough up their stored secrets readily. A deployable system must be able to detect and remove a compromised node. Which means the crypto design must handle revocation. The only way this can be done in the real world (compromised key lists being something less than a joke) is to rekey the whole net while leaving the compromised sites off the call list. Not pretty. Similarly, the designers need to consider the problem of humans with admin privilege going south on you. Even less pretty.

And that's all I have to say about **that** :-)

Cheers,  Earl”