1 Hello, David

My goal for this document is for it to contain most of our thoughts and sort of iteratively converge to a good final document.

2 Levels of Anonymity

There are three main properties that a anonymization system might want to satisfy.

NOTES: Lets cross-check this against the literature.

2.1 Receiver Anonymity

The server cannot determine the identity of the client.

2.2 Sender Anonymity

Clients cannot determine who is providing they content they download.

2.3 Sender-Receiver Unlinkability

An adversary with some level of knowledge of the whole system cannot determine who is communicating with whom.

2.4 Data Privacy

Any proxies invovled have no way of figuring out the data. This will of course require that the client get some sort of public key for the server.

2.5 Server Unawares

The server is unaware that it is being commuicated with via a proxy.

3 Scenarios

In this section, we present scenarios that would warrant the properties above.

3.1 End-to-End Connection

Two people know each other's addresses and need to stay unlinkable.

3.2 Unsecure chat room.

You want to connect to a Free Opressistan Forum. You probably don't want your password visible, and you don't want to be linkable to the site either.

4 General Approaches

4.1 Proxy

Simply route your traffic through a proxy.

4.2 Onion Routing and MIXes

We can do some things to make the proxy approach better.

4.3 Broadcast

Just send your packet to everyone and no one will know who the intended recipient was.

5 XIA

Some features of XIA have implications when it comes to anonymity. We briefly describe the key ideas here and discuss their impact on anonymous communication later.

5.1 Principal-Based Communication

In contrast with today's host-based Internet, XIA provides a framework for communication among *principals*. The idea of principal-based communication is that users should be able to address packets directly to their final *intent*. For example, a user searching for books on Amazon wishes to communicate with `www.amazon.com`; he doesn't care which particular Amazon server responds to his request.

In this example, `www.amazon.com` can be viewed as a *service* principal. In the case where communication with a particular machine truly is the intent, traditional host-based communication can still be achieved via the *host* principal. Other principals include static *content* (e.g. images) and *autonomous domains* (similar to today's autonomous systems).

5.2 Intrinsic Security

All addresses (or, better put, identifiers) in XIA are *intrinsically secure*; exactly what this means varies among principals. For example, a content identifier (CID) is the cryptographic hash of the content itself, enabling anyone receiving the content to verify its integrity. Hosts and services are required to have a public/private key pair; therefore their corresponding identifiers (HIDs and SIDs) are simply the hashes of public keys. A host can sign any communication it generates with its private key and anyone can publicly verify the signature using the host's ID.

The use of identifiers which are cryptographically derived from a user's public key naturally has implications concerning anonymity (or lack thereof) which we discuss further in §8.

6 Adapting Existing Methods for XIA

In this section we discuss how existing approaches work in XIA.

6.1 AD-Provided Proxy

AD routes based on SID, which we trust them not to link to us.

6.2 Bound-Route Proxy Service

Your requests all take the same route.

6.3 Altruistic Onion Routing

How should Tor work in an XIA setting?

7 New Approaches Made Possible by XIA

7.1 Multi-Route Proxy Service

In §5 we introduced the service principal. We gave the example of a user communicating with `www.amazon.com`; in this case, his true intent is to send requests to the Amazon service, not to a particular Amazon server. He addresses his requests to `www.amazon.com`'s SID and lets the network pick a particular server.

Now consider a similar situation in which a user routes to-be-anonymized packets through a proxy *service* rather than a particular proxy server. Each packet is addressed to SID_{proxy} and is routed by the network to various hosts providing the service (Figure 1). Even if only one proxy server is used per packet, simply by using the the service principle our anonymous user has made it harder for network traffic analysis to discover his use of a proxy.

7.2 The Content Principal

Something about publishing anonymously?

8 Challenges Posed by XIA

While XIA provides new mechanisms we can harness for anonymization, some features of the new architecture also introduce challenges to providing anonymity.

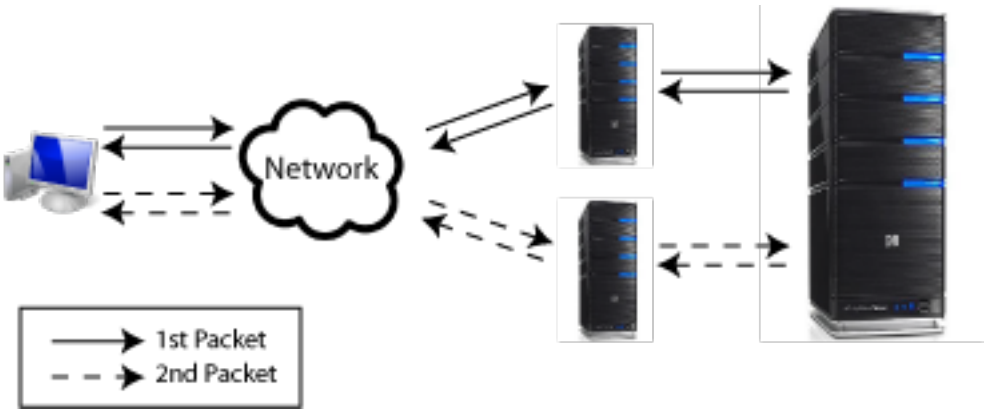

Figure 1: The client routes all requests to SID_{proxy}. The network selects a server implementing SID_{proxy} for each packet.

8.1 XIDs are inherently non-anonymous

8.2 The trouble with SIDs

In this section we talk about the "I'll only encrypt with the addressee public key" pitfall.

9 PDRSA

Fuck this section.

9.1 Algorithm

Here we present the algorithm for Piecewise-Decrypt RSA. The main difference now is that there are now two private keys, one of which is kept and the other of which is passed to the proxy. I forgot why this distinction is important. I may have wasted a few hours on this.

9.1.1 Key generation

Let p, q be large primes like in RSA. Define $n = pq$. Note that $\phi(n) = (p-1)(q-1)$. Now select e, d_2 that are coprime with $\phi(n)$. This implies that ed_2 is also coprime with $\phi(n)$. While e can be selected to minimize the cost of encryption, d_2 must be selected randomly. Let $d_2 = (d_1 e)^{-1} \mod (p-1)(q-1)$.

Publish (e, n) as the public key. (d_2, n) is the semiprivate key. (d_1, n) is the private key.

9.1.2 Encryption

$c = m^e \mod n$

9.1.3 First Pass Decryption

$h = c^{d_2} \mod n$

9.1.4 Final Decryption

$m = h^{d_1} \mod n$

9.2 Proof of Correctness

Pretty similar to the one on wikipedia for RSA. Fermat's is easier to understand, but Euler's is shorter.