

# Shrinking the Leap of Faith

Jacob Beal and Tim Shepard

March 30, 2005  
**UNPUBLISHED DRAFT**

## Abstract

The Internet is filled with strangers. We observe that humans cope with a world full of strangers by managing trust relationships with their acquaintances. On the Internet, too many strangers are capable of trying to make a computer's acquaintance. Direct encounters between computers, however, are sparse, as are face-to-face encounters between humans. We sketch an approach to extending trust based on direct encounters, in which buddy relationships formed during direct encounters are used to vet strangers.

## 1 The Problem of Strangers

The world of computer networks, like much of daily life, is filled with strangers, some of whom are trustworthy and some of whom are dangerous, whether through malice or incompetence. Interactions with strangers are guided by a tension between the competing concerns of safety and openness: too much openness allows dangerous strangers to victimize you, too much safety prevents beneficial interactions with trustworthy strangers.

Currently, there are two main models for trust used to try to separate trustworthy and dangerous strangers: Public Key Infrastructure (PKI) and the leap of faith. Under PKI, trust is distributed from a small number of commonly held certificate authorities, and a computer can verify itself using a certificate cryptographically signed by a certificate authority. Although elegant, the centralization of the PKI model makes it highly vulnerable to problems with certificate authorities and it has not been widely adopted except in the case of financial transactions (e.g. most HTTPS usage).

Under the leap of faith model, typified by `ssh`, there are no central authorities. Instead, each participant keeps its own list of trusted keys. The first time that a computer is encountered, it presents its

public key, and the recipient must make a leap of faith that the key represents the computer it was trying to connect with. Once the leap of faith has been made, it is possible to check that the same computer is being encountered, but there is no guarantee that it isn't a consistent imposter.

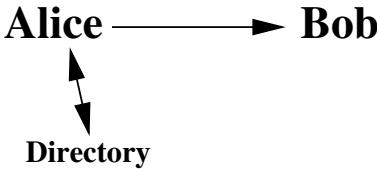
Humans have been dealing with the problem of strangers for millennia, and the way that human social networks create and maintain trust relationship looks more like leaps of faith than PKI. Our work is informed and inspired by human social networks. We do not, however, attempt to imitate them closely because there are significant differences between computers and humans.

We will outline a new approach to the problem of strangers, inspired by human behavior. First, we observe a common problem with trust models in which the computer and the user are confused together, leading to flawed software. Separating the computer and the user lets us see that current software does not take advantage of the ability of computers to meet one another directly. We then sketch one way that a computer initiating a transaction might shrink its leap of faith, given a collection of random buddies gathered from other computers it has encountered directly. Finally, we suggest ways that the random buddy idea might be applied in reverse to shrink the leap of faith for a computer responding to a transaction request.

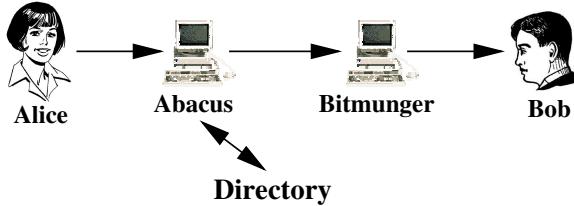
## 2 Separating Computers and Humans in Trust Models

Computer security thinking often elides together people and their computers. This could easily be a bad habit inherited from dealing with cryptography, in which it is perfectly reasonable to write paragraphs like this (Figure 1(a)):

When Alice wishes to send a secret message to



(a) Common Cryptographic Model



(b) Average Human Experience

Figure 1: When Alice sends a secret message to Bob, who does she need to trust to keep it secret? As commonly described in cryptography, Alice needs only to trust the directory and Bob, but must do quite a bit of math. If Alice is an average human, however, she must also trust her computer Abacus and Bob's computer Bitmunger to keep the secret, and trust Bob's computer to identify him correctly.

Bob, she looks up Bob's public key in a directory, uses it to encrypt the message and sends it off. Bob then uses his private key to decrypt the message and read it. No one listening in can decrypt the message. Anyone can send an encrypted message to Bob, but only Bob can read it (because only Bob knows Bob's private key). [1]

Alice and Bob might be people, computers, or anything at all capable of performing the appropriate calculations with large numbers. Most people who use computers, however, are unwilling or unable to do the math involved. The experience of an average human with cryptography, therefore, goes something more like this (Figure 1(b)):

When Alice wishes to send a secret message to Bob, she types it on her computer, Abacus, and asks Abacus to send it to Bob. Abacus looks up a public key for Bob in a directory, then uses it to encrypt the message and send it to Bob's computer, Bitmunger. Bitmunger receives the



(a) First, dire warnings!



(b) Internet Explorer passes the buck

Figure 2: The typical user experience of cryptography in a web browser (in this case Internet Explorer[3]). When connecting to a secure site not verified by its built-in certificate authorities, the browser gives up and hands over its trust problems to the user.

message and then uses Bob's private key to decrypt it for Bob to read.

In order to understand the trust dependencies in this situation, we need to explicitly include both computers and humans in the model. If Alice and Bob perform the algorithms themselves, she can trust that her message is secret if Bob keeps it secret and if the directory is accurate.<sup>1</sup> As an ordinary user, however, she must also trust that Abacus does not reveal the message to anyone and uses a trustworthy directory, and that Bitmunger can correctly identify Bob and chooses to reveal the message only to him.

We believe that effective computer security engineering requires analyzing threats with this more complete model, and that failure to do so leads to

<sup>1</sup>If the directory is Alice's own personal address book, she can be quite certain of its integrity.

designs which create insecurity in the interface between the human and the computer by not taking their trust relationship seriously.

Current web browsers like Internet Explorer or Mozilla Netscape are excellent examples of this failing. When making a secure connection, the browser hides all cryptographic details from the user as long as it can verify the server using its set of built-in trusted authorities. If the server doesn't use one of the built-in authorities, however, the browser simply punts the problem to the user, who must decide whether to make a massive leap of faith based only on a brief hexadecimal garble (Figure 2). Internet Explorer sums up the problem elegantly as it dumps the problem in the user's lap:<sup>2</sup>

... a single bad Certificate Authority completely negates the effectiveness of the entire system of Internet security.[3]

Being told that the fate of the entire Internet rests upon their belief that a particular wad of uninterpretable characters is correct, is it any wonder that many people choose either to completely ignore the warnings or to avoid Internet transactions altogether? We need to consider the security situation from the perspective of Abacus and ask how it can manage to recognize Bitmunger without the help of a certificate authority.

### 3 How Computers Meet

Let us define two computers to have met if they can reliably recognize communications from one another in the future. Public key cryptography is one way to accomplish this: if Abacus and Bitmunger give each other their public keys, then Bitmunger can use its private key to sign a message so that Abacus can verify it's really from Bitmunger.<sup>3</sup> Meeting is not the same as knowing identity; Abacus cannot generally verify what computer it is talking to, only whether it's the same computer that gave it a particular public key.

There are only three ways that Abacus can meet Bitmunger (Figure 3):

- **Direct Encounter:** Abacus and Bitmunger can meet directly when they are on the same communications medium, such that they can communicate directly, without any intermediaries. For

example, they might be on the same ethernet wire, or communicate wirelessly via infrared or radio.

- **Introduction by a 3rd Party:** Abacus can meet Bitmunger indirectly by means of either another computer (e.g. PKI), or a human (e.g. a PGP key-signing party or a buck-passing web browser). In this case, Abacus' connection to Bitmunger is at most as reliable as the weakest link in the chain of trust connecting it to Bitmunger.
- **Cold Call:** Abacus may have no direct or indirect connection to Bitmunger, in which case it must simply fling a packet addressed to Bitmunger into the Internet and hope that the public key in the response originates with Bitmunger and not an imposter. This is the situation that currently causes web browsers to decide instead to pass the buck.

The sparseness of direct encounters distinguishes them from cold calls, and makes them difficult for an attacker to accomplish. This critical observation allows us to capture a feature of human social relationships, that encountering somebody face-to-face is inherently special and more trustworthy than encountering somebody indirectly. Direct encounters break the all-to-all connectivity of the Internet and give a basis for forming relationships with strangers — if nothing else, I know that we two were at least really in the same place at the same time.

The leap of faith is always there, but the more direct the connection, the harder it is for a malicious machine to exploit it. Although direct encounters are mostly unused at present, they are the smallest leap of faith, and we will use them to shrink the leap needed during a cold call.

### 4 Shrinking the Initiator's Leap of Faith

Abacus may have to make a large leap of faith as an initiator when it makes a cold call to Bitmunger, which it has never previously met. For example, Bitmunger might be Alice's home computer, which she is trying to SSH into while she is at a conference, using her colleague's laptop Abacus and connected to the Internet via an open wireless access point. Since it's not her computer, Abacus is unlikely to have ever previously met Bitmunger. Bitmunger is far away, so

---

<sup>2</sup>Found in the text box shown in Figure 2(a).

<sup>3</sup>or from a machine that has Bitmunger's private key.

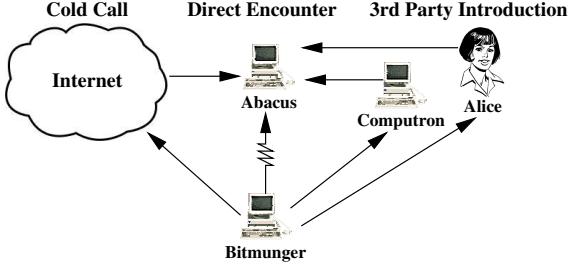


Figure 3: There are three ways Abacus can meet Bitmunger: a direct encounter (e.g. direct wireless link), introduction over secure channels by a third party (human or computer), or a “cold call” of insecure traffic through the Internet. The trustworthiness of a direct encounter depends only on Bitmunger, an introduction also depends on the introducer, and a cold call depends on every server on the path between Bitmunger and Abacus.

Abacus cannot contact it directly, and third party introduction is unlikely since PKI probably doesn’t cover Bitmunger and Alice probably doesn’t know its key. That leaves a cold call as the only means of introduction, and it must go through the dubious first hop of the unvetted access point, which could easily initiate an man-in-the-middle (MITM) attack, pretending to Abacus that it is Bitmunger in order to intercept traffic between Alice and Bitmunger (Figure 4).<sup>4</sup>

When Abacus makes a cold call to initiate a trust relationship with Bitmunger, assuming they have not previously met, it takes an identifier (e.g. a URL, server name, or IP address), resolves it, and sends a message requesting a relationship. If a third machine, Computron, is on the path taken by a message either during resolution of the identifier or by Abacus’ message to Bitmunger, it can do an MITM attack against Abacus and substitute itself for Bitmunger in the transaction, possibly initiating its own connection with Bitmunger in order to perfectly simulate the transaction. Computron thus compromises the secure channel between Abacus and Bitmunger for its own purposes.

Routing and resolution services in the big pipes of the Internet are difficult to compromise, so the greatest threat for MITM attacks is at the edges, where administration is more lax and attackers can easily introduce their own computers to the network. Thus

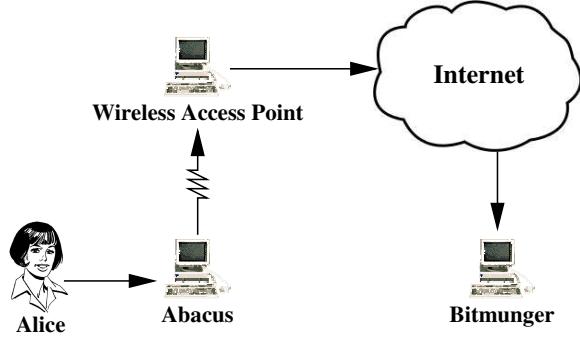


Figure 4: Abacus is faced with a problem. Alice has asked it to connect to Bitmunger, a server it has not met and can only be introduced to via a cold call mediated through an unfamiliar wireless access point. The access point is in an excellent position to play man-in-the-middle, pretending to be Bitmunger in order to intercept traffic between Alice and Bitmunger.

the majority of the threat can be reduced to three scenarios: first-hop MITM, last-hop MITM, and bad references. In the first-hop scenario, Computron links Abacus to the Internet, and thus can always intercept messages since all traffic in both directions passes through it. The last-hop scenario is the same, except that Computron is between the Internet and Bitmunger. In the bad reference scenario, on the other hand, Computron does not interfere with routing, but instead tricks Abacus into choosing to send messages to it instead of Bitmunger.

#### 4.1 Random Buddy Verification

Abacus can resist first-hop MITM attacks (e.g. the Alice conference scenario) by leveraging the ability of computers to meet directly. Although most computers are sessile and directly meet only a few machines during their lifetime, laptops and other mobile devices, may travel frequently and directly encounter many other computers in places like airports, hotels, coffee-shops, and family gatherings.

Number of encounters per computer is likely to be distributed as a power law, derived from the power law distribution of human encounters,[2] but that same scale-free network makes it likely that any sessile or encounter-poor computer can either directly encounter or be introduced to a trustworthy mobile device which directly encounters many computers.

If the mobile device takes advantage of direct encounters to meet a few of these strangers, then it can

<sup>4</sup>This is precisely the situation that wags at security conferences like to use to humiliate their colleagues.

build a collection of random buddies with whom it can establish secure channels. Further, it can share its wealth by introducing a few of these random buddies to the sessile computers it encounters.

When Abacus makes a cold call to meet Bitmunger, then, it can shrink its leap of faith with the aid of a few of its random buddies (Figure 5). Abacus establishes secure connections to its buddies Deadbeef, Electro, and Frotz, which are likely to be connecting to the Internet through servers other than Computron, and asks them to make cold calls to Bitmunger. If they report back the same key that Abacus has received from Bitmunger, then Abacus knows that, unless all of the buddies it chose are in league with Computron or connecting to the Internet through it, that the key it has received really is from Computron, thus heading off first-hop man-in-the-middle attacks.

If not all of the buddies report, or if not all of the reported keys are the same, things are more complicated, since there are many possible reasons for the difference, including first-hop MITM, lying buddies, and discrimination on the part of Bitmunger. The cause of the disagreement and whether it is due to malice or accident is not something which Abacus can hope to discover reliably, but maintaining estimates of the trustworthiness of its buddies may give good indications as to the situation. Certainly, however, Abacus can inform Alice of the particulars of the situation, allowing her to make a more informed decision as to whether she wants to go ahead and communicate anyway.

#### 4.1.1 Threats

We have identified the following threats to verification via random buddies:

- **Last-Hop MITM:** Diverse paths through the Internet do not do any good if the only way to reach Bitmunger is through Computron.
- **Bad Reference MITM:** If Computron can trick either Alice or Abacus into choosing to initiate a relationship with it instead of Bitmunger, then random buddies do not help, since Abacus will send them to the wrong location as well.
- **Poisoning the Random Buddy Pool:** If too many of Abacus's random buddies are malicious (e.g. a large set of fake identities generated by Computron), the system breaks down. If Abacus chooses a set of verifying buddies which are in collusion with Computron, the buddies can persuade Abacus that Computron really is Bitmunger. Alternately, if one of Computron's identities is cho-

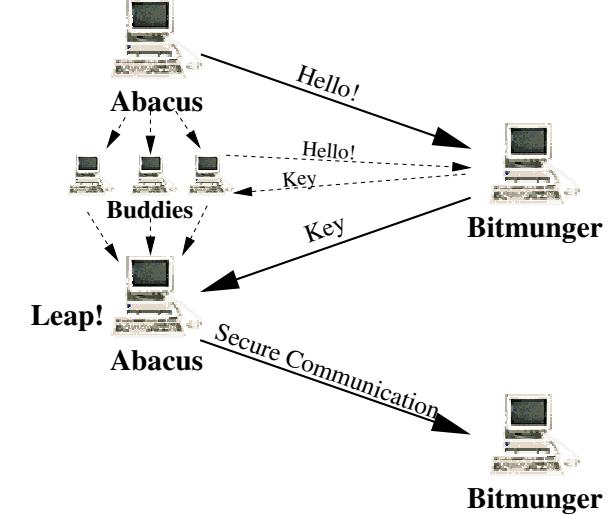


Figure 5: When Abacus initiates a relationship with Bitmunger, it makes a leap of faith in deciding to accept the key claiming to originate from Bitmunger. Abacus can shrink this leap of faith by asking a few random buddies to get keys from Bitmunger as well, since its buddies will likely take substantially different paths to get to Bitmunger.

sen as a verifying buddy, it can return a false key, casting doubt on a legitimate connection.

- **Monoculture Buddies:** If Abacus' buddies have too little topological diversity, it cannot gain an independent perspective by asking them to verify Bitmunger. A single buddy in an office filled with computers is just as valuable as its 100 others all sharing the same gateway to the Internet.
- **Gaming the System:** Computron can be assumed to know the code which Abacus is running, and can attempt to game its way into adding as many fake buddies as possible in a short time, being the most attractive buddies in the pool, or more trustworthy than they should appear. This might be diffused by having the system parameters wander chaotically through a wide range of plausible settings.
- **DDoS threat:** An attacker which gathers a large number of buddies might use them to amplify its denial of service attacks by asking them all to verify the same server. Limiting the rate at which buddies are willing to perform verifications should defuse this threat while not impeding legitimate users.

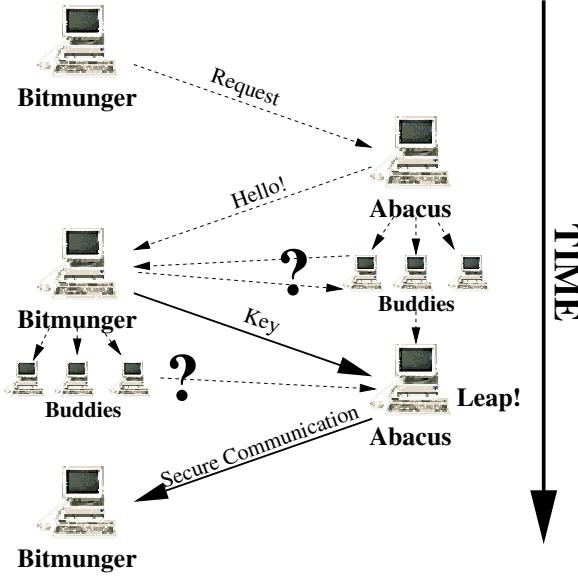


Figure 6: When Abacus responds to Bitmunger’s request to initiate a relationship, it has less leverage than when it is the initiator, since Bitmunger’s identity might be evanescent. When Abacus is the initiator, on the other hand, Bitmunger can be assumed to be a valid identity since Abacus is choosing to connect to it. Random buddies might be deployed, either Bitmunger or Abacus, to try to establish that Bitmunger is a real identity, but the mechanism is still unclear.

## 5 Shrinking the Responder’s Leap of Faith

When Abacus is responding to a cold call from Bitmunger rather than initiating it, it is faced with a different type of threat. Now the threat is that Bitmunger may be an evanescent identity generating unwanted traffic (e.g. DoS attacks or spam). Previously, Bitmunger could be assumed to be a real identity since Abacus chose to connect to it. Since fake identities are easy to come by, this is a much more difficult problem, and we do not yet have a clear solution.

We suspect, however, that a deployment of the random buddy system, as in the initiator case, might be able to shrink the leap of faith (Figure 6), as a diverse collection of Abacus and Bitmunger’s random buddies could be used to establish the reality of Bitmunger through methods such as:

- **IP address diversity.** It’s harder to fake many different addresses than a single one, so Abacus

might to return routability tests on a group of buddies willing to vouch for Bitmunger.

- **Topological/geographic diversity.** Similarly, buddies willing to vouch for Bitmunger which Abacus’s buddies can verify are near them in different parts of network or physical space gives credibility to Bitmunger’s claim to be real.
- **Distributed Reputation.** With some sort of distributed reputation system, Bitmunger might have to work to establish itself as legitimate in the reputation system before Abacus is willing to trust it.
- **Friend-of-a-Friend relationships.** A chain of buddies trusted by Abacus might lead to a computer willing to vouch for Bitmunger.

## 6 Final Thoughts

We have observed something novel and highly under-exploited in today’s Internet: there is something special about being able to communicate directly with another computer. Because a computer can only communicate directly with a few other computers at any given time, this sparseness may provide a basis on which to build trust with strangers.

A few mobile computers carried by highly mobile humans can encounter many other computers, generating a scale-free network of relationships in which every computer can expect to be introduced to a diverse collection of recognizable strangers.

Although we do not present complete solutions to the problem of extending trust to strangers on the Internet, we hope we are successful in suggesting an architectural direction, in which computers become acknowledged participants in the trust network.

## References

- [1] RSA Laboratories. “RSA Security - 2.1.1 What is public-key cryptography?” in *Crypto FAQ* <http://www.rsasecurity.com/rsalabs/node.asp?id=2165>
- [2] Newman, M. E. J. “The spread of epidemic disease on networks.” *Physical Review E*, 66 (016128), 2002.
- [3] *Microsoft Internet Explorer 5.2 for Mac*, Version 5.2.3 (5815.1), Microsoft Corporation.