

Public-key Support for Collaborative Groups

Steve Dohrmann, Carl Ellison

steve.dohrmann@intel.com, cme@jf.intel.com

Intel Labs

Abstract: *In this paper, we describe a use of public-key cryptography to achieve access control over communication and data transfers in order to support the work of collaborative groups. The participants form themselves into groups and access is granted to group members. The use of cryptography in this project is exceptional only in the care with which we designed the protocols for identity establishment. Our goal is to produce a working application that has the potential to be more secure than earlier alternatives, because it is easier to use correctly. This paper compares our identity establishment process, along the lines of SDSI, to that of an X.509 PKI or PGP, and shows the security advantages of the process we use. We also describe an experimental method for key verification intended to make strong key verification both easy and enjoyable for the average user.*

1 Introduction

This paper describes a project to build and run collaborative groups over ad hoc networks with strong access control for communications and data transfers, strong encryption for the privacy of those interactions and strong but easy to use administration of access control. It was our initial premise that cryptography and protocol development had achieved adequate security long ago, and yet weaknesses remained in fielded implementations that came primarily from human mistakes attributable to user interface elements [6], such as

1. confusion when the user is forced to deal with unfamiliar concepts,
2. mistaken identity when referring to people by name, or
3. the simple refusal to employ security features because of a distasteful user interface.

It was our intention to address these issues and thus make a family of devices that improve on the security offered by PKI-based mechanisms, such as PGP, S/MIME, and SSL. We want to handle corporate sensitive data with improved security while simplifying the user interface to the extent that an untrained user at home would use the system correctly. We stop short of implementing MAC (Mandatory Access Control) and labeling of data, although that is an area for future development.

The devices we use are PDAs and laptops. These are mobile computing platforms, and in our prototype implementation they are connected by wireless networking, although nothing in this design rules out interoperating with wired devices. Because we are using wireless networks, we have no control over who might connect to that network. We have no secure perimeter and therefore do not rely on one. In retrospect, this appears to be a good design choice even for wired networks, since it is becoming difficult, if not impossible, to establish a secure perimeter in wired networks as well.

We take as our paradigm of collaborative group the pattern we experience at Intel, where groups are formed to address tasks, perform their function and then dissolve when the function is complete. Such groups remain active anywhere from a half hour to years. These groups are formed via personal invitation (sometimes indirectly, via a referral from an invitee) and are constructed based on availability and needed skills without any special regard to the corporate organization chart. As a result, it is not uncommon for an individual to be a member of multiple groups and be the only participant in common among those groups. It is also not uncommon to meet more than one new person in each new group a person joins. These groups might address extremely sensitive matters, such as designs for new features for future microprocessors, but they might also address non-sensitive matters, such as planning an annual departmental party or raising money for a needy family. We assume that this model covers more than just Intel. It applies clearly to people's behavior at home. If there is a more structured work environment where task groups are constrained by an organization chart, such constrained groups can still fit into our model.

Although we envision creating small collaborative groups, typically the size of a group one would find in a conference room, the mechanism defined here scales easily to a community of any size. Meanwhile, even though the group may be small, the population from which we choose that group is large, up to the size of the global Internet. This introduces a naming problem, discussed in more detail below. It is that naming problem that would make a global PKI unacceptable for our purposes, even if such a PKI were to exist. Fortunately, from our experience there is no need for such a global PKI. Instead, we expect to see a

proliferation of the kind of public key authentication and authorization mechanism that we have implemented and that we describe in this paper.

This paper describes the full process of achieving strong authorization of communication and file access. In section 2, we cover physical discovery of other devices. In section 3 we cover the process of establishing identity of other participants, specifically of linking their identities as established biometrically with their keys as provided over the network. That section is perhaps the most controversial and accordingly occupies the bulk of this paper. In section 4, we describe the process of group formation, based on identities that have been established by the methods of section 3. In section 5, we list some of the uses to which these groups can be put. In section 6, we consider some user interface issues, especially the issue of key verification – something vitally important for security but something that most users find annoying and wish to skip entirely. In section 7, we give our conclusions and in section 8 we consider areas for future research.

2 Discovery

In our current implementation, we use laptops or PDAs with dual networks, one local-area (802.11 ad hoc) and one wide-area (GPRS). The discovery mechanism is different for the two, not merely because the underlying hardware is different but because the population size is radically different. Under 802.11, one would expect fewer than 200 machines within range. Under GPRS, there might be millions of users online (just as there would be on the whole Internet). In neither case do we trust information obtained by discovery without the further proof that is provided during the identity establishment phase, but in both cases we need to find the party with whom we intend to do that identity establishment.

2.1 GPRS Discovery

Discovery here is by sign-on name, a name programmed into the cellular card at time of service activation. It is by these names that the cellular provider identifies and catalogs subscribers. These names are arbitrarily chosen and not necessarily known by the person encountering the name, so they are not necessarily meaningful to users. They are used as indexes into a database, typically under verbal instruction.

As part of our project, we have created a directory to be operated by the cellular provider, in which we record presence information: whether a given subscriber is online at the moment and the current IP address of that

subscriber. Discovery over GPRS is achieved by consultation of that directory. Write-access to that directory is authenticated strongly, via public key operations, using a key installed during provisioning and bound to the user's sign-on name. This key is empowered only to give directory access and is not used for other access control.

2.2 802.11 Discovery

With 802.11, there is no need for a sign-on name, but in order to be consistent across networks, we invent and use a sign-on name for the 802.11 discovery process as well. This is a potential weakness. There is a certain level of security provided by the GPRS discovery mechanism, since one must be strongly authenticated to place an entry in the presence directory. When the directory under GPRS returns an IP address for a given sign-on name, one can rely on the fact that the binding of name to IP address was strongly authenticated and was provided by the holder of that sign-on name. However, under 802.11, there is no authentication of the sign-on name. It is merely a claim. If the user had been trained by GPRS experience to rely on the validity of this name binding, this is not safe. We do not rely on our users to keep that distinction between GPRS and 802.11 in mind. As a result, a machine that has been freshly discovered over either network is assumed by our system not to have been authenticated at all and is not granted any restricted access until after Identity Establishment and Authorization.

3 Identity Establishment

During the **introduction** phase, we establish the identity of correspondents. The identifier we use is not the sign-on name, for two reasons.

1. We allow a user to generate multiple personae and use them as she sees fit, choosing which one to use in introducing herself, just as a user chooses which business card to beam from a PDA.
2. We do not believe in one-name-fits-all-uses. The login name, introduced in the 1960's (or even earlier), is a good method of identifying a person to a computer, but we have seen numerous failures in attempts to use such names to specify a person, through a computer, to another person.

The description of **introduction**, which follows, may seem pedantic and perhaps elementary. However, we have tried to show all our steps so that we can compare this process to that used by a more traditional global-name PKI such as X.509 or PGP.

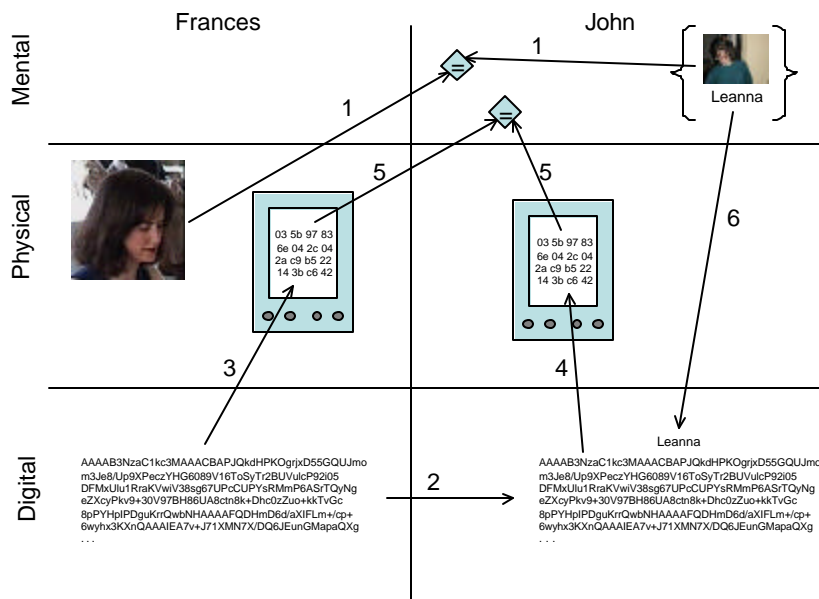


Figure 1: The Process of Establishing Identity

Within the computers and over the network, nodes in our networks are known by various transitory addresses, such as an IP address, but also by a permanent, globally unique ID: a public key associated with the user's chosen persona. The **introduction** job is therefore to establish the identity of that key.

By "establishing identity of a key", we mean **establishing that the key belongs to the person you think it does**. The phrase "the person you think it does" implies that you have some concept of the person. If you have never met the person, and therefore have no concept of him or her, the phrase has no meaning and you cannot establish identity. The most you can do in that case is to learn facts about that keyholder based on statements by some other party. However, here we are interested in establishing identity.

In our analysis of the introduction process, we look at three slices of reality:

1. **Digital:** things that reside in and happen inside computers and networks (keys)
2. **Physical:** people and things that have physical existence (people, computer screens), and
3. **Mental:** thoughts and memories inside a person's mind (knowledge about a person, biometric matching procedures, decision making, etc.).

"**The person you think it does**" exists in the mental slice of reality. It is a body of memories about the person in question. The purpose of introduction is therefore to establish a binding between a body of

memories and a public key. This implies that the introduction phase requires personal acquaintanceship. Our system does not limit all system use to personal acquaintances of one person. Non-acquaintances are made accessible during the **invitation** phase. But, we do block system use by complete strangers (those not known to anyone in the collaboration group).

3.1 The Identity Establishment Process

Each person who is party to an introduction operates in three different spaces: one physical, one digital and one mental. For mutually establishing identity between two parties, there are then six spaces involved, and steps in the process used to establish identity must cross from one space to another. The boundary crossings must be considered carefully because they offer increased likelihood of errors.

In Figure 1 we show the process of introduction from Frances Chamish to John Wilson. Ms. Chamish does not use the name Frances, except on official documents, like her driver's license, passport and income tax return. With everyone else, she uses the name Leanna. So, we will refer to her by the name Leanna, unless we are being formal.

The process described in Figure 1 might be mistaken for the PGP key signing ritual, but it is different in that it does not assume knowledge or relationships that are not actually present. [The comparison to the process used by a traditional PKI like PGP or X.509 is given in section 3.2.]

The identity establishment process of Figure 1 has six steps.

1. John sees Leanna and since he knows her already, he compares the person he sees before him to a template stored in his memory. This is a biometric comparison, based on face or voice recognition and possibly other characteristics, processed by John's senses and brain, rather than some hardware biometric sensor. A similar biometric comparison would happen if the encounter between them were by telephone or videoconference.
2. As part of normal background activity, Leanna's and John's PDAs broadcast **discovery** messages containing their sign-on names and IP addresses. By mutual agreement, Leanna and John start the **introduction** phase by releasing their public keys and associated information to each other's PDAs, using the IP addresses learned during the discovery phase. [Figure 1 shows only one half of this exchange.]
3. John wants to change Leanna's key (an entry in his Contact List) from anonymous to known. This requires a verification phase. For Leanna's part of that phase, she displays verification graphics of her public key, on her PDA. [In Figure 1, this is shown as a key hash, but it could be any appropriate display carrying enough entropy to verify the key.]
4. John's computer simultaneously displays the verification graphics of the newly arrived key.
5. John compares these two images, by seeing them displayed on the two PDAs, held side by side (or if they are connected over a telephone connection, he listens to Leanna read displayed data or listens to her computer and his own simultaneously render verification data as sounds). From this, John now knows that the key he has selected in his PDA is the one belonging to Leanna. He knows this in his own brain, the same brain that established Leanna's biometric match in step 1. Therefore, those two match results are communicated to his decision-making without having to cross reality-slice boundaries.
6. With the success of the two equality tests, John gives a name to that selected public key using the name "Leanna", which is the name he uses to index his set of memories that include her biometric templates. This name comes from his memory and its sole purpose is to be a link back to his memory from his computer display. It does not have to be a

name that anyone else would recognize as belonging to Frances L. Chamish. This binding of the name Leanna to her public key must be protected from tampering. John establishes that protection by leveraging the protection of his own private key. He creates a SDSI [5] name certificate binding the name "Leanna" to her public key and signs the certificate with his private key. After John has accepted and labeled Leanna's key, future encounters with her will not require any of the steps of this introduction process. Her key remains marked as fully introduced.

At the conclusion of this protocol, John has a Contact List entry that ties a public key to a body of memories, including one or more biometric templates, that stands for his concept of the person he calls Leanna. In other words, **he has established that the key belongs to the person he thinks it does.**

The relationship established here is immediately between John's mind and John's PDA's digital memory (with linkage by use of the name "Leanna"). There is a secondary linkage to Leanna's private key, by virtue of the fact that a given public key has only one corresponding private key, at least in our public-key algorithms. From there, there is a linkage to any digital signature made by Leanna's private key, and from there to any message or file thus signed.

This process has been tuned to link information via identifiers appropriate to the domain in which they are used. Between John's mind and his PDA's memory, a local name, meaningful only to him, is used. Between John's PDA and Leanna's PDA, a globally unique identifier (the public key) is used.

3.2 Establishing Identity via Traditional PKI: X.509 or PGP

X.509, PGP and SDSI ID certificates differ in format, process and meaning. The difference in format is irrelevant for this paper. We focus on the difference in process and meaning. Most especially, we note that both X.509 and PGP deal with globally unique IDs that are expected to be meaningful to whoever intends to use the key. Since this ID carries a global meaning, the binding of ID to key is an act that must be performed by a trusted service. In X.509, that trusted service is a specially trusted Certificate Authority (CA). In PGP, that trusted service is a collection of less trusted key signers who, taken together, constitute a distributed trusted service (the web of trust). By contrast, the SDSI (local) names we use are intended to have meaning only for the person who creates the name and binds it to a key. That one person is the sole authority on this name

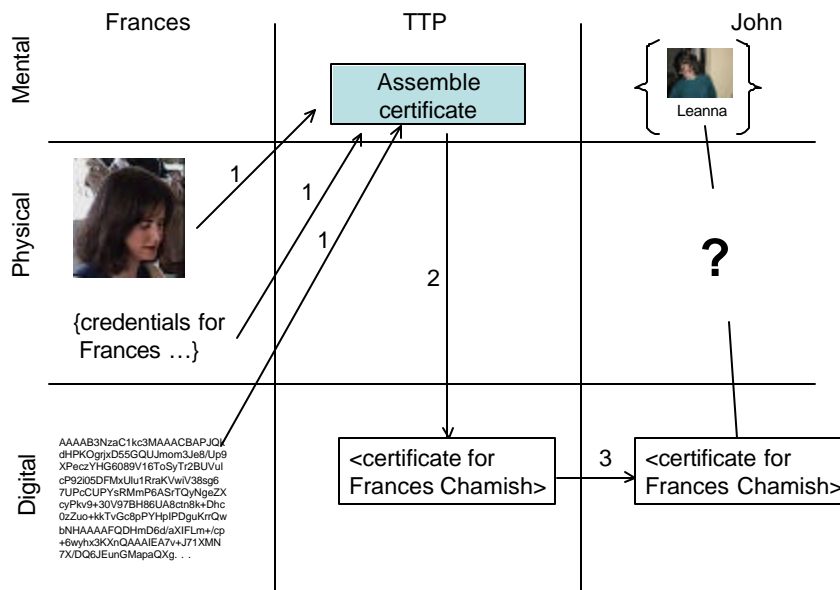


Figure 2: Establishing Identity via PKI

binding and therefore the only one who can bind that name to a key.

In Figure 2 the “person” labeled “TTP” stands for a Trusted Third Party and can be either an X.509 CA or a set of PGP trusted introducers.

3.2.1 TTP Process: Leanna to John

The process of Figure 2 appears simpler than the process of Figure 1, because it omits the detail effort involved in creating a certificate. In the case of PGP, for example, that effort often involves the hash computation and comparison steps shown in Figure 1.

The process shown in Figure 2 is:

1. Leanna takes various credentials and a copy of her public key to the TTP. At PGP key signing parties, those credentials might include a driver’s license or passport. By means of these credentials, Frances lays claim to her true name. That is, she demonstrates to the TTP that she is not impersonating someone else. These official credentials all list Leanna as “Frances Chamish”, some using the middle initial “L”.
2. The TTP instructs his or her computer to generate a certificate binding Leanna’s name, “Frances Chamish”, to her public key. In the case of PGP, the certificate construction will have been done already by Leanna and the TTP(s) merely sign(s) that certificate body. In the case of an X.509 CA, the TTP builds the

certificate and most likely chooses a name for Leanna in the process. PGP does not require that IDs in certificates be globally unique, but X.509 practices often require name uniqueness, at least over the set of individuals certified by that CA. As a result, the X.509 certificate will bind a Distinguished Name (DN) to the public key, where that DN may include the name Frances Chamish but may also include other information to make the DN unique.

3. The certificate issued to/for Leanna is delivered to John at a time when John is not in direct contact with Leanna and he must make a decision based on the information contained within that certificate. This delivery can be via a directory service (e.g., the PGP key server or some directory of X.509 certificates) or from Leanna as part of a communication (e.g., via S/MIME). If he is acting properly, he will fail to make any connection between the certificate and his memory of Leanna, since the two have too little information in common to confirm with high probability that they refer to the same physical person.

Note that PGP has a slight advantage here. Under PGP, Leanna chooses the name she wants bound to her public key and needs only to convince some number of key signers to sign that association. On the other hand, a high quality PGP key signer should refuse to sign a key with a name not backed up by official documents.

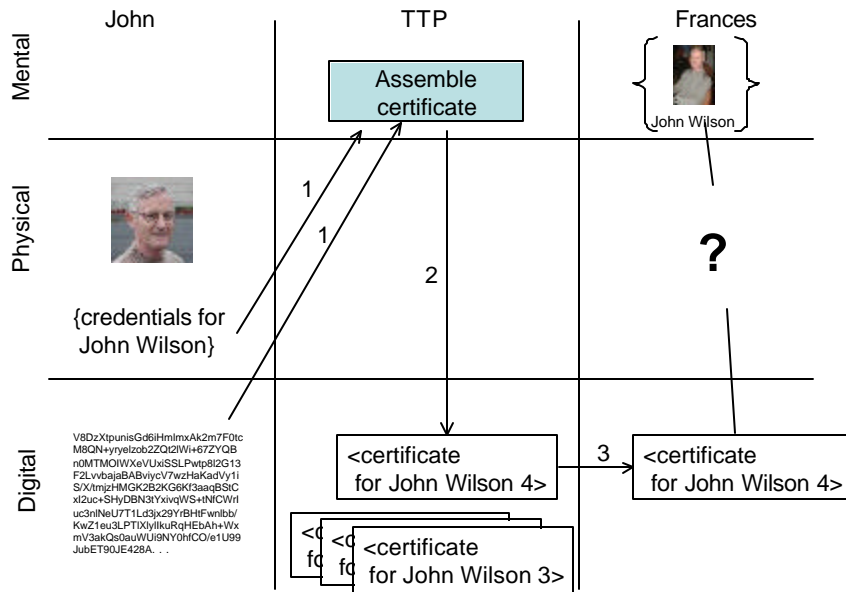


Figure 3: Second PKI Example

3.2.2 TTP Process: John to Leanna

In the other direction, there is a different problem. John Wilson uses the same, true name in all his official credentials, on all his documents and with all people. But, in Figure 3 we see that Leanna is still unable to connect his certificate to his identity in her mind. Although the names compare between the certificate and Leanna’s memory, Leanna does not know which of the TTP’s John Wilsons this certificate corresponds to. She knows only one John Wilson, but the TTP might know and have certified hundreds. It is true that a good CA will make the certificates for each John Wilson different, by including additional information beyond the common name “John Wilson”. (That information is shown in Figure 3 as serial numbers.) However, if Leanna does not know this additional information about John, then all of these certificates would equally match Leanna’s memory of John and therefore the certificate in Leanna’s computer could be for any of those John Wilsons. In the best case, she will discount the certificate as worthless to her because she knows she doesn’t know which John Wilson it belongs to, but there is a more serious threat. She does not get all of the certificates issued to all the John Wilsons. She gets only one, especially if it is delivered (e.g., by S/MIME or SSL) from someone claiming to be John Wilson. If she were a naïve user, she might not think about the hundreds of other John Wilsons that the TTP could have certified and, since she knows only one John Wilson, accept the offered certificate as referring to the John Wilson she knows. That is, she might assume that

she has verified John’s identity via that certificate when she hasn’t.

By contrast, when Leanna creates a SDSI name certificate with the name “John Wilson” by the process of Figure 1, since she knows only one John Wilson she knows to which John Wilson her certificate refers. If she knows more than one John Wilson, then she must choose additional information to append to the name to make it unique for her, just as a CA needs to do. However, she will choose information that she knows and that should therefore be meaningful to her when she gets around to using that certificate in the future.

3.3 Security of Private Keys

There may be suspicion of the personal introduction processes we use for their lack of use of a CA. As we have shown above, the use of global names that comes along with using a CA adds substantial insecurity to the introduction process. However, an X.509 CA is expected to be very good at protecting its own keys. In our mechanism, by contrast, certificates are generated by keys that are not specially protected. In PGP, the key signers do not specially protect their keys, but the fact that a key is supposed to be signed by multiple signers (the web of trust) implies that any attacker must have compromised all of those keys. PGP aims to achieve through redundancy what an X.509 CA tries to achieve through a guarded vault. At some number of signatures, the attack effort required becomes greater for PGP than for an X.509 CA and therefore the strength of PGP would be greater. Our certificates have neither form of protection.

In spite of the relatively unprotected signing keys in our mechanism, we can show that we have lost no security for lack of the TTP. At the same time, as shown in the previous sections, we would have lost security via naming had we used either of the global-name ID mechanisms.

Our argument is that if an attacker can steal (or operate at will) a user's private key, that attacker can impersonate the user as well as generate certificates. Since confidentiality keys are established in our system by signed Diffie-Hellman key agreement[3], forward secrecy is maintained and the attacker does not gain access to any past (recorded) messages or file transfers. This is not to deny the severity of theft of a private key. The ability to impersonate the attacked user is a wide security breach. The ability to generate certificates as that attacked user, however, does not give any extra access. No user in our system is in the role of a TTP – certifying memberships, IDs or authorizations that the attacked party does not herself possess and therefore that the attacker does not himself possess after theft of her key.

If the attacker chooses to use the stolen key to generate a certificate for his own key, to invite it to join a group (see section 4, below), then the attacker would have access to activities of that group as a full participant without continued use of the stolen key. However, he would also leave a trail of use of his own key. That key, although not tied to any locator information, is an identifier and has forensic value. Therefore, a savvy attacker would continue to impersonate the attacked person by using her stolen key, rather than generate a certificate giving group membership to his key.

In summary, the theft of a private key is undesirable, but the ability of the thief to generate certificates gives the thief no powers beyond those already gained just by possession of the private key and might, in fact, work against the attacker. A TTP would not increase private key security on an individual node. It would only increase certificate-issuing security, and therefore is of no benefit to us.

4 Group Formation

We start with the concept of a secured collaboration, or **collaboration** for short. A **collaboration** is a group of principals, known as **members**, who are permitted to share messages and files as part of that collaboration. Some of these members also have the permission to add new members to the collaboration.

A collaboration starts out as a name in the namespace of the creator of the collaboration. It is expressed as an SPKI/SDSI name: “(name <public key> <ASCII name of collaboration>)”. [4]

The creator of a collaboration might be a private individual, creating a set of friends, or a project leader in a corporation, creating a digital reflection of her project team. The official or unofficial nature of a collaboration is a function of the intention of the creator and does not show up in any difference in the software used.

Given correspondents who are known with assurance, the process of **Invitation** is that of granting authorization to those known correspondents to participate in a secure collaboration. An invitee can be granted membership in the collaboration and might also be granted the right to invite others into that collaboration.

We grant membership without permission to add new members by creating an SPKI/SDSI ID certificate:

```
(cert
  (issuer (name <public key> <ASCII
name of collaboration>))
  (subject <public key of invitee>)
  (valid (not-after <end date>)))
)
```

We grant the ability to add new members as well by issuing the certificate:

```
(cert
  (issuer (name <public key> <ASCII
name of collaboration>))
  (subject (name <public key of
invitee> <large random value>))
  (valid (not-after <end date>)))
)
```

That is, we create a named group in the grantee's namespace and add that named group to the collaboration. That grantee then adds individual members to that new named group, via certificate:

```
(cert
  (issuer (name <public key of
invitee> <large random value>))
  (subject <public key of next
invitee>)
  (valid (not-after <end date>)))
)
```

The members of a collaboration are those public keys that are direct members of the top level named group or of some named group contained within that top level group, at whatever nesting depth.

4.1 Cross-corporate Invitations

In our system, invitations are issued only to acquaintances, but these do not have to be close

personal friends. These can be people one had met for the first time just prior to issuing the invitation.

Such might be the case with cross-corporate working groups, such as standards bodies, corporate acquisitions or venture capital funding activities.

The invitation process does not require an act of the IT departments of the various corporations involved. It does not give any access into any of the corporations by members of the other except for the strictly limited functionality of the collaboration for which the invitation was issued. In this way, it models current business practices.

Other PKI mechanisms for permitting cross-corporate interactions do not share this attribute. A bridge CA [1], for example, effectively merges the certificate space of the two bridged corporations. The very existence of the bridge CA might, in fact, leak sensitive information (for example, evidence that an acquisition or merger is in the secret negotiation stage).

By contrast, with the **invitation** process, corporation A learns nothing about the employee database of corporation B. Members of corporation B are represented in the group as public keys. No names of keyholders are exchanged as part of the invitation. One does not know if a second key invited by someone in corporation B was that of another employee or was a second key of the original employee. Therefore, one does not even learn anything about the headcount of corporation B beyond that which was learned during the in-person negotiation meeting(s) during which the **introduction** phase crossed the inter-corporate boundary.

5 Use

From the point of view of the user, the collaboration tool is just another instant messaging tool that happens to operate over dual networks and offers peer-to-peer file sharing. It happens to have a peculiarly rigid introduction process, but we are tuning the prototype to make sure that that process is not onerous.

The user has no choice over whether or not to use cryptography and, if so, how strong. User keys are all 1024-bit DSA. All messages and file transfers are encrypted with 168-bit triple-DES CBC, with session keys and IVs derived from 1024-bit D-H key agreement. All messages and file transfers are digitally signed. This use of cryptography is transparent to the user.

Full details of the features of this prototype belong in a product data sheet rather than this paper, but that data sheet has not been written yet. In summary, then:

1. Users can send messages to
 - a. an entire named group,

- b. a set of members of a named group, or
 - c. a single member of a named group
2. Users can make files available to a named group
3. Users can fetch a file that is available to a named group from the machine that holds it
4. Users can send files as if attached to a message (i.e., addressed the same way)

With every operation, a group must be specified. It is the named collaboration group that constitutes the only access control at this time. That is, in order to keep the UI simple, we provide for only one level of access control. If you are in the group, you can read any message or file made available to that group.

Each computer in a group maintains state for that group, including the list of group member keys and the list of any files that have been made available to the group. Whenever two group members regain contact, they synchronize this group state. The synchronization is automatic and gives users the impression of common state, although at times of network partition, that common state loses consistency.

The resulting use model is very basic and we hope easy to understand. Wider trials of the prototype will let us confirm that hope or give us information with which to improve the user's experience.

6 User Interface issues

It is essential to do proper cryptographic engineering, both in writing code and in designing protocols. However, that careful engineering is not sufficient to achieve security in an end-user product [6]. The user interface needs to be designed in such a way that the user would naturally do the correct thing and avoid doing the wrong thing.

We must assume that there is always an attacker trying to gain access to our collaborations, even though we realize that in most cases there will be no attackers.

This lack of evidence of attack makes motivation of the user especially difficult. It is therefore incumbent upon us to make the user interface as pleasant and simple as possible

Computer software engineers, no matter how well meaning, cannot be expected to get a user interface right. There must be extensive testing, with real users. We have just started that extended testing and cannot report full results at the time of this writing. However, we have learned a number of things that are worth reporting here. These are cases where lessons we learned go against the inclination of our own developers.

6.1 Minimizing Choices

We have found that we need to minimize choices and options, especially when there might be a bad choice. Our initial users are more comfortable when given fewer options.

We have limited options by defining a Contact (a Java Object) that goes through state changes. It starts out, after **discovery**, as an anonymous, non-trusted thing. It has a sign-on name and may have an IP address. The only thing that can be done with this non-trusted object is to engage in **introduction**.

After introduction, the Contact has a public key that has been verified and named by the user. An introduced Contact is only then available to participate in **invitations** to join one or more named collaboration groups.

All message traffic and file transfers are associated with a named group and are limited to members of that group. It is not possible to engage in messaging or file sharing outside a named group.

Groups and Contacts are shown to the user as names, but the state of a Contact is shown by color and icon so that the user does not need to look beyond the top-level screen to tell what can be done with the Contact.

One invites a Contact to join a group by dragging and dropping the Contact name onto the group.

6.2 Sign-on Names

For security purposes, it is best not to display sign-on names to the user at all. These names are weak identifiers at best and are subject to the John Wilson problem, described in section 3.2.2.

On the other hand, both developers and experienced users have been well trained to use sign-on names. Many users view sign-on names as a way to deliver a message – e.g., the name “fundude”, or the name “fund00d” that conveys a slightly different message.

In the prototype, we have compromised. We use sign-on names during discovery, but have the person who is building a personal Contact List choose a name for each entry in that list. This name will probably turn out to be the original offered sign-on name most of the time and we expect that to be a potential weakness, due to the John Wilson problem, with or without actual attacks.

We take it as ongoing work to look for a solution to this problem that is acceptable to users.

6.3 Key Verification

We recognized early in the development process that key verification (e.g., the comparison of hex key fingerprints) is the geekiest, slowest, most painful and most cumbersome part of the **introduction** process.

This can be made a little easier by converting key hashes to lists of words to be read aloud, as PGP did several versions back. However, the task is still time consuming. The list of words from a SHA-1 hash on a PGP key takes on the order of 24 seconds to read. The hex version of the hash takes about the same length of time.

When keys are verified over a telephone connection, in our prototype, we currently have the two parties read words alternately to each other, to achieve mutual verification of the hash. However, when keys are verified by placing two mobile computers next to each other, so that the person receiving the key can verify correct receipt, we can use a graphical mechanism that permits entropy comparisons to be much faster.

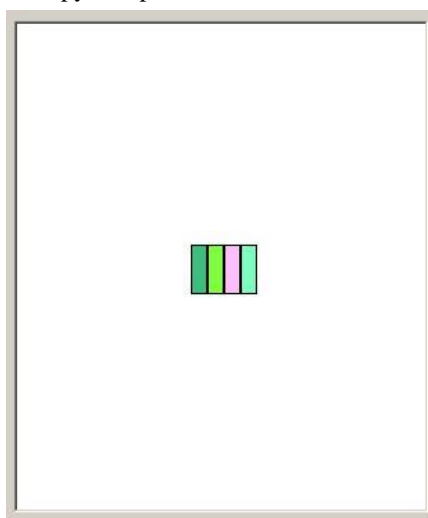


Figure 4: Verification graphic

Figure 4 shows a PDA screen displaying a graphic that we call a “flag”. Preliminary experiments show that people can compare a time sequence of these apparently random graphics on two side-by-side screens, at a rate of 2 per second, with comfort. Assuming the verifier is not color-blind, each flag carries 25 bits: 1 for horizontal vs. vertical orientation; 6 for the color of each rectangle. This rate needs to be confirmed by more extensive testing, but assuming it is confirmed, this permits a key hash comparison at 50 bits per second. One can then compare a full 160-bit hash in just over 3 seconds, for a speed-up of a factor of 8.

If the graphic is black and white, e.g., for fully color-blind users, we expect to get at least 10 bits/second of comparison, for a full 160-bit hash in 16 seconds, but we have not yet experimented with shapes to see how much more rapidly we can do comfortable entropy comparison.

It is our goal to get the verification time low enough that a user would verify the correctness of a key’s hash in the time it takes to move a stylus or mouse to accept a key as valid. This does not eliminate the verification

step, but does permit it not to add time to the user's process.

7 Conclusions

The problem of making sure that only those who truly should be authorized to access some data actually end up with access to that data is a very hard problem in general. We have addressed a subset of the family of security policies that have this requirement. We provide for policies in which every member of a defined group is permitted the same access as every other member, but we allow for the definition of an arbitrary number of groups. We have been very careful to make sure that groups are made up only of individuals known personally by someone with the authority to add to the group membership. This does not cover all possible groups, in theory, but does cover all groups we encounter in practice, both at work and at home.

Prior to this work, we had observed that the greatest leakage of confidential information came from misdirection of communication, through name confusion, and only secondly from a failure to employ security mechanisms to protect data. To respond to those problems, we have been careful to keep the named people and groups that any individual must deal with down to the personal acquaintances and group memberships of that individual. No choices are made from a larger namespace. We allow the individual to choose his or her own names for these individuals and groups, to minimize confusion of names. We have made all communications encrypted and digitally signed, with no user choice, so that all accesses to data handled by this system must be via the access control mechanisms we have defined.

This system is doubtless not perfect. However, it has addressed the greatest needs we have identified and further improvements can follow as we gain experience with use of this system.

8 Future Work

We have chosen not to deal with revocation of keys or of authorizations (group memberships). The underlying SPKI mechanism supports a variety of revocation methods, but the complication of the user interface did not seem warranted for what are almost always short-lived groups of long-lived keys.

The limitation of operations to group members, and the labeling of files as available to a group, can be thought of as a poor man's MAC/DAC architecture. We could possibly improve the security of our mechanism by implementing it on top of an operating system that supports data labeling and mandatory access controls.

We need to continue our user trials of key hash comparison mechanisms, including audio trials alongside graphical ones, in order to determine the actual number of bits being compared per second by the user.

Because our underlying engine is the AuthCompute library from CDSA [2], we have the full power of SPKI and SDSI at our command. However, we have not found a reason to use all that power. It is still an open question whether the refinement of access controls full SPKI would make possible would be of use to a naïve user base or whether it would add an unacceptable amount of confusion. For example, it is possible to use the SPKI **threshold subject** mechanism to have more elaborate security policies – such as permitting access only if a group member is also still employed and has a non-revoked key. However, this functionality would require a complication of the user interface and might lead to more errors than the extra refinement of authorization would prevent.

9 Acknowledgements

We want to thank Leanna Chamish and John Wilson for granting permission to use their names, stories and images in this paper. We especially want to thank the many engineers within Intel Labs who collaborated in producing the prototype whose characteristics we report here.

10 References

- [1] Bridge-CA: for one discussion of a bridge CA, see <http://www.bridge-ca.org/english/index.html>
- [2] CDSA: <http://developer.intel.com/ial/security/>
- [3] Whitfield Diffie and Martin Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, November 1976, pp. 644-654.
- [4] Ellison, Frantz, Lampson, Rivest, Thomas, and Ylonen, "SPKI Certificate Theory", RFC2693, September 1999.
- [5] Ronald L. Rivest and Butler Lampson, "SDSI - A Simple Distributed Security Infrastructure", September 1996, <http://theory.lcs.mit.edu/~rivest/sdsi10.html>
- [6] Alma Whitten and J.D. Tygar, *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0*. Proceedings of the 8th USENIX Security Symposium, August 1999, <http://www.usenix.org/publications/library/proceedings/sec99/whitten.html>