Modular PKI Status Report 9 September 1997

Background:

-September 1996 [redacted] develops modular PKI concept that accommodates key recovery for law enforcement, data recovery for users, privacy, and authentication. Modularity of design allows for components to be mixed and matched as appropriate, in a cost efficient, commercially viable manner.

-April 1997 patent application filed with U.S. patent office. C1 performs COMSEC evaluation: design is secure except tier-1 applications are reticent to escrow keys.

-June 1997 IS3G considers formal release of PKI proposal. Paper is ready and has been circulated among a selected few in industry/academia (IBM,Microsoft, BBN, Dorothy Denning, etc.) with measured response. No strong criticism or praise.

IS3G actions: (?) means I am not directly aware of event.

-N5 to outline basic principles of paper for DIRNSA review prior to release. (?).
-NIST is to be notified (?)
-D/DIR to notify Undersecretary of Commerce (?)
-D/DIR to schedule formal presentation of PKI at Deputies Meeting (?)
-[redacted] to undertake proof-of-concept demonstration of modular PKI using COTS products throughout. Minimize the amount of "home-grown" code required.

(b)(1)
(b)(3)-P.L. 86-36

Proof of Concept Demo:

-Tasking to complete bare bones proof of concept demo based on COTS products, all unclassified.
-Cross agency team formed to design and build PKI (X3,R2 [redacted] Q).
-JAVA demo completed (all code locally produced).
-Microsoft Exchange based demo nearing completion. This demo will work on Windows 95 and Windows NT platforms and support all the basic PKI functions:encryption, authentication, data recovery, and key recovery.
- October 1, team will travel to Microsoft in Redmond to discuss technical issues with Microsoft CAPI developers.

Future Issues:

-Dissolution of IS3G (?) hinders corporate support for PKI goals.
- Lack of interest in proof of concept demo.
[redacted]
[redacted] PKI design can be made operational in *months* on most platforms.

Personal aside: this represents some real exciting work that is more relevant today than when designed, given current administration policy.

Approved for Release by NSA on 02-08-2007, FOIA Case # 19136

DOCID: 3097966

~~CONFIDENTIAL//MR~~

Prototype and Demonstration Specifications

A Modular Public Key Infrastructure for Security Management

DRAFT

Abstract

A set of specifications and comments are outlined for developing and demonstrating a prototype based on the [redacted] *modular public key infrastructure (PKI) which supports key escrow for law enforcement, a separate data recovery component for restoration of archived information, secure message encryption, and strong authentication.*

(b)(1)
(b)(3)-P.L. 86-36

~~CONFIDENTIAL//MR~~

~~FOR OFFICIAL USE ONLY~~

DOCID: 3097966 ~~CONFIDENTIAL//MR~~

(b)(1)
(b)(3)-P.L. 86-36

DRAFT

Introduction

It is assumed the reader is familiar with th[redacted] Modular Public Key Infrastructure for Security Management, simply referred to here as the PKI. This paper outlines the one-week effort of a small technical group to develop a set of specifications for both a prototype of the infrastructure and a demonstration of the unique features of this PKI. The charge to the group was to lay the groundwork for demonstrating proof of concept in two areas:

1) validating the PKI design, and
2) determining if Commercial Off the Shelf (COTS) products could be used in the process.

This portion of the paper only deals with the specifications of the demonstration and the prototype. To start, notation and parameter specifications are covered in a Background section. The demonstration specifications detail what a demonstration of the PKI is to involve. The prototype specifications give some details about each of the five major applications to be implemented along with what general network communications will be required.

There are a number of issues left as "to be determined," and other issues which only building the prototype will surface. Some of the major decisions to make the implementation of the prototype easier, and thereby possible within a six month period are

1) All communication between the user and the other components in the PKI can be accomplished via e-mail.
2) Signed certificates will be placed in a certificate directory in a file named with the user's distinguished name, which we assume is *sid@nsa*. This directory will be exported to the world with read permission only. The certificate authority will be the only entity having write access. Thus, NFS and normal file manipulation programs can be used.
3) Only one escrow agent will be used. To streamline communication, the escrow agent will send the user's public encryption parameter's directly to the certificate authority.

In the current prototype description, at least on the surface, there seems to be no need for additional secure communications channels other then those provided by the prototype. As the prototype is build, this may need to change.

~~CONFIDENTIAL//MR~~

~~FOR OFFICIAL USE ONLY~~

Background, Notation, and Parameter Specifications

This section defines the notation to be used throughout the rest of the paper and provides a general overview of the PKI functions. In addition, the value, size, and/or scope of parameters are specified for use in the prototype and demonstration.

1. **Notation:** The following notation will be used in describing the specifications.

Global Parameters:

Universal 1024 bit prime *P*, base *G*=2, and maximum exponent size *E*=512.

The value of the prime *P* in hexadecimal (stored with most significant byte first) is

```
Length: 128 bytes
E193214D6D81118715C136EF8EE164ABF0CD994DAE4D7AF10B411533F4E3A731
6913AF71C1CB5FE13E39D91BAE672D53F8CF38430B0D3B9D3DFB5A81384B3BFD
974976F14335043D0AA3A1C7C991B907DFA19C410555EA858957E01395878157
2C4FC609CCBD4C3BA4F5D4099C5B0D696B0977293DD9B0FD5353D303222BBA93
```

Universal signature exponent e = 0x10001 = 65377.

Users: A, B, C

User IDs: User A has identification ID_A, the format to be determined.

Hashing Algorithm: Generically denoted by *H*. Prototype implementation will use the 160-bit SHA1.

Signature Algorithm: Message digest is generated using SHA1 and signed using RSA encryption.

Users Secret Signature Parameters: User A generates two secret primes p_A and q_A, and a secret exponent d_A such that $e \times d_A \equiv 1 \mod ((p_A - 1)(q_A - 1))$. For the prototype, the primes will be 1024 bits.

Users Public Signature Parameters: User A's public signing parameter is the 2048 bit composite modulus $N_A = p_A \times q_A$.

Users Signed Messages: $\langle M \rangle_A$ indicates that user A has signed message M.

Users Secret Encryption Key: User A generates a secret random number *a*, satisfying $2^{E-1} \leq a < 2^E$.

Users Public Encryption Key: User A's public encryption parameter is $G^a \bmod P$.

D R A F T

Certificate Authority: CA, with encryption secret f and associated encryption public parameter $G^f \bmod P$. Messages M signed by the CA are indicated by $\langle M\rangle_{CA}$.

Key Recovery Center: KRC, with encryption secret k and associated encryption public parameter $G^k \bmod P$.

Escrow Agent: EA, with encryption secret e and associated encryption public parameter $G^e \bmod P$. Messages signed by the escrow agent are indicated by $\langle M\rangle_{EA}$. Only one escrow agent will be used for the demonstration.

Encryption algorithm: Generically denoted by E, with cryptovariable denoted by CV. $E^{CV}(M)$ denotes the encryption of the message M.

Default Encryption Algorithm: DES. DES to be used in the cipher block chaining mode with padding (FIPS 46-1 and FIPS 81) using padding scheme in PKCS #7.

Other Encryption Algorithms: Possible use of other algorithms, namely Triple DES and/or RC4 is to be determined.

Key Exchange Key: User A in sending an encrypted message to user B, will generate a key encryption key, denoted by *AB,* used to encrypt the cryptovariable *CV* employed in the encryption process. User A will send with the encrypted message the encrypted variable, that is, $E^{AB}(CV)$.

DRAFT

2. **Policy Authority**

Normally the root of any certification chain is the Policy Authority. The prototype will assume only one CA which will also act as the Policy Authority. This begs a number of issues and serves to simplify the prototype implementation. These issues are not unique to this design and solutions developed for other efforts will also work here.

3. **User's Distinguished Name**

We assume for the prototype that user sid's are unique within NSA, so if *sidforA* is user A's sid, then *sidforA@nsa* would be a unique identifier for user A.

4. **Certificate Data Base**

User certificates will need to be available to all on the network. There are many aspects associated with servicing, maintaining, and using such databases. Protocols are currently being established, and many different methods can be used. For the prototype, we will take the easy way out, and assume that a user's certificate(s) is available in a file named with the user's distinguished name.

The directory containing these files is called the certificate directory, and will be nfs exported to everyone, read only. The certificate authority will be the only one having write access to the directory. Thus, we will be relying on NFS and the normal directory and file structure to handle the certificate server functions. It is clear that for a large number of users, this is not a reasonable method.

DRAFT

5. **Certificate Authority**

For the prototype, only one certificate authority will be used, so there will be no need for a policy authority. Users will only need to verify a single chain of trust, namely between the CA and user. The prototype will treat the CA as a privileged user, having normal signing and encryption capabilities with all parameters chosen in the range specified for the normal user. The CA's privilege is write access to the certificate database.

6. **Identification for User A**

It is not the intent of this prototype to become involved with any controversies concerning user identifications, distinguished names, and other aspects concerned with identifying a user's privileges and capabilities. The decisions made here are ad hoc.

For prototype purposes, the user's identification information will be taken from Searchlight and include the following information

Full Name	32	ASCII Character Field	Left Justified, Blank Filled
SID	8	ASCII Character Field	Left Justified, Blank Filled
UID	8	ASCII Numeric Field	Left Justified, Blank Filled
Organization	16	ASCII Character Field	Left Justified, Blank Filled
E-mail Address	32	ASCII Character Field	Left Justified, Blank Filled

For Example: The user ID for rawisni would be

```
000000000011111111112222222222333333333344444444
012345678901234567890123456789012345678901234567
[redacted] 10125                                    (b)(3)-P.L. 86-36

445555555555666666666677777777778888888888999999
890123456789012345678901234567890123456789012345
[redacted]     [redacted]@[redacted].nsa            (b)(1)
                                                    (b)(3)-P.L. 86-36
```

7. **Certificates**

Information about User A's public signature and encryption parameters need to be publicly available. Currently, the industry seems to be reluctantly embracing use of the X.509v3 certificate structure for this purpose. While it is not the intent of this prototype to become involved in any controversies surrounding use of the X.509v3 certificate, the prototype should make an effort to use this structure. This may have a direct bearing on the assumptions made about the user ID above. Details will need to be worked out as the prototype is built.

8. **Digital Signatures for User A**

The prototype will need to supply a method for user A to generate secret and public parameters for the signing algorithm and incorporate the public parameters into a certificate to be signed by the CA.

Some mechanism will be required for the CA to authenticate A before signing the certificate. This mechanism needs further definition, but, perhaps, an Identifying Authority (IA) could be used. The advantages are that the IA can be local to the network and aid the user in forming the certificate for the CA's signature. The disadvantage is that this is another layer of trust. How this will work for the prototype needs to be determined. One possible mechanism is to have the IA, who we assume is already set up for both signing and encrypting submit the request to the CA on the user's behalf. Another possibility is that the IA gives the user a token to use for communication with the CA.

Note, there will be no escrowing of user A's secret signing parameters. The prototype will need to supply to the user a mechanism for securely storing the secret signing parameters. It is expected that this will differ between implementations.

DRAFT

9. Signature Algorithm

A message digest *H(M)* is generated by applying the hash function H to the message M to be signed. User A then "encrypts" the digest using the RSA secret decrypt exponent d_A associated with the user's public modulus N_A, i.e. $(H(M))^{d_A} \bmod N_A$. The message digest is small (160 bits for SHA1) compared to the RSA modulus size of 2048, so appropriate padding may need to be determined.

The signature can be verified by "decrypting" the "encrypted" message digest to obtain *H(M)*, i.e. $[(H(M))^{d_A}]^e \equiv H(M) \bmod N_A$, The recipient can then check *H(M)* against their own computation of *H(M)* from the received message.

Some additional work needs to be done to determine the exact format of the signed message.

10. Universal Key Exchange Parameters

For generating session keys, a Prime P and base G along with maximum secret exponent size E is needed. The values for these parameters have been specified above. For the prototype, these parameters will be hard-wired into the appropriate programs.

11. Encryption of a Message M from User A to User B

Assume for the moment that User A and User B both have enrolled their public encryption parameters with their respective CA's. User A wants to send a secure e-mail message to user B. A needs to obtain B's signed certificate from the certificate server. As discussed above, such information will be in the certificate directory in the file named *sidforB@nsa.* A will then need to verify the certificate up to the proper chain of trust.

Assuming one escrow agent, A computes the following session key for use with B:

$$AB = H\left\{H[H(H(G^{ab} \bmod P, ID_A, ID_B), \text{month}), \text{day}], \text{random}\right\}.$$

Notice this is a slight departure from the original PKI proposal in that an additional hash is performed, namely, $H(G^{ab} \bmod P, ID_A, ID_B)$.

A also generates a CV used to encrypt the signed message. User A then sends the following concatenated information to user B, in some properly encoded format:

$$\text{Control Information} \mid \text{month} \mid \text{day} \mid \text{random} \mid E^{AB}(\text{ID}_A, CV) \mid E^{CV}(\langle M\rangle_A)$$

Notice the (signed) message M is encrypted using the cryptovariable CV, and the CV is then encrypted using the information in AB. Also, notice that AB is 160 bits in length, but the encryption algorithm may require fewer (or more) bits for its cryptovariable. In this case, an agreement needs to be determined about how to produce the common cryptovariable to use. The Control Information will contain additional information needed to properly process the message, for example, encryption algorithms used, message indicators/initialization vectors/ salts, etc.

In order to decrypt, user B needs to obtain user A's certificate from which can be computed $H(G^{ba} \bmod P, ID_A, ID_B)$. Then, using the month, day, and random sent with the message, user B can compute AB. From $E^{AB}(\text{ID}_A, CV)$, B can recover CV and then use it to decrypt $E^{CV}(\langle M\rangle_A)$.

D R A F T

12. Encryption Enrollment for User A

The prototype will assume that only one escrow agent will be used. In addition, A must have signature capability already registered with their CA. In order for the CA to sign user A's public encryption parameter, A must first register its secret encryption parameters with the EA. The following steps are performed:

- User A generates a secret exponent a to register with the escrow agent. In the same manner User A would compute a key encryption key to communicate with another user (see part 11. above) user A forms

$$A, EA = H\{H[H(H(G^{ea} \bmod P, ID_A, ID_{EA}), \text{month}), \text{day}], \text{random}\}.$$

  A randomly generates a cryptovariable CV and uses it to encrypt the secret exponent a. A then sends the following signed message to EA

$$\langle M = [ID_A, ID_{CA}, G^a \bmod P, E^{CV}(a)]\rangle_A.$$

  Of course, user A will need to include the other necessary information as well, namely,

$$\text{Control Information} \mid \text{month} \mid \text{day} \mid \text{random} \mid E^{A,EA}(\text{ID}_A, CV) \mid \langle M\rangle_A$$

  User A will need to determine EA's available encryption algorithms, and pick one for this use.

- The escrow agent EA verifies A's signature and then, using $G^a \bmod P$, forms A,EA, recovers CV, and uses it to decrypt $E^{CV}(a)$ thereby recovering a. The EA then computes $G^a \bmod P$ and compares it against the value sent in the message. If both agree, the EA places a in escrow and sends the following signed message back to user A's CA:

$$\langle ID_A, G^a \bmod P\rangle_{EA}.$$

- The CA verifies the Escrow Agent's signature and then produces a signed certificate for user A, containing A's public encryption parameter $G^a \bmod P$.

Note, there is an issue here about how many certificates will be maintained for user A. To get an encryption certificate, user A already needed one for a signature. It needs to be determined if user A's encryption parameters should be folded into one large certificate, or should multiple certificates be used.

It is the responsibility of the CA to maintain the escrow agent's signed message for user A.

In addition, A's public signing certificate is pushed onto the certificate server for other users to access. In this case, the CA will enter additional information in the file named *sidforA@nsa* in the certificate directory.

D R A F T

13. Key Recovery

A Key Recovery Center (KRC) will be part of the prototype. The method works as follows. Suppose User A encrypts a message M using the cryptovariable *CV.* In exactly the same manner as sending a message to any user, A forms $E^{A,KRC}(\mathrm{ID}_A, CV)$, only A retains this information along with the encrypted message and other information (Control Information | month | day | random). Nothing is sent to the KRC. Note that as long as A has access to the secret encryption exponent a, the CV can be recovered from $E^{A,KRC}(\mathrm{ID}_A, CV)$.

If, for any reason, user A cannot recover the secret a, then $E^{A,KRC}(\mathrm{ID}_A, CV)$ can be sent to the KRC, requesting that the CV be recovered. Notice, the KRC during the decryption processes also recovers ID_A, thus insuring the identity of the user is known. Once verified, the KRC can return the CV to A. Since the KRC never sees the encrypted data, it cannot recover the message. Note too that this does not recover user A's secret exponent.

While there are many uses for KRC services, one in particular is pass phrase recovery. The prototype should demonstrate this capability. Recall it is up to the application to arrange for the secure storage of a users secret signature and encryption keys. Most likely this will be accomplished by having the user supply a pass phrase to a program which in turn will generate a cryptovariable for securing the required information. This pass phrase itself could be stored encrypted with the appropriate KRC information to aid it's recovery.

14. Escrow Agent

The role of the escrow agent is varied. In addition to it's intended law enforcement role, it can be used to recover a user's secret encryption key thus giving decryption capability for all messages either sent from or to user A. If time permits, the prototype should give some demonstration of the Escrow Agents functions.

Demonstration Specifications

Assumptions

The user, say A, already has an account on the LAN with access to the proper CA, KRC, EA's and other network services needed for enrollment, signature generation and verification, and encryption. The demonstration will take the user through the following steps in enrolling and using the system. The tags below represent which modules will be involved with the particular feature being demonstrated.

Tags:		
	U:	User or Client Application
	KM:	User Level Key Management Application
	CA:	Certificate Authority Application
	EA:	Escrow Agent Application
	DRC:	Data Recovery Center Application
	PA:	Policy Authority Function

DRAFT

Basic Application for Demonstration

U: E-mail will be used for building the prototype and the demonstration. Communications between any two entities should be able to be handled by e-mail alone.

Signature Verification

U: Using an e-mail tool, the user can receive signed e-mail messages and ask that the signature be verified.

Implementation Dependent: Is verification automatic, or can it be user controlled?

User Enrollment for Signature Purposes

1. **KM: User A via an implementation determined interface will generate secret and public parameters to be used with the signature algorithm.**

   The secret parameters will be stored locally in a secure fashion, protected by a user generated pass phrase.

   The public parameters will be used to generate the users signing certificate.

2. **CA: User A, after satisfactorily being identified by the CA, will provide the necessary information for the CA to issue a signed X.509v3 certificate containing A's public signing parameters.**

   Format of this certificate and it's extensions need to be determined.

   The CA will be responsible for insuring uniqueness of A's ID.

   The CA will write the user's certification into the certificate directory, as well as returning a copy to the user.

   ISSUE to Resolve: Use of an Identification Agent and its interaction with the CA.

3. **U: The user will verify that the signed certificate is correct, probably by signing a document and then verifying the signature.**

4. **U: User A can now manufacture an e-mail document and have the digital signature applied.**

   NOTE: This will require the user to uncover the secret signing parameters which have been securely stored on A's local system.

5. **U: Any recipient can also verify A's signature.**

DRAFT

Global Parameters

PA: The global infrastructure must be set up off line. It is composed of a universal prime *P* base *G*, and a size parameter, *E*, for the users secret.

It is expected these parameters will be well known and incorporated into any software.

User Enrollment for Encryption Services

1. **KM: User A via an implementation dependent interface will generate a secret encryption parameter (exponent), *a***

   A will need to securely store the generated secret encryption parameter, again, probably using a pass phrase to generate the covering variable. A can (should) also store a token for the KRC to use in recovering this secret information in the event the pass phrase is forgotten.

   ISSUE: Should this information be coupled with A's signing parameters?

   Should the same pass phrase be used/allowed to protect all secret information?

2. **KM: User A will use the secret exponent *a* to communicate with the Escrow Agent (see background) and send a signed message securely transmitting A's secret encryption parameter to the escrow agent. Information about A's certificate authority needs to be included in the message.**

   This message will be sent by e-mail. See background section for details.

3. **KM/EA: The escrow agent recovers A's secret parameters, verifies it is correct, and stores it securely. The escrow agent then computes A's public parameter, appends A's ID, and signs the information, sending it to user A's CA, the identity of which was sent as part of the message from A to the EA.**

   For the prototype, this is done by sending an e-mail message to the CA. The user A can (should) be cc'ed.

4. **KM/CA: The CA verifies the escrow agents signed information, pulls out A's public encryption parameters, and together with other identifying features of user A, issues a signed certificate.**

   This certificate is both returned to A and written into the certificate database

   The CA might also want to send A an encrypted test message for verification purposes.

   In addition, the CA must log information about A's escrow agents.

5. **U: Upon receipt of the information from the CA, A can verify the certificate against its stored secrets by decrypting the test message sent from the CA.**

6. **U: User A can now send an encrypted e-mail message to other users on the system.**

KRC Demonstration

Assume that A forgets the pass phrase needs to unlock the secret keys.

The following will/may need a GUI interface.

1. **KM:** If user A can still sign messages, then A generates a signed request to the KRC along with the token between A and the DRC which contains an encrypted version of the *CV* used to cover A's secret parameters. (See background.)

   ISSUE: Does A need to set up a secure channel with the KRC in the same way it communicated with the EA? Possibly. In reality, if the channel is not secure, it can be argued that access to the channel can only yield information, in this case the key, *CV*, which in and of itself is useless. Access to the cipher on the user's disk is necessary to recover the protected information. None the less, this is an issue worth discussing.

2. **KM/CA: If A cannot sign the request message because the pass phrase covered both the encryption and signature keys, then another method of identifying A to the DRC needs to be established.**

   One way to demo this is to have A contact the CA and ask for a temporary certificate of limited duration to be issued for communication with the KRC only.

   Another alternative is to get a new set of signature keys for identification purposes. This would require expiration of the old signature keys.

3. **KRC: Once the DRC has verified that the request has come from user A, it then decrypts the token**

$$E^{A,\,KRC}(ID_A,\,CV)$$

   recovering ID_A along with the *CV*. The KRC can now check that the *CV* was indeed meant for user A.

   If so, the KRC sends the *CV* back to A.

   ISSUE Continued: Note, this could be encrypted if the encryption feature was used in 1. Here again, the same secure channel set up by A to communicate with the KRC can be used. This requires A to hold onto the previously established variable, or to make it interactive.

4. **KM/U: User A can now use the recovered *CV* to decrypt the secret keys. A will the need to generate a new pass phase to use for securely restoring the secret parameters.**

DRAFT

DRAFT

Escrow Demonstration

Assume user A has received encrypted e-mail messages from both B and C. Assume further that B did not need to use an Escrow agent, but A has. Finally, assume a valid need is demonstrated to read the mail message from B to A. Show how A's escrowed information can be used to accomplish this without revealing A's secret information. and without decrypting messages between A and C.

Assume the e-mail message from B to A has the form

Control Information | month | day | random | $E^{BA}(\mathrm{ID}_B, CV)$ | $E^{CV}(<\mathrm{M}>_B)$

where

$$BA = H\left\{H[H(H(G^{ba} \bmod P, ID_B, ID_A), \text{month}), \text{day}], \text{random}\right\} \quad .$$

The warrant specifies that only mail messages from A to B sent on month, day can be read.

1. **The warranted authority contacts the CA for the escrow agent employed by user A.** The CA should return the signed messages $\langle G^a \bmod P, ID_A \rangle_{EA}$ submitted by A as part of the certificate application. This message contains the signed certificates for the escrow agent, thus implying its identity.

2. **A request is made to the escrow agent. The agent must verify the validity of the request, which contains the users ID's, and in this case the month and day for which decryption capability is to be given. Upon doing so, the Escrow Agent EA recovers the securely stored user secret *a* and returns information allowing the warranted authority to read the appropriate messages.**

   Also, knowing the ID's from A and B from the warrant, the escrow agent can obtain their certificates, and hence public parameters. In particular, the agent can obtain $G^b \bmod P$ from user B's certificate. The escrow agent, knowing a, can then compute

$$H[H(H(G^{ba} \bmod P, ID_B, ID_A), \text{month}), \text{day}] \quad .$$

   This information is then securely sent to the warranted authority.

3. **The warranted authority, upon obtaining this information, can now finish computing the quantity BA above and then use it to decrypt $E^{AB}(\mathrm{ID}_A, CV)$. This in turn gives *CV*, which can be used to decrypt $E^{CV}(<\mathrm{M}>_B)$, resulting in the signed message from B to user A.**

Prototype Specifications

1. **User/Client Application**

   Basic client application to prototype will be e-mail. It is expected an e-mail tool will be modified to provide additional functionality to include:

   Verification of signed e-mail messages,

   Authentication of an e-mail message by applying the users signature,

   Confidentiality of e-mail messages through encryption/decryption, and

   Establishing a verification chain of certificates between a recipient and the trusted CA.

   The e-mail application will need access to the certificate directory used by the PKI network services to retrieve certificates of other users.

2. **User/Client Key Administration Application**

   Another interface will be required for helping the user with the following functions:

   Generation of signature parameters (2048 bit RSA modulus, secret prime factors and secret decrypt exponent),

   Registration of the public signing parameters with the CA,

   Securing private signing parameters in the user's work space, and

   Testing of the users signature algorithm.

   The problem of authenticating the user to the CA is a universal one, and the prototype will not necessarily make a major effort to solve this problem.

   A similar interface will be necessary for helping the user to enroll for encryption services. Note the user must have signing capability before enrolling for encryption. The interface should guide the user through the following functions:

   Generate a secret encryption parameter and the associated public parameter,

   Securing the secret information in the user's workspace,

   Making use of the KRC for aiding the user in key recovery,

   Securely contacting the escrow agent to escrow the secret parameter,

   Receiving from the CA a signed certificate containing the users public encryption parameters, and

   Testing the users capability for decrypting/encrypting e-mail.

   Additional functions for this application might include

   Helping the user communicate with the KRC to recover a lost pass phase.

3. **CA Server Application**

   The CA will need to communicate with the client application for both the signature and encryption enrollment procedures. Functions to be supported:

   Generating and signing user certificates,

DRAFT

Posting certificates to the network certificate directory,

Verifying EA signatures as part of the client enrollment process, and

Possibly helping the user authenticate to the DRC for pass phase recovery.

A fully functioning CA will provide a number of additional services involved with administering user certificates. This level of functionality is not limited to this PKI alone and will not be required as part of the prototype.

4. **Key Recovery Center Application**

Being a unique aspect of this PKI, the KRC application will need to respond to a users request for pass phrase recovery. This might include functions to perform some, or all, of the following:

Communicating with the client, possibly in a secure mode,

Authenticating the user's request, especially if the user does not have a signing capability, and

Recovering message encryption parameters (*CV*'s, not the user's secrets), and verifying the user's *ID*.

5. **Escrow Agent Application**

The escrow agent application will need to communicate with the client in a secure mode. It will need to perform the following functions:

decrypt the users secret encryption parameter,

verify that it generates the users public parameter,

escrow the users secret, and

generate a signed message to pass on to the user's CA.

For purposes of this prototype, a data recovery via the escrow agent may not be required, unless time permits.

6. **General Network Communication**

All communications between the user's, EA, CA, and KRC can be handled via e-mail.

Since the certificate database will be in an exported directory off of a file server, NFS along with file manipulation programs can handle all necessary certificate management.

D R A F T

Demonstration Specifications

Basic Application for Demonstration

e-mail

Demo #1: Signature Verification

- Any user can validate a digital signature

Demo #2: Signature Enrollment

- **User** generates secret & public signature parameters - securely stores secret locally
- With proper ID, the **IA** signs user's public signature parameter and ships to the **CA**
- **CA** issues X.509v3 certificate for signature use only
- **User** can now sign e-mail messages

Demo #3: Enrollment for Encryption

- **User** generates a secret a & public $g^a \bmod P$
- Securely transmits authenticated a to **EA**
- **EA** escrows a & transmits $g^a \bmod P$ to **CA**
- **CA** modifies users X.509v3 certificate with public encryption information
- **CA** send encrypted message to user
- **User** can encrypt e-mail messages

Demo #4: KRC - Recovering User Secrets

- **User** has stored on disk

$$E^{\text{User,KRC}}(CV) \mid E^{CV}(\text{Secrets})$$

- **User** sends $E^{\text{User,KRC}}(CV)$ to KRC
- **KRC** recovers *CV* and returns to User
- **User** can now decrypt $E^{CV}(\text{Secrets})$

Demo #5: Escrow Usage

- User receives from Suspect daily messages

$$E^{\text{Suspect,User,May } x}(CVx) \mid E^{CVx}(\text{Msg: May } x)$$

- **EA**, with proper authorization, generates

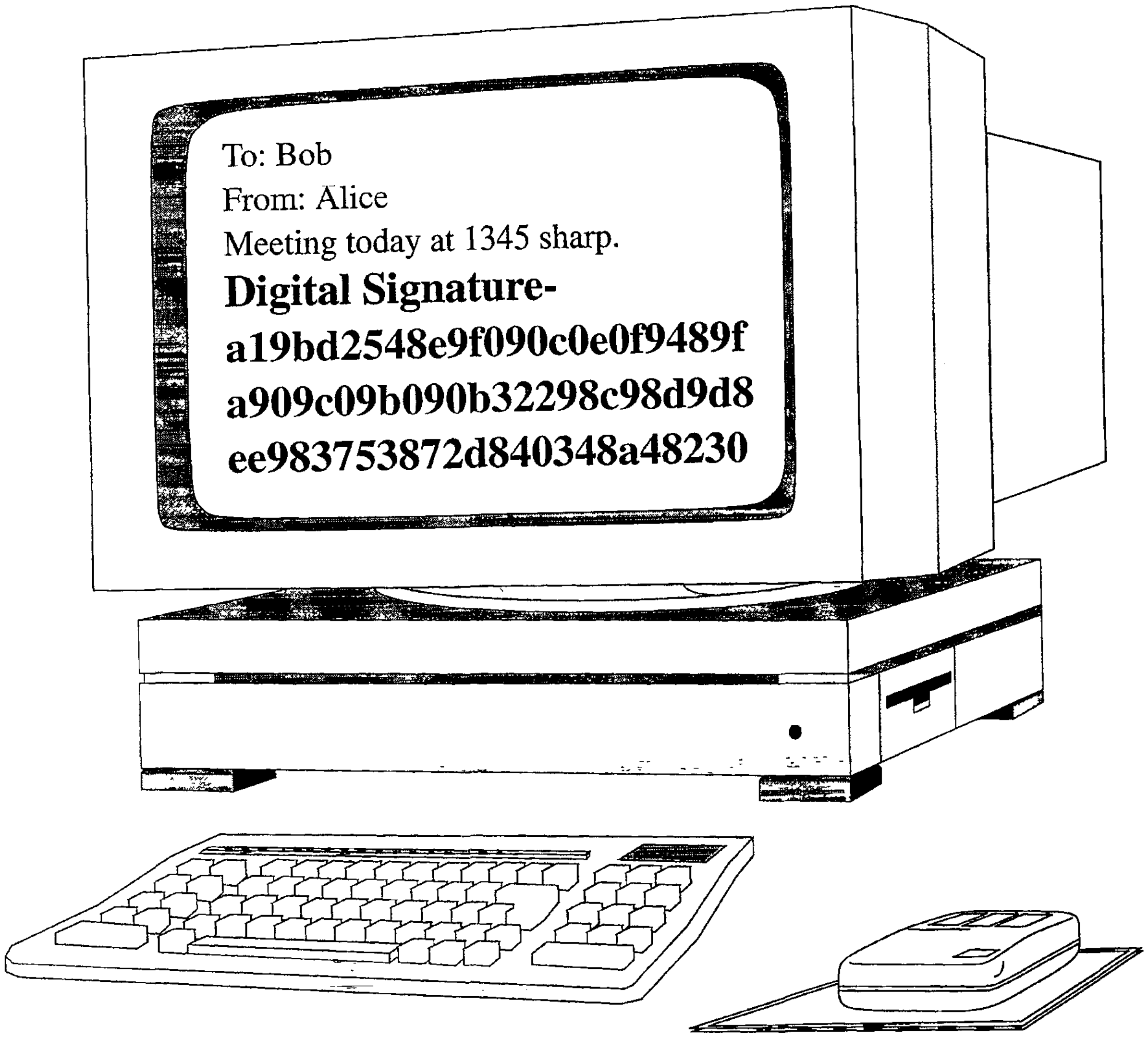
$$\text{Suspect,User,May } x$$

for a set of dates in May

- **Authorities** can only recover *CVx* for specified dates

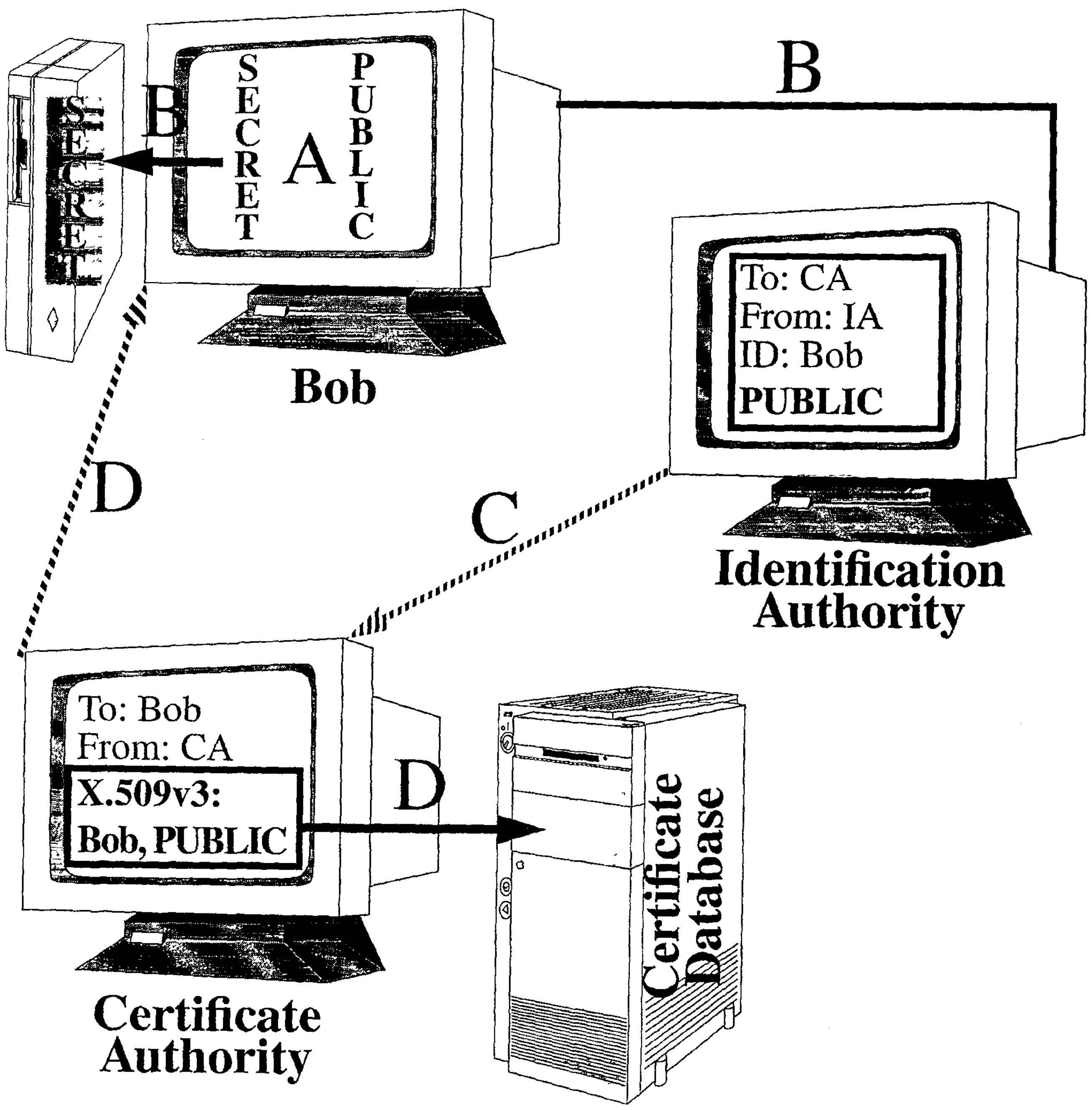
E-mail Demonstration

Demo #1: Signature Verification

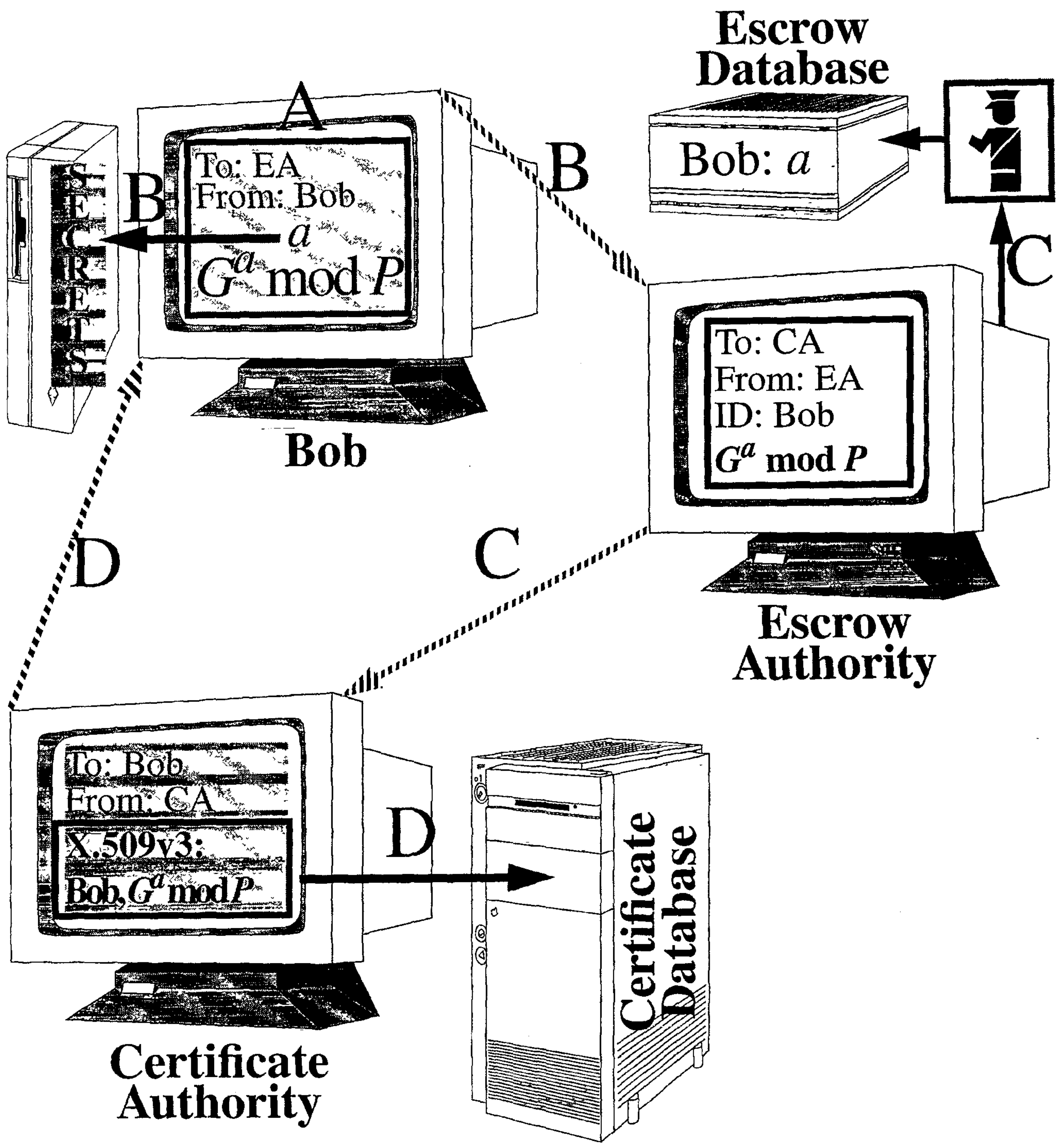


To: Bob
From: Alice
Meeting today at 1345 sharp.
Digital Signature-
a19bd2548e9f090c0e0f9489f
a909c09b090b32298c98d9d8
ee983753872d840348a48230

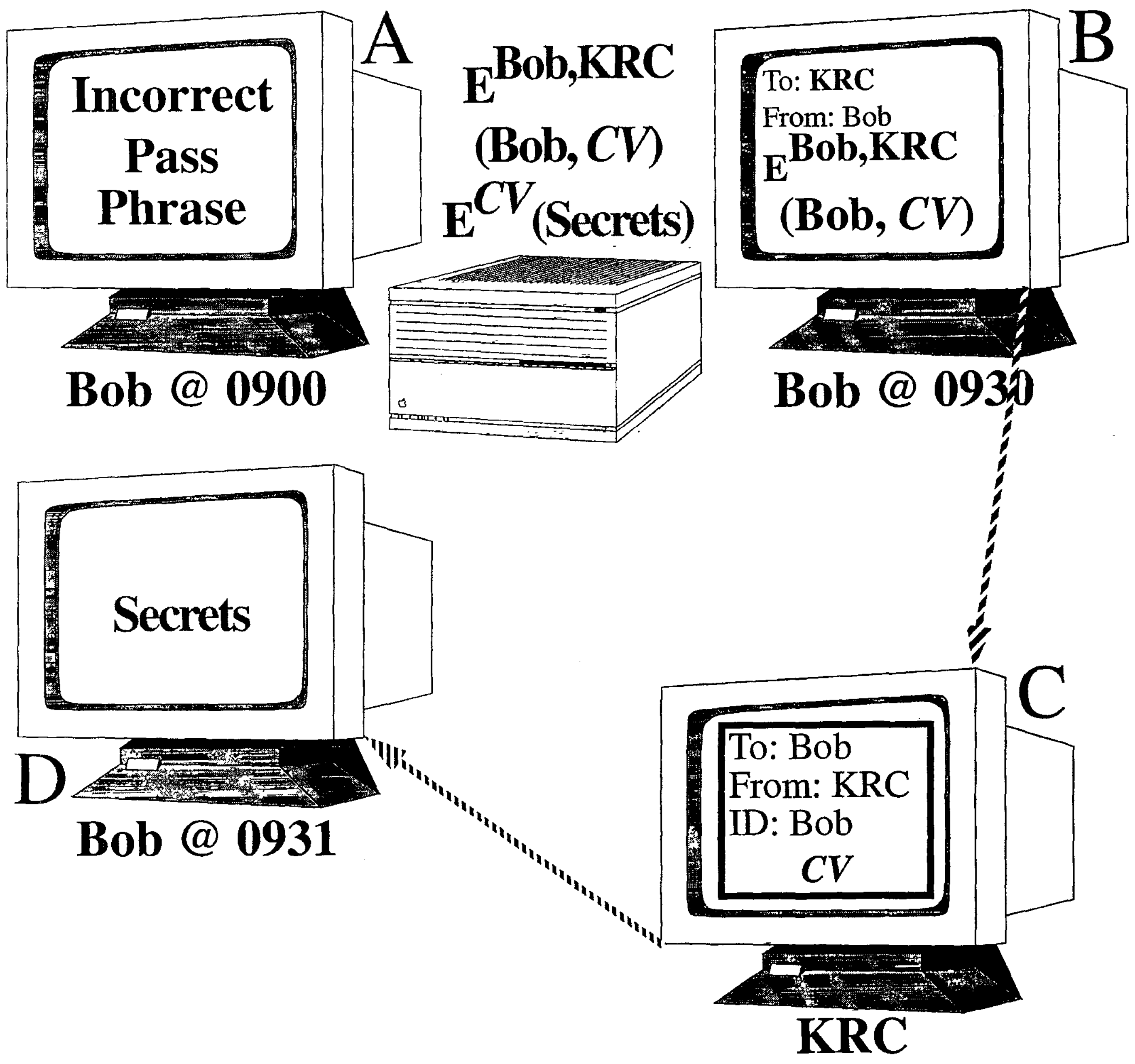
Demo #2: Signature Enrollment



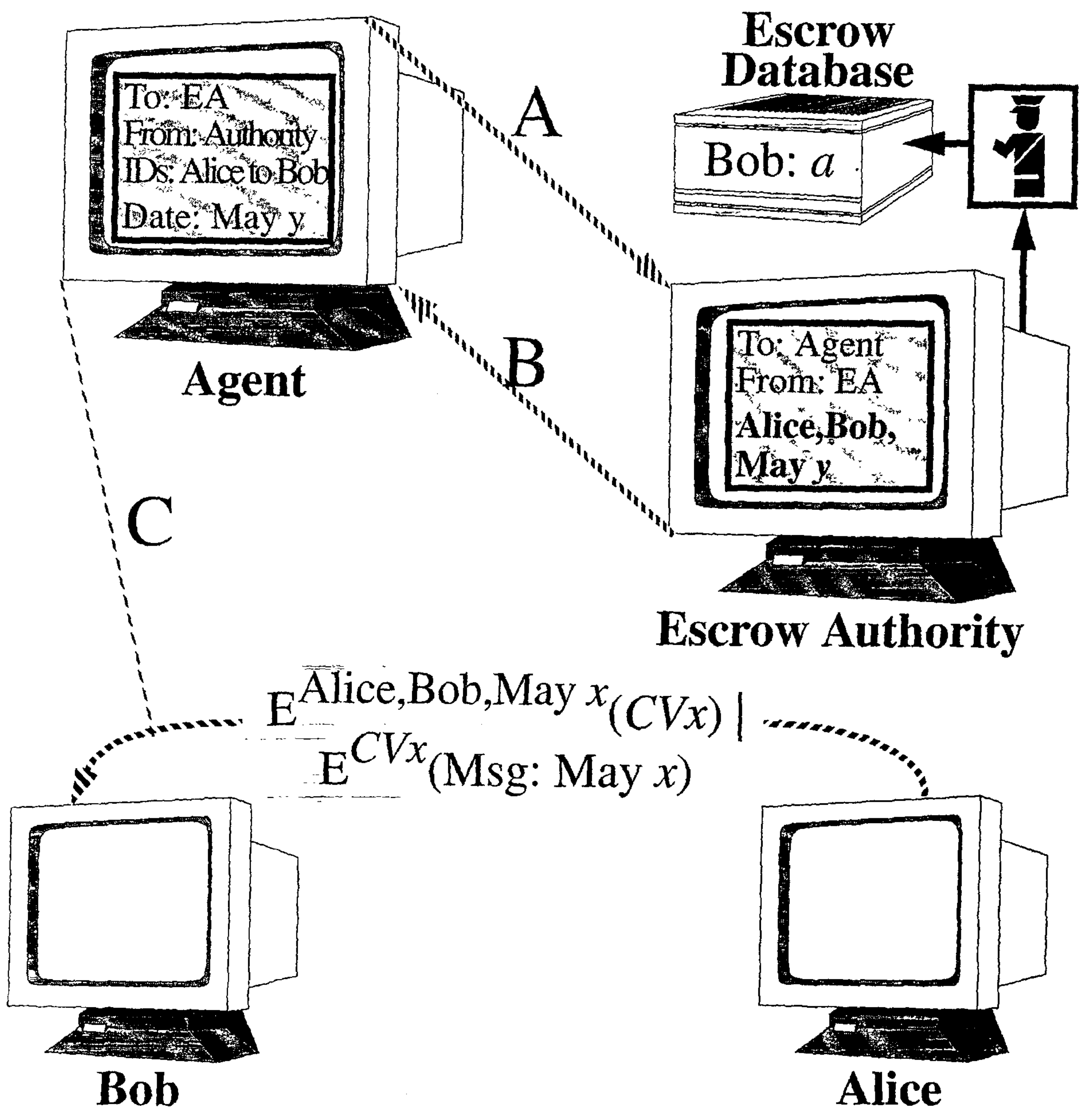
Demo #3: Encryption Enrollment



Demo #4: Key Recovery Center



Demo #5: Escrow Usage



~~FOR OFFICIAL USE ONLY~~