Failsafe Key Escrow

(Extended Abstract)

Joseph Kilian
NEC Research
Princeton, NJ 08540

Tom Leighton
Mathematics Department
Laboratory for Computer Science
MIT
Cambridge, MA 02139

August 17, 1994

Abstract

"Fair" Public Key Cryptosystems (FPKCs) have recently been proposed as a method for providing secure escrowing of keys without relying on special purpose hardware. In a fair public key cryptosystem, the cryptosystem users are allowed to choose their own public and private keys, but they must share their secret keys with a group of trustees (escrow agencies) in a manner that allows the trustees to reconstruct the secret key of any user in the event of a court order. The United States Government has recently acquired a license to Fair cryptosystem technology from Silvio Micali.

The claimed advantage of the Micali FPKC over alternative approaches to key escrow is that the user in the Micali system is supposedly assured that his or her secret key will remain protected (unless the trustees collaborate to reconstruct the secret key), and the government is supposedly assured that criminals will not be able to abuse the escrow system in a manner that prevents government deciphering of wiretapped communications.

In this paper, we expose a serious weakness in the Micali FPKC which allows criminals to abuse the system in precisely the manner which is not supposed to be possible. In particular, we show that the FPKC is subject to the sorts of subliminal key attacks discovered by Simmons and Desmedt in the 1980s [7, 17, 18, 19]. As a consequence, we show how a government-sanctioned FPKC as envisioned by Micali can be subverted by criminals to form a "shadow" public key cryptosystem that is untappable by the government. In some cases, the shadow cryptosystem is even *more* secure against the government than the original cryptosystem is against nongovernmental adversaries. Even if the shadow cryptosystem is run using only public knowledge and even if the government is fully aware of the workings of the shadow cryptosystem, there is no obvious way that the shadow system can be thwarted by the government.

In the paper, we also describe a new approach to key escrow that we call *Failsafe Key Escrow.* The Failsafe approach is characterized by the use of government-user interaction to select the secret and public keys of each user. Failsafe key escrow has all the supposed advantages of Micali's FPKC, along with a formalizable guarantee that the system cannot be abused by criminals. The Failsafe method also guarantees the government that every user's secret key will be secure even if the user selects his or her portion of the secret key poorly (e.g., by using one's birthday instead of a random number). Finally, the method can be adapted for use with any of the commonly-cited cryptosystems, and it is particularly well suited for use in escrowing DSS keys.

Keywords: Failsafe Cryptosystems, Cryptosystems, Secret Sharing, Verifiable Secret Sharing, Key Escrow, Fair Public Key Cryptosystems, Digital Signatures, Encryption, Subliminal Channels

1 Introduction

In this paper, we describe a method for escrowing cryptographic keys (which we call **Failsafe Key Escrow (FKE)**) in which the authorities interact with the users to select the cryptographic keys that are to be escrowed. The system has the following five properties:

Property 1: Each user in the system should have sufficient control over his or her secret key to be sure that the key is chosen securely.

Property 2: The central authority will also be guaranteed that the secret key for each user is chosen securely even if the user doesn't have access to a good random number generator or if the user fails to use the random number generator properly.

Property 3: Each user will be guaranteed that his or her secret key will remain secret unless a sufficient number of trustees release their shares of the key to the central authority.

Property 4: The central authority needs to be assured that it can obtain the secret key for a user who is suspected of using his or her escrowed public key for encryption in the context of illegal activities by retrieving shares of the key from a certain number of trustees.

Property 5: The central authority needs to be assured that the escrow system will not be abused by criminals in a way that helps them to communicate without fear of court-authorized wiretapping. More precisely, if two criminals abuse the FKE by using the information contained in their public keys to communicate using any published public-key encryption algorithm, and the central authority is provided knowledge of the criminals' escrowed secret keys by the trustees, then one of the following two cases should hold:

1. it should as easy (at least on a probabilistic basis) for the central authority to decrypt the message traffic between the criminals as it is for the criminals themselves to decrypt that traffic, or

2. the criminals already had a way to communicate that could not be decrypted by the government.

One can never disallow the possibility that criminals will use a completely different means for covert communication, but one does not wish to in any way assist them in this process.

The new method is substantially more secure than the Fair Public-Key Cryptosystem (FPKC) approach advocated by Micali [13]. This is because the FPKC approach does not satisfy Properties 2 and 5.

In particular, we will show in Section 3 how the public keys stored in Micali's FPKC escrow scheme can be used by criminals to communicate in a way that the Government will not be able to decipher, even if the secret keys for the users in the FPKC are provided to the Government by the trustees. The method is based on the ability of the criminal to embed a "shadow" public key within the public key that is escrowed in the FPKC [7]. Criminals can then communicate using the shadow keys whenever they want to avoid detection by the government. The shadow cryptosystem can be created without any private communication among its users. Moreover, the government has no way of knowing which keys contain shadow keys. This means that it is easy for criminals to subvert the Micali FPKC so as to prevent the Government from deciphering their communications. Such abusive use of the key escrow system is not possible in the Failsafe Key Escrow approach described here.

The Failsafe Key Escrow method described here also has the advantage of insuring that legitimate but technically unsophisticated users will be prevented (with overwhelmingly high probability) from choosing keys which are not cryptographically secure. Hence, the Government or a company can be sure that its employees are getting secure keys even if they fail to properly access a secure random number generator. Such assurances are not possible in the Micali FPKC.

The FKE method described here is no more expensive to use than (and, in some cases, it is much less costly than) Micali's FPKC technology. In addition, by Properties 1 and 3, it provides the same basic assurances of fairness to legitimate users as does the Micali FPKC. Hence, the Failsafe Key Escrow approach offers all of the benefits of FPKC while providing the substantial advantage of security for the Government as well as the unsophisticated user.

The remainder of the paper is partitioned into sections as follows. In Section 2, we provide a brief overview of the prior literature in the area of key escrow and secret sharing. The flaw in the Micali FPKC is explained in Section 3. We describe the Failsafe approach to key escrow in Section 4. Some applications of the new approach are discussed in Section 5 and its limitations are discussed in Section 6. We conclude with some acknowledgments in Section 7.

It should be understood by the reader that this paper represents a preliminary draft of a paper that will soon be submitted for publication. Many details and extensions of the results have been omitted in this version of the paper. In addition, the paper almost certainly contains errors and omissions of important references. Of course, we would appreciate learning of any comments or criticisms the reader may have in this regard.

The current draft of the paper represents the combination of the work of Joe Kilian (who discovered the flaw in the Micali FPKC that is described in Section 3) and Tom Leighton (who invented the Failsafe approach to key escrow that is described in Section 4 [11]). The Failsafe approach to key escrow is the subject of a patent filing.

2 Background

In a Public Key Cryptosystem, each user is assigned or chooses a matching pair of keys (P, S), where P is the public key corresponding to the pair and S is the secret key. For ease of access, as well as authentication purposes, the public key for each user is catalogued and/or certified by a central authority (or authorities) so that other users in the system can retrieve the authentic public key for any individual. Public Key Cryptosystems can be used for many purposes, including encryption and/or digital signatures. For a survey of the extensive literature in this area, we refer the reader to [6, 21, 15, 4].

One problem with a PKC (and Cryptosystems in general) is that they may be abused by non-law-abiding users. For example, two criminals could communicate using a PKC established by the Government and law enforcement authorities would have no way to decrypt their message traffic, even if the authorities had received a court authorization to wiretap the communication. Such activity might take place even if the PKC were established solely for the purposes of digital signatures since the criminals might use the PKC for other purposes such as encryption.

The general issue of the need for government wiretaps versus the need for individual privacy has been debated in society for decades. With the advent of inexpensive and fast cryptographic technology, this debate has intensified. This is because wiretaps can be effective against encrypted traffic only if the government can gain access to the secret key that is used to decrypt the traffic. In France, for example, all cryptographic material must be revealed to the government before it can be used. Even in Germany, where there is great sensitivity to government monitoring of individuals, the issue of government escrow of secret keys for the purposes of government wiretapping has been under discussion for many years [1].

The simplest method of key control is to have a trusted government agency (or agencies) simply escrow the secret key for each individual. Then, in the event of the proper authorization, the government can retrieve the secret key from storage and decipher the intercepted communications of a suspected criminal. In such a system, the government would have the same power that it had before the advent of public key cryptography, and the citizens would have no less privacy than before. (This is essentially the proposal made by Beth [1] to the German Parliament in 1990.)

As observed by Blakely [2], Shamir [16], and many others, however, it may be cheap to simply store copies of the secret keys, but such a solution can be corrupted. In an effort to prevent such corruption, Blakely and Shamir propose methods for splitting a secret key into n shares so that the secret can be reconstructed from any k of the shares. In addition, no information about the secret key is revealed given only $k-1$ shares. By providing each government trustee with one share of each secret key, the chances for corruption of the escrow system are substantially reduced, since the secret key of an individual can be recovered if and only if k of the trustees reveal their shares.

Since the Blakely and Shamir schemes were first proposed in 1979, a wide variety of "secret sharing" schemes have been discovered (e.g., see the survey paper by Simmons [20]).

One difficulty with the secret sharing schemes discovered by Blakely and Shamir is that there is no provision for insuring that the trustees have received valid shares of each user's secret key. Indeed, when the trustees reveal their shares under a court order (say), the shares may be found to be useless because the criminal user did not provide proper shares of his or her secret key. This problem is partially resolved in [5], where it is shown how shares of a secret can be provided in a way so that each trustee can be assured that he or she has received a valid share of the secret.

A secret sharing scheme in which each trustee can be assured that he or she has a valid share of a secret is known as a *Verifiable Secret Sharing (VSS)* scheme. Many VSS schemes are known in the literature. Typical VSS schemes proceed by having the user choose a secret m, and then publish an encryption $E(m)$ of m. The user then splits m into shares for the trustees, and the trustees verify that they have valid shares by checking against the published value of $E(m)$.

In order to be useful in the context of key escrow, it is necessary and sufficient that the pair $(m, E(m))$ form a (secret key, public key) pair of the public key cryptosystem that is being used. (This is because the secret being shared is the secret *key* of the user.) Feldman [9] and Pedersen [14] describe such VSS methods where $E(m) = g^m$ is based on the discrete-log problem. The Feldman and Pedersen VSS schemes can thus be used to share secret keys in the Diffie-Hellman, DSS, El Gamal, and elliptic curve cryptosystems. Micali provides some alternative VSS schemes based on discrete logarithms in [13], but these methods are less efficient than the Pedersen scheme. Micali also provides a VSS scheme that can be used to share secret keys in the RSA system.

In [13], Micali proposes the Fair Public-Key Cryptosystem approach to key escrow. In the Micali FPKC approach, each user shares his or her secret key with the trustees using a VSS scheme that allows each trustee to verify that they have a share of the secret key for the user that corresponds to the public key for that user. A key claim about FPKCs is that they "cannot be misused by criminal organizations."

3 The Flaw in the Micali FPKC

At first glance, it seems clear enough that the Micali FPKC cannot be misused by criminals. This is not to say, of course, that criminals can be prevented from communicating securely using secret information or an alternative escrow system, or from using other protocols for secret key agreement. But it in such situations, the criminals are not using the government key escrow system at all, and thus the government escrow system has not made it any easier for criminals to communicate (i.e., the criminals have not *abused* the FPKC). Indeed, as long as the trustees can recover the secret key corresponding to the public key in the escrow system, it would seem that the system cannot be abused.

Unfortunately, this rational makes the critical implicit assumption that the criminal is, in fact, using the same secret key for which shares were provided to the trustee. As we will now show, however, there is no reason to believe that the criminal will be so cooperative. In fact, we will describe a very simple way that criminals can exploit the Micali method for Fair PKCs without fear of eavesdropping by the government.

The key weakness we will exploit is one of the FPKCs advertised features – the ability of the user to choose his public and private keys. Our attack is therefore directed at the public-key system itself as opposed to the VSS protocol for breaking the secret key into shares. The ability of the user to select his own keys allows for a number of simple attacks using subliminal channels. The use of subliminal channels in cryptographic protocols is a well-studied field – see [17, 18, 19, 8, 7, 22]. We essentially observe that FPKCs fall prey to the same attacks, and give concrete examples of such attacks. In particular, our attack is based on the ability of the criminal to embed a "shadow" public key into the public key that is escrowed in the FPKC. In other words, the criminal can use the public key of the FPKC as a subliminal channel to advertise his shadow public key.

In what follows, we first give a high-level description of the attack, and then show how to apply it with varying degrees of efficiency to the most popular public-key cryptosystems.

3.1 Shadow public-key systems

Our attack is essentially a subliminal attack on a given public-key cryptosystem. A normal user generates a pair (P, S), publishes P and gives the government the ability to reconstruct S. In the simplest form of our attack, the attacker instead generates two key pairs (P, S) and (P', S'), where (P, S) is a proper (public-key, private-key) pair, (P', S') is a shadow key pair, and $P' = f(P)$ where f is an easily computed and publicly known function. The attacker uses (P, S) in the same way as would an ordinary user, but keeps S' reserved as his shadow secret key. In order for someone to send a truly secret message (i.e., one that cannot be deciphered by the government) to an attacker, the sender computes $P' = f(P)$ and then encrypts the message using P'. (The truly-encrypted message could then be superencrypted using P, if desired, so that it would appear as if the government FPKC were being used in the normal fashion.) The receiver of the message then decrypts it using S' (as well as S if superencryption by P was used).

The key to this approach is to find efficient ways of generating P, S, P' and S' along with an easy to compute f that generated P'. We call such a system a *shadow public-key cryptosystem.* (Note that since the attacker generates a valid (P, S) pair, and uses it in exactly the same way as does a legitimate user, the trustee verification protocols will not detect any cheating.)

3.2 A shadow public-key system based on RSA

Our attack is most straightforwardly implemented against the RSA cryptosystem. Recall that an RSA public key is of the form $P = (n, e)$ where $n = pq$ is a product of two primes and e is some exponent which is typically represented as a number mod n.[1] We first note that e is essentially unrestricted. Thus, given a security parameter k (e.g., where the k-bit product of two $k/2$-bit primes is considered hard to factor), one can encode k bits in e. This is already enough to encode the public key to Rabin's public-key cryptosystem or to public-key cryptosystems based on discrete logarithms (such as the Diffie-Hellman scheme), using the same security parameter k.

As observed by Desmedt [7], an attacker can publish roughly $k/2$ additional bits in the escrow system by suitably choosing n. Given a string m of approximately $k/2$ bits (we ignore small factors that will not affect the theoretical analysis or practical utility of the attack), an attacker can choose a random $k/2$-bit prime p, and then divide p into $2^{k/2}m$ to obtain a q and r such that $pq + r = 2^{k/2}m$ and $r < 2^{k/2}$. If q is also prime, then choose $n = pq$, in which case m is contained in the higher order bits of n. Otherwise, start over with a new p. Making reasonable assumptions on the distribution of primes,

[1] Mathematically, it is an element of $Z_{\phi(n)}$, but this is irrelevant to how it is represented, especially since $\phi(n)$ is secret.

$O(k)$ iterations will suffice to find a suitable n.

Thus, by choosing n and e correctly, the attacker can encode an arbitrary shadow public key of size $3k/2$ in the RSA key escrowed in the FPKC. While this isn't as many bits as was used to set up the RSA public key, it allows one to use a discrete-log based scheme or Rabin's scheme with a *higher* security parameter than the one supported by the government. He can simply choose an arbitrary (P', S') such that $|P'| = 3k/2$, and then generate $(n = pq, e)$ to encode P', publish (n, d) (where $d = e^{-1} \bmod \phi(n)$) and share e with the escrow agency.

3.3 A shadow public-key system based on Rabin

The preceding method of subliminally placing extra bits into the public key of a user can also be applied to the Rabin cryptosystem (where $e = 2$). Although no bits can be subliminally encoded into e, we can still encode $k/2$ bits into n.

Alternatively, there is another method for subliminally embedding P' into P where P is a Rabin public key and P' is a Diffie-Hellman public key. In this attack, the criminal finds x and y so that $q_1 = g^x$ and $q_2 = g^y \bmod p$ are primes. The criminal then sets $n = q_1 q_2$. The shadow secret key is then $P' = n \bmod p$ and the matching shadow secret key is $x + y \bmod p - 1$.

3.4 A shadow public-key system based on Diffie-Hellman, DSS, or El-Gamal

In discrete-log-based FPKCs, a string m can be subliminally encoded in the public key $P = g^x$ by simply trying various choices for x until a specified segment of the bits in g^x matches the desired value of m. For greater computational efficiency, one can choose an initial value of x at random and then choose consecutive values of x until a match is found In this way, g^x may be incrementally computed using a single modular multiplication instead of a more expensive exponentiation.

A more efficient technique exploits the baby-step giant-step method of computing discrete logarithms. In order to embed a short subliminal string of bits $m \in [R+1, \ldots, R+T^2]$ (where R and T are publicly known), the attacker simply sets $S = P' = m$. A sender can recover $P' = m$ from $P = g^m$ using $O(T)$

modular multiplications, by first computing a table $(g^{R+T}, g^{R+2T}, \ldots, g^{R+T^2})$ and then computing $g^m, g^{m+1}, \ldots$ until one computes a pair (i, j) such that $g^{m+i} = g^{jT}$. Note however that in contrast to the previous, less efficient method, S is clearly nonrandom. Also, the computational burden is then shifted to from the attacker to anyone wishing to send the attacker a message. We note that this attack would not be considered a subliminal channel, since the abuse is detectable. The point is that fair cryptosystems never check for abuse, giving us greater leeway in our attacks.

How big can T be? Assuming 512-bit moduli, one might expect to compute $1,000$ modular multiplications a second on the next generation of PC's. Thus, making $T = 2^{20}$ requires less than 20 minutes on a single processor and making $T = 2^{30}$ requires less than two weeks.

In discrete-log case, the number of bits that can be encoded in a reasonable amount of time is much smaller than before. However, we will show in what follows how to boost the number of bits that can be subliminally encoded by spreading a shadow public key across several FPKC public keys.

3.5 Boosting the number of encoded bits

In the examples given above, the attacker was able to subliminally encode various numbers of bits into the public file, depending on the underlying PKC. For example, if one is using Rabin's public-key cryptosystem, the above technique allows one to subliminally encode $k/2$ bits into a public key, and for discrete-log problems we obtain smaller coding rates. By spreading a shadow public key across several FPKC public keys, however, we can overcome this deficiency and allow for shadow keys to have arbitrarily high security. For example, the available subliminal space on two FPKC keys can be combined to form a single shadow public key with a security parameter that is twice as large as before.

The opportunity to have multiple keys is another advertised feature of the Micali FPKC. In particular, in order to support time-bounded wiretapping, Micali advocates the use of myriad keys for each user, with one key for each interval of time. Having many keys escrowed thus allows the criminal to construct a very large shadow key.

There are also other reasons that a user might have several keys listed in the escrow system. For example, current yellow-pages listings allow larger organizations to list many phone numbers, one for each subdepartments, and there is no reason why this flexibility would be abandoned. Alternatively, if we presuppose a society where businesses and other organizations insist on holding the private keys of their employees, then individuals will need a public key for every such organization in which they belong. In each of these scenarios, the space available on the collection of keys for subliminal activities is sufficient to form a very large shadow key.

It is worth noting that for most of our attacks, the "official" public-key generated in our attack is still a useful and reasonably secure (except against the escrow agents) public key. Furthermore, the secret key S' is not needed in order to generate (P, S). This facilitates the use of multiple-key attacks, since someone will much more readily agree to "rent" the subliminal space on their public key if it does not compromise their own security.

Also, we note that attackers can make effective use of fewer bits than are needed for a standard public-key file. For example, in [23, 10], public key cryptosystems with very small public-keys are proposed which may still tie up the computational resources of law enforcement officials.

4 The Failsafe Key Escrow Approach

The flaw in the Micali FPKC is derived from the fact that it is possible for a user to choose a pair of keys (S, P) with the special properties that:

1) the trustees can be provided with valid shares of the secret key S, and

2) the FPKC public key P can be easily converted into a shadow public key P' (using a published algorithm) for a shadow cryptosystem for which the user has also precomputed a shadow secret key S'.

The criminal user can then communicate using the shadow cryptosystem and the shadow pair of keys. The central authority (with the aide of the trustees) can retrieve S but this will not be useful in deciphering traffic encrypted with S'. Moreover, the central authority may have no hope of discovering S'. Unfortunately, it appears that such an attack can be mounted against any escrow system in which the users are given the freedom to select their own keys.

The subliminal key attacks can be avoided by having the trustees themselves select the pair of keys for each user. But schemes in which the trustees select the secret key for each user may leave the user with no assurance that his key has been properly generated (so as to be secure). Such a scheme would not satisfy Property 1. While this might be OK for corporate applications, it may not be acceptable in a democratic society.

It would be desirable to have a method for the selection of key pairs for individuals that protects the privacy and security concerns of law-abiding users as well as the security concerns of the central authority. That is the goal of the Failsafe approach to key escrow. The Failsafe approach is characterized by having the user and government collaborate in the selection of the keys for the user. That way (and only that way), both the user and the government can be assured that the keys have been selected so as to be secure (from their respective points of view).

It is worth noting that it matters a great deal precisely how the government and user interact to select the keys. In particular, if the protocol for the interaction is not designed carefully, then neither side will have the assurances that it might desire. For example, consider the protocol whereby the government chooses 100 possible key pairs for a user to choose from. (Presumably, the government chooses the keys in a random fashion in order to be secure, although the user may not believe this.) The user then picks one of the pairs to become his own.

Although the preceding scheme may seem to be fair enough, it can actually be abused by both sides. For example, the user can be cheated if the government offers the user 100 insecure key pairs to choose from. This is not to suggest that the government would ever do such a thing, but the point is that the user has no assurances that the government has not done such a thing.

The foregoing scheme also has problems from the government's point of view, too. This is because the criminal user can select a public key so as to subliminally embed a few bits of a shadow key. For example, if the criminal is offered 100 randomly-chosen keys, almost surely he will be able to choose one for which the last few bits match a preselected (but short) string m. In this fashion, the user can subliminally embed m in his public key. If the user has many key pairs (as was discussed in Section 3.5), then there will be enough overall space to embed a secure shadow public key.

Several methods have been proposed in the literature for overcoming subliminal attacks. Desmedt [7], in particular, has proposed a general method for defending against subliminal attacks in public-key cryptosystems, and our methods have a number of similarities to his approach. In both cases, the user and the government collaborate to generate a fair key by a "coin-flipping" technique in which one side precommits its half of the final key. However, there are also a number of differences which we briefly list:

1. The Desmedt scenario assumes a trusted center (warden) who can be relied on to make his bits random. In our scenario the user is guaranteed a random key even if all of the governmental agents conspire against him.

2. We consider a key-escrow setting. Thus, as well as agreeing on a public key, the government must have a guarantee that the escrow agents have proper shares of the private key. In Desmedt's protocol, the secret key is completely reserved by the user.

3. Desmedt's solution works in polynomial time, but is not practical. In one step of the protocol, the user and government must engage in a zero-knowledge proof of the form "There exists an R whose encryption is a particular value, and the public-key obtained by using this value of R (and other values in the protocol) is P." To accomplish such a proof, Desmedt uses general protocols for NP which are not practical. In contrast, we more efficiently exploit the algebraic properties of our public-key cryptosystem to yield a practical system which is easily implementable in software.

4. Finally, and most importantly, protection against subliminal channels from the user to the outside world is necessary *but not sufficient* for our security properties to hold. Indeed, some further technical subtleties seem to be required to guarantee that no attack on our system will succeed.

In what follows, we describe one embodiment of the Failsafe Key Escrow approach. This embodiment is based on a Discrete-Log PKC such as Diffie-Hellman or DSS. Here we assume that a prime modulus Q and a generator g for Z_Q^* are publicly known. In this case the public key P that is escrowed for a user is $g^S \bmod Q$, where S is the secret key for the user. The escrow system that will be used in conjunction with the US Digital Signature Standard has this form.

The keys for a user are selected as follows:

Step 1: The trustees and/or the central authority select a random value B from the interval $[0, Q-2]$ and commit to B with the user using an information-theoretically secure commitment protocol. One very simple family of protocols, based on the discrete-log problem is given in [3]. (In fact, depending on the security desired, each trustee i might select and commit to a B_i, with the value of B being formed by taking the XOR of the B_i's. Then only one trustee needs to be trustworthy for the user to be assured of security. For the best theoretical results, it may be useful to also use a chameleon-blob system, which allows for easier simulations in the proofs later. For slightly greater efficiency, a bit-commitment scheme without this simulatability property may be used, but then the formal security properties become more complicated.)

Step 2: The user picks a random secret value A from $[0, Q-2]$ and announces the value of $g^A \bmod Q$ to the trustees and/or the central authority.

Step 3: The user "shares" A with the trustees using a VSS scheme such as that described by Pedersen [14]. (The precise VSS scheme that is used depends on the degree to which the trustees can be trusted to behave properly and the degree to which the users distrust the trustees.) This requires X to send the shares of A to the trustees and it requires the trustees to verify that they received valid shares of A.

Step 4: The trustees and/or the central authority reveal B to the user (who verifies that it is the value previously committed to) and set the public key to be $P = (g^A)g^B \bmod Q$. The value of B is escrowed with the public key for the user. The value of B is not released to the public.

Step 5: The user then sets his secret key to be $S = A + B \bmod (Q-1)$.

In what follows, we show that Properties 1–5 hold for this system. For simplicity, we will argue informally.

Verification of Property 1: Every user who follows the protocol can be sure that he or she has a randomly chosen secret key. This is because the user chooses A at random in $[0, Q-2]$. The authority chooses B, but does so with no knowledge of A. In order to renege on the commitment, the authorities must break the discrete-log problem, in which case they could easily break the whole system anyway. This means that if A was selected at random by the user, then the user can be assured that the distribution on $S = A + B \bmod (Q-1)$ is indistinguishable from the uniform distribution on $[0, Q-2]$. Dishonest authorities can skew the distribution slightly by, for example, trying to guess a discrete logarithm that allows them to break the commitment scheme, which will happen with positive but negligible probability. However, this will not measurably affect the security of the key.

Verification of Property 2: Even a user who fails to select the value of A correctly (e.g., by using a birthday instead of a random number generator) will get a random secret key. This is because the value of B is selected randomly by the authorities and it is revealed to the user after the user commits to the value of A. Hence, the authorities can be assured that $S = A + B \bmod (Q-1)$ is a random integer in $[0, Q-2]$.

Verification of Property 3: Each user can be assured that his or her secret key stays secret unless a sufficient number of trustees release their shares. This is because knowledge of A can be revealed only with the assent of a sufficient number of trustees by the properties of the VSS scheme. Even if B were to be public, this means that $A + B \bmod (Q-1)$ will remain secret unless a sufficient number of trustees cooperate to reveal A.

Verification of Property 4: The central authority is guaranteed to be able to retrieve the secret key of any user provided that a sufficient number of trustees reveal their shares. This is because the properties of the VSS scheme assure that a sufficient number of trustees can collaborate to reveal A. Since B is escrowed, it is then a simple matter to compute $S = A + B \bmod (Q-1)$.

Verification of Property 5: The proof of Property 5 is the most technically complicated. We defer a completely formal treatment to a later draft, and instead give a brief sketch of the technical difficulties involved and how we surmount them.

We first observe that since the bit-commitment scheme is information theoretically secure against the user, the distribution on S will be uniform. One would like to say that this clearly obviates any subliminal attack, since no information may be transmitted, but this argument is fallacious. While our attacks have all been based on transmitting arbitrary bits of information, this is not the only attack possible. For example, suppose that through the use of our protocol, the user became aware of a discrete-log x of some publicly known $y = g^x$. Then regardless of how the public-key is distributed, anyone can use y as an alternate public key for the user. Note however that the "leak" in this case is not to any other user, but to the user himself. Indeed, this example points to a stronger notion of security against subliminal attacks than appears in the literature. The crux of the technical difficulty is that while the escrow agencies (combined) receive only the public-key and the private-key while the user receives the public and private keys, but also knows his own coin tosses, and this extra information could conceivably allow him to read a message that the agencies cannot.

If the combined escrow agents could magically obtain a malicious user $\hat{U}$'s coin tosses, then they would know everything $\hat{U}$ knows. We cannot do this, but instead we can show that for any malicious user $\hat{U}$, the escrow agents can sample from the conditional distribution on $\hat{U}$ view, conditioned on the public-key/private key pair that was selected. It is in this sampling argument that we use the chameleon-blob property of the bit-commitment scheme. We note that they will not reconstruct the actual view obtained by $\hat{U}$, but this does not matter. Using conditional probabilities, we can view the whole key-escrow protocol as follows:

1. A public-key/private-key pair (P, S) is generated according to some distribution that everyone can compute.

2. P is publicized and S is given to the user and the (combined) escrow agents.

3. The user is allowed to sample a view of the protocol, conditioned on (P, S) being computed.

The last step is what could distinguish the user from the combined agents, but by showing that this last step may be also performed by the agents, we intuitively establish that the agents and user are in an exactly symmetrical situation, and hence the agents can read any message the user can read.

This completes the sketch of the proof that Property 5 holds for the FKE protocol. For practical purposes, it probably suffices to dispense with Step 1, and simply have the authorities choose B after seeing g^A. Of course, S would no longer be random (since the authorities could influence the value of P so as to have certain properties), but there is no way known for the authorities to negatively affect the security of $A + B$ given only knowledge of g^A.

Similar protocols can be developed for use with other PKCs such as RSA, but the details become more complicated since the authorities need to interact with the user to choose a "random" number with some special structure. For example, the public keys used with RSA need to be the product of a small number of primes. (If we relax the constraint of having to formally prove that the scheme is secure, then it can suffice for the trustees to multiply the RSA modulus supplied by the user by a random prime, and to add a random number to the RSA exponent.)

The proof method just described can also be extended to show that the FKE system provides security against collections of criminals that band together to produce public keys which can be combined to form a single public key in another cryptosystem.

5 Applications

Failsafe Key Escrow systems can be used in conjunction with any PKC to protect the interests of both law enforcement and the users. FKE may prove to be particularly valuable in the context of the new US Digital Signature Standard (DSS). In particular, it will be important to insure that criminals are not able to use DSS keys for the purposes of encrypting communications in a way that is indecipherable to the Government. This issue is of particular concern in the context of DSS since DSS keys can be easily adapted for encryption. The FKE approach described in Section 4 prevents precisely this sort of abuse.

6 Limitations

It is also worth pointing out the limitations of the Failsafe Key Escrow Approach. Most importantly, the FKE approach does not prevent a pair of criminals from communicating securely using secret information or an alternative escrow system, or from using other protocols for secret key agreement. The main point of the FKE is to prevent criminals from abusing the public keys in the key escrow system. In other words, by designing the key escrow system in a failsafe fashion, the Government can be more assured that the escrow system will not make it any *easier* for criminals to communicate securely.

Our formal proof of Property 5 also requires that the precise name (and other header information) listed in the public file is easily computable given already publicly available information. Otherwise, one can subliminally hide information by declaring one's name to be John "2134fewlr4323423423423...." Doe. Similar restrictions must also be placed on other information available in the public file such as the number of keys for an individual, etc.

7 Acknowledgments

We would like to thank Bonnie Berger, Ron Graham, and John Rompel for their help with this work.

References

[1] T. Beth. Zur diskussion gestellt. *Informatik-Spektrum*, 13(4):204–215, 1990.

[2] G. Blakley. Safeguarding cryptographic keys. *In AFIPS – Conference Proceedings*, 48:313–317, June 1979.

[3] J. F. Boyar, S. A. Kurtz, and M. W. Krentel. A discrete logarithm implementation of perfect zero-knowledge blobs. *Journal of Cryptology*, 2(2):63–76, 1990.

[4] Brassard, G. (1988). *Modern Cryptology; A Tutorial.* Lecture Notes in Computer Science, No. 325 Springer Verlag.

[5] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. *Proceedings of the 26th IEEE Symposium of Foundations of Computer Science*, pages 383–395, 1985.

[6] Denning, D. E. (1982). *Cryptography and Data Security.* Massachusetts: Addison-Wesley.

[7] Y. Desmedt. Abuses in cryptography and how to fight them. *Crypto '88*, pages 375–389, August 1988.

[8] Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. *Crypto '87*, pages 21–39, August 1987.

[9] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. *Proceedings of the 28th IEEE Symposium of Foundations of Computer Science*, pages 427–437, 1987.

[10] G. Harper, A. Menezes and S. Vanstone. Public-key Cryptosystems with Very Small Key Lengths. *Eurocrypt '92*, pages 163–173, May 1992.

[11] T. Leighton. Failsafe key escrow systems. Technical Memo 483, MIT Lab. for Computer Science, August 1994.

[12] T. Leighton and S. Micali. Secret key distribution without public-key cryptography. *Crypto '93*, August 1993.

[13] S. Micali. Fair public-key cryptosystems. Technical Report 579, MIT Lab. for Computer Science, September 1993.

[14] T. P. Pedersen. Distributed provers with applications to undeniable signatures. *Eurocrypt '91*, April 1991.

[15] Schneier, B. (1993). *Applied Cryptography.* John Wiley.

[16] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[17] G. J. Simmons. The prisoners' problem and the subliminal channel. *Crypto '83*, pages 51–67, August 1983.

[18] G. J. Simmons. The subliminal channel and digital signatures. *Eurocrypt '84*, pages 364–378, April 1984.

[19] G. Simmons. A secure subliminal channel (?). Crypto '85, pages 33–41, August 1985.

[20] G. J. Simmons. How to really share a secret. *Crypto '90*, pages 390–448, August 1990.

[21] Simmons, G. (1991). *Contemporary Cryptology.* IEEE Press.

[22] G. J. Simmons. Subliminal communication is easy using the DSA. Eurocrypt '93, pages 218–232, May 1993.

[23] Y. Yacobi. Discrete-log with compressible exponents. *Crypto 90*, pages 639–643, August 1990.