# Fair Cryptosystems, Revisited

A Rigorous Approach to Key-Escrow

(Extended Abstract)

Joe Kilian[1] and Tom Leighton[2]

[1] NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, USA. joe@research.nj.nec.com

[2] Mathematics Department and Laboratory for Computer Science, MIT, Cambridge, MA 02139. ftl@math.mit.edu

**Abstract.** Recently, there has been a surge of interest in key-escrow systems, from the popular press to the highest levels of governmental policy-making. Unfortunately, the field of key-escrow has very little rigorous foundation, leaving open the possibility of a catastrophic security failure. As an example, we demonstrate a critical weakness in Micali's Fair Public Key Cryptosystem (FPKC) protocols. Micali's FKPC protocols have been licensed to the United States Government for use with the Clipper project, and were considered to be a leading contender for software-based key escrow. In the paper, we formally model both the attack and what it means to defend against the attack, and we present an alternative protocol with more desirable security properties.

## 1 Introduction

### 1.1 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 [7, 25, 19, 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 Blakley [2], Shamir [20], Karnin-Greene-Hellman [13] 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, Blakley 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 Blakley 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 [24]).

One difficulty with the secret sharing schemes discovered by Blakley 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 [6], 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 [11] and Pedersen [18] 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 [16], 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 [16], 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."

### 1.2 The results of this paper

**Cryptanalysis of FPKCs** Naively, it might seem that the Micali FPKC cannot be misused by criminals. The government-escrowed keys cannot be used for encryption without the government being able to listen in since the escrow agents can collaborate to reveal the secret key for any user. While the criminals can use alternative means for secure communication, they are not using the *government* key escrow system except, perhaps, for the purposes of authentication, and thus the government escrow system has not been "abused" per se. Moreover, even if criminals use the government key escrow system for authentication purposes during a protocol to exchange other secret keys, they would still have to go through some for of interactive secret key exchange protocol prior to the initiation of secure communications, thus losing the convenience of a noninteractive public-key system.

Unfortunately, this reasoning assumes that the criminals use the same secret keys that were provided to the trustee. However, there is no reason to believe that the criminal will be so cooperative. We in fact describe a very simple way that criminals can exploit the Micali method for Fair PKCs without fear of eavesdropping by the government.

We exploit one of the FPKCs advertised features – the ability of the user to choose his or her public and private keys. Indeed, the defining features of the Micali FPKC are that each user can have the security of selecting his or her own secret key and that the government can be assured that criminal users cannot use the escrowed keys in a manner that is secure against government wiretaps. We demonstrate that it is impossible to achieve both goals at the same time: any escrow system which allows users to select their own keys can be easily abused by criminal users.

**Defining security** We suggest a number of desirable properties for key-escrow systems. Most of these properties are well understood from the extensive work on verifiable secret sharing. However, we know of no "standard" property that implies immunity from the weakness we found in FPKCs. We give a simple example demonstrating that immunity from subliminal attacks is insufficient for our purposes. We give the first formal definition of being *shadow-public-key resistant*, i.e. being secure against untappable messages.

**An alternative key-escrow protocol** In the paper, we also describe an alternative approach to Fair Cryptosystems that is provably immune to such subliminal attacks. The approach, which we refer to as *Failsafe Key Escrow*, is characterized by the use of government-user interaction to select keys. Indeed, an important conclusion of our work is that only by having interaction between users and the government is it possible to attain the security features that are desired by both the users and the government. In particular, our protocol 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, even if all the trustees and central authorities are malicious.

**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 (e.g., by using a birthday or phone number instead of a random number).

**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 assist them in this process.

Whereas the first four properties are well understood, the last property requires a more detailed discussion. Indeed, one section of our paper is devoted to making this property, which we call *shadow-public-key resistance*, well defined. Our main theorem is as follows:

**Theorem:** *Failsafe key-escrow is shadow-public-key resistant.*

We note that achieving this property requires complications to our protocol that would seem to be extraneous without a rigorous standard. This motivates further foundational work in this area.

In comparison, the Fair Public-Key Cryptosystem (FPKC) approach advocated by Micali [16] does not satisfy Properties 2 and 5, and at least one proposed variant of his FPKC does not satisfy Property 3.

**Techniques used** Our attack is based on the subliminal channel attacks developed by Simmons and Desmedt in the 1980s [21, 22, 23, 9, 8, 26]. Using such an attack, a government-sanctioned FPKC can be subverted by criminals or other users 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. The shadow cryptosystem can be set up using only publicly available information, yet we know of no way for the government to prevent its use or to determine who is using it.

In our protocol, we also make important use of information theoretically secure bit commitments, first proposed by Brassard, Chaum and Crépeau [5]. In addition, we require protocols with the *chameleon* property. Informally, the chameleon property says that the recipient of a committed bit is able to open the bit as either a 0 or a 1. This property is necessary for our proof of security to go through. Such schemes are well known, in particular we can use the protocols of [3] based on the hardness of computing discrete logarithms.

**Outline of the paper** The remainder of the paper is partitioned into sections as follows. The flaw in the Micali FPKC is explained in Section 2. We describe the Failsafe approach to key escrow in Section 3. We formalize our attack and show the resistance of Failsafe escrow system to this attack 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.

## 2 The Flaw in the Micali FPKC

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.

### 2.1 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.)

### 2.2 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 $n$.[3] 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 [8], 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, $O(k)$ 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

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

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.

### 2.3 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 $x + y \bmod p - 1$.

### 2.4 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.

### 2.5 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 subdepartment, 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 [27, 12], public key cryptosystems with very small public-keys are proposed which may still tie up the computational resources of law enforcement officials.

## 3 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 central authority or the trustees themselves select the pair of keys for each user. But schemes in which the central authorities 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.

Several methods have been proposed in the literature for overcoming subliminal attacks. Desmedt [8], 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 (first proposed by Blum) in which one side precommits its half of the final key. However, there are also a number of differences: The Desmedt scenario assumes a trusted center (warden) who can be relied on to make his bits random. Whereas we consider a key-escrow setting, in Desmedt's protocol, the secret key is completely reserved by the user. Also Desmedt's solution works in polynomial time, but is not practical. We more efficiently exploit the algebraic properties of our public-key cryptosystem to yield a practical system which is easily implementable in software. 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.

### 3.1 Why abuse-freeness is insufficient

The attack on the previous key-escrow scheme worked by *abusing* the protocol to set up a subliminal channel. The creation of abuse-free protocols [8, 10] is well understood, so it is tempting to simply require that the key-escrow protocols be abuse-free. Unfortunately, there exist abuse-free protocols which are nevertheless vulnerable to this attack. The example we give below is artificial, and would never be reasonably proposed, but illustrates that abuse-freeness is a technically insufficient for our purposes.

Given a canonical public-key cryptosystem for generating secret-key/public-key pairs $(S, P)$, we first construct a new public-key cryptosystem as follows. To generate a pair in the new cryptosystem, one independently generates $(S_1, P_1)$ and $(S_2, P_2)$ in the canonical system. The secret-key/public-key will be $(S_1, P_1P_2)$. To encrypt a message $m$ according to public key $P_1P_2$, the sender simply encrypts $m$ with $P_1$ in the canonical manner, ignoring $P_2$.

Now consider the following key-escrow protocol: A genuinely trusted entity

1. Constructs $(S_1, P_1)$ and $(S_2, P_2)$ in the canonical manner,
2. Publishes $P_1 P_2$ as $U$'s public key,
3. Shares $S_1$ among the escrow agents and
4. Sends $(S_1, S_2)$ to $U$.

Note that for the purpose of this discussion, we assume that the agents have combined their shares, so Step 3 is equivalent to sending $S_1$ to $G$. Also, one can replace the genuinely trusted entity with a secure protocol without affecting our analysis.

The above protocol is clearly abuse-free, since $U$ cannot influence the output in any way. However, there is an obvious shadow public-key system: $O$ can simply encrypt his message according to $P_2$, which the government has no way of knowing. The point is that our ultimate goal is not just to keep information from leaving $U$ but to keep information from being sent (in an untappable manner) to $U$.

### 3.2 An example of failsafe key-escrow.

In what follows, we describe one example of the general Failsafe Key Escrow approach. This example 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 with the chameleon property.[4] 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.

**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 [18]. (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

[4] For slightly greater efficiency, a bit-commitment scheme without this simulatability property may be used, and we know of no major problems with such a protocol, but a formal security analysis becomes problematic.

$A$ 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 Properties 1–4 informally, since they are well understood. In the next section, we consider Property 5 in detail.

**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 $S = A + B \bmod (Q - 1)$.

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 may 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.)

## 4 A formal foundation for security

The flaws in previous attempts at key-escrow highlight the need to put this area on a firmer theoretical foundation. We give a first step in formalizing the subtle security issues that surround key escrow. While the issue of protecting the user's privacy against improper subsets of escrow agents is well understood, protecting the government against shadow public-key attacks is somewhat more complicated. In this section we give present a more formal discussion of this problem.

First, let us state a reasonable objective. We cannot possibly prevent interactive participants from agreeing on a secret-key that is unavailable to the government. Nor can we guard against a user $U$ and and outsider $O$ from having a previously agreed upon convention or secret key that allows $O$ to noninteractively send a message to $U$. What we *can* do is to ensure that $O$ cannot exploit our key-escrow system to noninteractively send a single untappable message to $U$, *unless he could already do so.*

### 4.1 Modeling the participants' knowledge

It is often useful to model the participants' knowledge before and after the protocol. For example, in zero-knowledge proofs the verifier's auxiliary input [17] represents whatever information he may have had before the protocol began. In even the simplest of key-escrow scenarios we must represent many types of knowledge held by different subsets of the participants. For simplicity, we consider the case of a single outside entity user $O$ who wishes to send a bit $b$ to $U$ in a manner untappable by the combined set of escrow agents, denoted $G$.

**Definition 1.** We define a *knowledge ensemble* $\mathcal{K}(k)$ to be a parameterized ensemble on 4-tuples $(K_O, K_U, K_G, K_P)$, where $K_O, K_U, K_G, K_P$ are of expected size polynomial in $k$.

The components of the knowledge vector correspond to information held by the outsider, the user, the government (once the escrow agents have combined their information) and the general public. We consider a parameterized ensemble to, among other things, model the fact that the key escrow protocol is being run with a security parameter $k$.

We don't make any assumptions about the joint distribution of the components of a knowledge ensemble. For example, one legal knowledge ensemble is where $(K_O, K_U, K_G, K_P)$ uniformly distributed over

$$\{(0, 0, \emptyset, \emptyset), (1, 1, \emptyset, \emptyset)\},$$

corresponding to the user and the outsider possessing a single private random bit in common.

Running any sort of protocol causes the knowledge ensemble to evolve. Indeed, if $U$ flips a fair coin this will cause $K_U$ to change. Given a key-escrow protocol $\mathcal{P}$, we define the knowledge ensemble

$$\mathcal{K}^{\mathcal{P}} = (K_O^{\mathcal{P}}, K_U^{\mathcal{P}}, K_G^{\mathcal{P}}, K_P^{\mathcal{P}})$$

by the following procedure for sampling from $\mathcal{K}^{\mathcal{P}}$. Given parameter $k$:

1. Sample $(K_O, K_U, K_G, K_P) \leftarrow \mathcal{K}(k)$
2. $U$, with inputs $k, K_U, K_P$, and $G$, with input $k$, execute protocol $\mathcal{P}$, generating public-key $P$. Denote by $\mathsf{View}_U$ and $\mathsf{View}_G$ the views obtained by $U$ and $G$.
3. Output $(K_O, K_U \mathsf{View}_U, K_G \mathsf{View}_G, K_P P)$.

### 4.2 Modeling the attack

What does it mean for $O$ to send a (single-bit) message to $U$? We model this by a pair of probabilistic polynomial-time procedures $\mathsf{send}(b, K_O, K_P, k) \rightarrow M$ and $\mathsf{receive}(M, K_U, K_P, k) \rightarrow \{0,1\}$. We model the government's attempt to tap this message by a probabilistic polynomial-time procedure $\mathsf{tap}(M, K_G, K_P, k) \rightarrow \{0,1\}$. Given a security parameter $k$ and $b$ chosen uniformly from $\{0,1\}$, we define $(b_U, b_G)$ as the result of the following operations:

1. Choose $(K_O, K_U, K_G, K_P) \leftarrow \mathcal{K}(k)$,
2. Choose $M \leftarrow \mathsf{send}(b, K_O, K_P, k)$,
3. Choose $b_U \leftarrow \mathsf{receive}(M, K_U, K_P, k)$ and
4. Choose $b_G \leftarrow \mathsf{tap}(M, K_G, K_P, k)$.

Using this notation, we define our basic concepts.

**Definition 2.** We say that $(\mathsf{send}, \mathsf{receive})$ is *valid* with respect to $\mathcal{K}$ if for $b \leftarrow \{0,1\}$,

$$Pr(b_U = b) > \frac{1}{2} + \frac{1}{k^c}$$

for some constant $c$ and $k$ sufficiently large.

**Definition 3.** We say that $(\mathsf{send}, \mathsf{receive})$ is *immune* to $\mathsf{tap}$ with respect to $\mathcal{K}$ if for $b \leftarrow \{0,1\}$,

$$|Pr(b_U = b) - Pr(b_G = b) > \frac{1}{k^c}$$

for some constant $c$ and all sufficiently large $k$. Otherwise, we say that $\mathsf{tap}$ is successful against $(\mathsf{send}, \mathsf{receive})$ with respect to $\mathcal{K}$.

**Definition 4.** We say that $(\mathsf{send}, \mathsf{receive})$ is *untappable* with respect to $\mathcal{K}$ if it is *immune* to all probabilistic polynomial-time procedures $\mathsf{tap}$. We say that $\mathcal{K}$ is *vulnerable* if there exist PPT procedures $(\mathsf{send}, \mathsf{receive})$ that are untappable with respect to $\mathcal{K}$.

Less formally, we want the receiver to receive some nonnegligible information about the message-bit $b$ and to learn more than the person tapping the line. Note that untappability implies validity, since the send-receive pair must by immune against the "tapping" procedure that flips a fair coin. We purposefully define a very weak notion of untappability, since we will show that even this is impossible using our scheme.

Finally, we now give the key definition of resistance against shadow public-key attacks.

**Definition 5.** We say that protocol $\mathcal{P}$ is *shadow-public-key resistant* if for all $\mathcal{K}$, $\mathcal{K}^{\mathcal{P}}$ is vulnerable iff $\mathcal{K}$ is vulnerable.

We note that $\mathcal{K}^{\mathcal{P}}$ is certainly vulnerable if $\mathcal{K}$ is. Informally, $\mathcal{P}$ is shadow-key resistant if running it does not open up an opportunity for a covert message where none before existed.

Our main security theorem is as follows:

**Theorem 6.** *Failsafe key-escrow is shadow-public-key resistant.*

*Proof.* **(Sketch)** Let $\mathcal{P}$ denote the failsafe key-escrow protocol. Given a knowledge-ensemble $\mathcal{K}$ such that $\mathcal{K}^{\mathcal{P}}$ is vulnerable, we must prove that $\mathcal{K}$ itself is vulnerable. Let $(\mathsf{send}^{\mathcal{P}}, \mathsf{receive}^{\mathcal{P}})$ be untappable with respect to $\mathcal{K}^{\mathcal{P}}$. We construct a pair $(\mathsf{send}, \mathsf{receive})$ that is untappable with respect to $\mathcal{K}$.

We first make a simplifying approximation on $K_G^{\mathcal{P}}$. Using the properties of the underlying secret-key sharing protocol we have that with probability exponentially (in $k$) close to 1, the secret-key $S$ is computable from $\mathsf{View}_G$. All of the $\mathsf{tap}$ protocols we describe will only make use of $K_G S$ as opposed to $K_G \mathsf{View}_G$. This simplification only makes our theorem stronger. The rare possibility that $S$ is not computable from $\mathsf{View}_G$ will not affect our analysis, and for simplicity will be ignored.

Next, we consider an intermediate "idealized protocol" that generates $(S, P)$ uniformly, publishes $P$ and sends $S$ to $U$ and $G$. In an analogous manner to the definition of $\mathcal{K}^{\mathcal{P}}$, we define $\mathcal{K}^+$ in which $(K_O, K_U, K_G, K_P)$ is transformed into $(K_O, K_U S, K_G S, K_P P)$. Our proof may be broken down into two parts: Showing that $\mathcal{K}^+$ is vulnerable iff $\mathcal{K}$ is vulnerable and showing that $\mathcal{K}^+$ is vulnerable iff $\mathcal{K}^{\mathcal{P}}$ is vulnerable.

**$\mathcal{K}^+$ is vulnerable iff $\mathcal{K}$ is vulnerable** It is easy to show that $\mathcal{K}^+$ is vulnerable if $\mathcal{K}$ is vulnerable (engaging in a protocol only increases ones vulnerability). To show the other direction it suffices to show that if there exists an untappable $(\mathsf{send}^+, \mathsf{receive}^+)$ pair with respect to $\mathcal{K}^+$ then there must be an untappable $(\mathsf{send}, \mathsf{receive})$ pair with respect to $\mathcal{K}$. We define $\mathsf{send}$ and $\mathsf{receive}$ as follows:

$\mathsf{send}(b, K_O, K_P, k)$ /* untappable send procedure */

1. Generate $(S, P)$ according to the correct distribution.
2. Generate $M^+ \leftarrow \mathsf{send}^+(b, K_O, K_P P, k)$.

**3.** Compute $M = M^{+}SP$.

$\mathsf{receive}(M, K_U, K_P, k)$ /* untappable receive procedure */

**1.** Compute $b_U = \mathsf{receive}^{+}(M^{+}, K_U S, K_P P, k)$.

We prove that $(\mathsf{send}, \mathsf{receive})$ is untappable with respect to $\mathcal{K}$ by contradiction. Given a PPT procedure $\mathsf{tap}$ that is successful against $(\mathsf{send}, \mathsf{receive})$ with respect to $\mathcal{K}$, we construct $\mathsf{tap}^{+}$ as follows:

$\mathsf{tap}^{+}(M^{+}, K_G^{+}, K_P^{+}, k)$ /* tapping procedure */

**1.** Compute $K_G, K_P$ and $M$ by the relations $K_G^{+} = K_G S, K_P^{+} = K_P P$ and $M = M^{+}SP$.
**2.** Compute $b_G \leftarrow \mathsf{tap}(M, K_G, K_P, k)$.

It is straightforward to show that for a given value of $b$, the distribution on $b_U$ and $b_G$ will be the same (up to negligible factors) whether computed by $\mathsf{receive}$ and $\mathsf{tap}$ with respect to $\mathcal{K}$ or computed by $\mathsf{receive}^{+}$ and $\mathsf{tap}^{+}$ with respect to $\mathcal{K}^{+}$. Hence, if $\mathsf{tap}$ is successful then $\mathsf{tap}^{+}$ will also be successful.

**$\mathcal{K}^{+}$ is vulnerable iff $\mathcal{K}^{\mathcal{P}}$ is vulnerable** It is easy to show that $\mathcal{K}^{\mathcal{P}}$ is vulnerable if $\mathcal{K}^{+}$ is. The proof of the other direction proceeds along roughly the same lines as the first half of the proof, but must incorporate the views obtained by running our actual key-escrow protocol. The key to our proof is the following simulatability lemma. For any $K_U$, consider the distribution on $(\mathsf{View}_U, S, P)$ generated by $U(K_U, K_P, k)$ and $G$. In the previous argument $\mathsf{View}_U$ was simply $(S, P)$ and hence could be neglected. In the case where we use the actual protocol, we show that $K_U$ can be statistically simulated given $(S, P)$. Intuitively, this means that we can again neglect it in our equivalence proof.

**Lemma 7.** *There exists a PPT algorithm* $\mathsf{simulate}$ *that on input* $(K_U, K_P, S, P, k)$ *and having access to* $U$ *as an oracle, samples from a distribution on* $\mathsf{View}_U$ *that is statistically close to that of* $\mathsf{View}_U$ *conditioned on* $(S, P)$ *being generated by* $U(K_U, K_P, k)$ *and* $G$.

**Remark:** Since $\mathsf{simulate}$ uses $U$ as an oracle, only $U$ has the information necessary to run $\mathsf{simulate}$.

*Proof.* **(Sketch)** Our proof makes heavy use of the properties of information-theoretic chameleon bit commitments. In such bit commitment schemes there is a preliminary phase executed between the prover (who commits to bits) and the verifier. With probability exponentially close to 1, either

- The prover or verifier rejects at the end of this phase or,
- An proper information-theoretically bit commitment scheme is set up for the prover and the verifier. However, a prespecified *extractor* can, after interrogating the verifier as a black box, produce some trapdoor information that will allow one to break the bit commitment scheme by opening a bit commitment as either a 0 or a 1 at will.

For intuition, this means that the verifier knows how to break the bit commitment scheme.

We define simulate as follows:

simulate$(K_U, K_P, S, P, k)$ /* simulates View$_U$ */

1. simulate first runs $U(K_U, K_P, k)$ and $G$, setting up the chameleon bit commitment system. It then runs the extractor and recovers the trap-door information needed to break the bit commitment scheme. It then simulates the commitment by $G$ in Step 1 of the protocol, committing to arbitrary bits. Eliminating $U$'s interaction with the extractor, simulate generates a simulation of $U$'s view of Step 1 of the key-escrow protocol.
2. simulate runs $G$ and $U$ through Steps 2 and 3 of the protocol. Note that $G$ is in fact a collection of trustees that are being simulated by simulate. It uses the trustees to recover $A$, or simulates the trustees rejecting if they would have aborted at this stage.
3. simulate computes $B = S - A$ and simulates the bit commitments from Step 1 being revealed as equal to $B$.

The proof that simulate's correctness is relatively straightforward. We can view View$_U$ in two parts: The view generated by Steps 1–3 of the key-escrow protocol and the view obtained when $B$ is revealed. One fact we need is that these halves of the view are statistically independent, which follows from the statistically-secure nature of the bit commitment and the uniform distribution of $B$. The importance of this fact is that the one can easily simulate the earlier part of the key-escrow protocol, ignoring the possible conditioning caused by knowing that the output if $(S, P)$. More precisely, this conditioning is statistically insignificant and can be ignored.

We show that if there exists an untappable (send$^{\mathcal{P}}$, receive$^{\mathcal{P}}$) pair with respect to $\mathcal{K}^{\mathcal{P}}$ then there must be an untappable (send$^+$, receive$^+$ pair with respect to $\mathcal{K}^+$. We define send$^+$ and receive$^+$ as follows:

send$^+(b, K_O^+, K_P^+)$ /* untappable send procedure */

1. Generate $M \leftarrow$ send$^{\mathcal{P}}(b, K_O^+, K_P^+)$.

receive$^+(M^+, K_U^+, K_P^+, k)$ /* untappable receive procedure */

1. Compute $K_U, S, K_P$ and $P$ from the relations $K_U^+ = K_U S$ and $K_P^+ = K_P P$. Compute $K_P^{\mathcal{P}} = K_P^+$ and $M = M^+$ (for notational consistency).
2. Compute View$_U \leftarrow$ simulate$(K_U, K_P, S, P, k)$. Compute $K_U^{\mathcal{P}} = K_U$View$_U$.
3. Compute $b_U =$ receive$^{\mathcal{P}}(M, K_U^{\mathcal{P}}, K_P^{\mathcal{P}}, k)$.

We prove that send$^+$, receive$^+$) is untappable with respect to $\mathcal{K}^+$ by contradiction. Given a PPT procedure tap$^+$ that is successful with respect to (send$^+$, receive$^+$), we construct tap$^{\mathcal{P}}$ as follows:

tap$^{\mathcal{P}}(M^{\mathcal{P}}, K_G^{\mathcal{P}}, K_P^{\mathcal{P}}, k)$ /* tapping procedure */

1. Compute $M^+ = M^{\mathcal{P}}$, $K_P^+ = K_P^{\mathcal{P}}$ (for notational consistency). Compute $K_G$ and $\mathsf{View}_G$ from $K_G^{\mathcal{P}} = K_G \mathsf{View}_G$ and use $\mathsf{View}_G$ to compute $S$. Compute $K_G^+ = K_G S$.
2. Compute $b_G = \mathsf{tap}^+(M^+, K_G^+, K_P^+, k)$.

As before, a straightforward analysis shows that for a given value of $b$, the distribution on $b_U$ and $b_G$ will be the same (up to negligible factors) whether computed by $\mathsf{receive}^+$ and $\mathsf{tap}^+$ with respect to $\mathcal{K}^+$ or computed by $\mathsf{receive}^{\mathcal{P}}$ and $\mathsf{tap}^{\mathcal{P}}$ with respect to $\mathcal{K}^{\mathcal{P}}$. Hence, if $\mathsf{tap}^+$ is successful then $\mathsf{tap}^{\mathcal{P}}$ will also be successful.

The formalism and 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 3 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, Dorothy Denning, Yvo Desmedt, Ron Graham, John Rompel and Moti Yung 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. G. Brassard, D. Chaum and C. Crépeau. *Minimum Disclosure Proofs of Knowledge.* In *JCSS*, pages 156–189. 1988.
6. 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 on Foundations of Computer Science*, pages 383–395, 1985.
7. Denning, D. E. (1982). *Cryptography and Data Security.* Massachusetts: Addison-Wesley.
8. Y. Desmedt. Abuses in cryptography and how to fight them. *Crypto '88*, pages 375–389, August 1988.
9. Y. Desmedt, C. Goutier, and S. Bengio. Special uses and abuses of the Fiat-Shamir passport protocol. *Crypto '87*, pages 21–39, August 1987.
10. Y. Desmedt and M. Yung. *Minimal Cryptosystems and Defining Subliminal Freeness* In *Symposium on Information Theory*, 1994.
11. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 427–437, 1987.
12. G. Harper, A. Menezes and S. Vanstone. Public-key Cryptosystems with Very Small Key Lengths. *Eurocrypt '92*, pages 163–173, May 1992.
13. Karnin, Greene and Hellman. On secret Sharing Systems, *IEEE Transactions on Information Theory*, vol. 29, 1983.
14. T. Leighton. Failsafe key escrow systems. Technical Memo 483, MIT Lab. for Computer Science, August 1994.
15. T. Leighton and S. Micali. Secret key distribution without public-key cryptography. *Crypto '93*, August 1993.
16. S. Micali. Fair public-key cryptosystems. Technical Report 579, MIT Lab. for Computer Science, September 1993.
17. Oren. On the cheating power of cunning verifiers. *Proceedings of the* $28^{\text{th}}$ *FOCS*, IEEE, 1987.
18. T. P. Pedersen. Distributed provers with applications to undeniable signatures. *Eurocrypt '91*, April 1991.
19. Schneier, B. (1993). *Applied Cryptography.* John Wiley.
20. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
21. G. J. Simmons. The prisoners' problem and the subliminal channel. *Crypto '83*, pages 51–67, August 1983.

22. G. J. Simmons. The subliminal channel and digital signatures. *Eurocrypt '84*, pages 364–378, April 1984.
23. G. Simmons. A secure subliminal channel (?). Crypto '85, pages 33–41, August 1985.
24. G. J. Simmons. How to really share a secret. *Crypto '90*, pages 390–448, August 1990.
25. Simmons, G. (1991). *Contemporary Cryptology.* IEEE Press.
26. G. J. Simmons. Subliminal communication is easy using the DSA. Eurocrypt '93, pages 218–232, May 1993.
27. Y. Yacobi. Discrete-log with compressible exponents. *Crypto 90*, pages 639–643, August 1990.