MIT/LCS/TR-579.b

# FAIR CRYPTOSYSTEMS

Silvio Micali

November 1993

(TR-579.b replaces TR-579)

# Fair Cryptosystems

by

Silvio Micali

Laboratory for Computer Science
Massachusetts Institute of Technology
545 Technology Square, Cambridge, MA 02139

16 November 1993

(An earlier version of this draft was presented at CRYPTO 92)

**Abstract.** There is a growing concern that the wide use of encryption may be more dangerous than helpful to society. In particular, good encryption schemes make court-authorized line-tapping, an effective tool for law enforcement, impossible.

Addressing this concern, we show how to construct cryptosystems in a *fair* way, that is, so as to allow a democratic country to strike the desired balance between the needs of the Government and those of the Citizens. Fair cryptosystems enjoy the following properties: (1) they cannot be misused by criminal organizations and (2) they guarantee to the Citizens exactly the same rights to privacy they currently have under the law.

We actually show how to transform any cryptosystem into a fair one. The transformed systems preserve the security and efficiency of the original ones. Thus one can still use whatever system he believes to be more secure, and enjoy the additional properties of fairness. Moreover, for today's best known cryptosystems, our transformation is particularly efficient and convenient.

Our solution compares favorably with the Clipper Chip, the encryption proposal more recently put forward by the Clinton Administration for solving similar problems. In particular, our solution

(a) allows citizens to choose whatever algorithms they prefer and *all* of their secret keys,

(b) can be efficiently and securely implemented in software, and

(c) ensures that any court-authorized line-tapping ends at the prescribed time.

**Note For The Reader.** Many people would agree that, for crimes such as terrorism or kidnapping, court-authorized line-tapping is well justified. On the other hand, many would also agree that encryption is an excellent way to prevent illegal (or just frivolous) line-tapping. Unfortunately, encryption will also help the perpetrators of hateful crimes to go unpunished, because tapping their communication lines after a legitimate court order will not enable anyone to understand the content of their encrypted conversations. Is this a reasonable price to pay for the privilege of a greater privacy? "To tap or not to tap" (using the title of Dorothy Denning's informative article on cryptography and law enforcement [De]) is a difficult question.

Debating this question is not, however, the goal of this paper. Rather, this paper offers a *technical* solution to the problem of making cryptography compatible with law enforcement. Our goal is offering a new *tool*. Whether such a tool should be used at all, or how, is up to Society. We believe, however, that having options is good.

A second proposal for making encryption compatible with law enforcement has been more recently put forward by the Clinton Administration. The number of our options has thus increased, which is even better. Together with having more options, however, goes the difficult task of identifying the "best one" --or, at least, the one capable of generating the greater consensus.

Because privacy and law enforcement are of interest to most of society, and because we would welcome an informed debate before making crucial policy decisions in this area, we have made a sincere attempt to reach a broad audience. We thus hope that the goals and the properties of our approach will be understandable also by those Government officials and Citizens who do not have any familiarity with cryptography. Indeed, to make our paper easier to read, most of its mathematical details appear in a clearly marked technical appendix.

Further, the basic technical ideas of our approach --which are quite simple to begin with-- are presented at a very intuitive level, so as to be enjoyable for the reader generally familiar with the field of cryptography, though not necessarily expert in secure protocol design. The expert reader will have no difficulty in filling in the formalization and the occasionally subtle details that have been omitted in this draft. (We actually hope to have given her sufficient indications to make her journey through this draft as short as possible.)

## Table of Contents

# 1. Introduction

**A wrong debate**

Court-authorized line tapping is an effective method for bringing criminals to justice. More importantly, in our opinion, it prevents the further spread of crime by deterring the use of ordinary communication networks for unlawful purposes. Thus, there is a legitimate concern that wide-spread use of cryptography may be a big boost for criminal and terrorist organizations. Indeed, many bills propose that a proper governmental agency should be able, under proper circumstances, to obtain the clear text of any communication over a public network. At present, this requirement would translate into coercing citizens into either (1) *using weak cryptosystems* --i.e., cryptosystems that the proper authorities (but also everybody else!) could crack with a moderate effort-- or (2) *surrendering, a priori, their secret key* to the authority. It is not surprising that such alternatives have legitimately alarmed many citizens, generating the feeling that privacy should come before national security and law enforcement.

It is our opinion that this debate is wrong. It is wrong because it is a "one-bit debate," that is, it envisages either unconstrained privacy or no privacy at all. Extreme positions are more likely to be unjust and, indeed, having to choose only between the above alternatives is quite uncomfortable. Fortunately, we are not bound to choose only among what is currently available. It is indeed our goal to change the *status quo* and broaden our options.

**A better alternative**

In this paper we show how to build cryptosystems that are "*fair,*" that is, striking a better balance, in a democratic country, between the needs of society and those of the individual. More precisely, we exhibit cryptosystems that are:

1 *Unabusing* --i.e., the privacy of the law-obeying user *cannot* be compromised-- and

2 *Unabusable* --i.e., unlawful users *will not* enjoy any privacy.

Fair cryptosystems can be easily obtained in the conventional *private-key* model and, more importantly both from a social and a mathematical point of view, also in the more recent *public-key* model. Since we believe that the latter model is the best of the two for a large nation (particularly, for a *democratic* large nation), we focus our effort on the construction of fair public-key cryptosystems. Fair conventional cryptosystems are instead the main focus of the Clipper Chip. (A brief explanation of our preference for the public-key model in this context, and a brief discussion of the Clipper Chip, can be found in Section 9.)

Our construction of fair public-key cryptosystems is actually very general and convenient. We in fact provide a simple methodology for transforming any cryptosystem into a fair one, and our transformation preserves the original security and efficiency of the underlying system. Thus one can "make fair" his favorite cryptosystem without any particular loss.

## 2. Public-Key Cryptosystems

A conventional cryptosystem allows two users X and Y, who have *beforehand* and *securely* agreed on a common secret key K to exchange private messages over a public network. To send Y a private message M, user X computes from M and K a string C (the *ciphertext*) and sends it to Y. Thanks to her knowledge of this same K, Y easily retrieves M (the *cleartext*) from C. The message thus sent is private in the sense that an adversary, though he may easily learn the ciphertext C by tapping the communication line between X and Y, cannot, since he does not know the secret key K, easily compute the cleartext M.

While there is plenty of need for private communication, *securely* agreeing on a common secret key with anyone we wish to talk to in private may not be easy. (For instance, if meeting in a secure physical location for the purpose of agreeing on a common secret key is impractical, sending a secretly chosen key over the telephone is certainly insecure.) In addition, it is not always clear *beforehand* with whom one needs to exchange private messages. (For instance, in most business applications, it is very hard to know *a priori* with whom we will need to conduct private negotiations.)

In response to the common-secret-key-agreement problem, Diffie and Hellman in [DiHe] put forward a new type of cryptosystem, the *public-key cryptosystem* (PKC for short). While in a conventional system each secret key was used both for encrypting and decrypting, in a PKC the encryption and decryption processes are governed by pairs of *matching* keys, which are generated together so to satisfy the following three properties: letting (E,D) be one such pair of matching encryption/decryption keys,

1 Any message can be encrypted using E.

2 Knowledge of D enables one to read any message encrypted with E; on the contrary, without knowing D it is practically impossible to understand messages encrypted with E.

3 Knowing E does not enable one to compute its corresponding decryption key D.

PKCs thus dismiss the need for agreeing beforehand on a common secret key, by using instead a bit of initial interaction and trust. Assume that a user X generates a pair of matching encryption/decryption keys ($E_X$,$D_X$), and that a user Y wants for the first time to send him a private message and tells him so. Then X sends $E_X$ to Y over the phone; Y easily encrypts her message to X with $E_X$ because of Property 1; X easily decrypts it because of Property 2; and, because of Properties 2 and 3, no one else can understand the message so exchanged. In the above protocol, however, Y must trust the authenticity of X's public encryption key $E_X$. (In fact, if an impostor Z substitutes X's public encryption key with his own, Y will end up telling Z her private messages to X.) Moreover, interaction (like in the case of electronic mail) is not always an available commodity. For these reasons, most people prefere to use PKCs in conjunction with what essentially is a "*social agreement*" between users and a *key-management center*. Each user X comes up with a pair of matching encryption and decryption keys ($E_X$,$D_X$). After generating a ($E_X$,$D_X$) pair, the user keeps $D_X$ for himself and gives $E_X$ to the key-management center. The center is responsible (and is trusted!) for *updating* and *publicizing* a directory of *correct* encryption keys, one for each user --i.e., a list of entries of the type $(X, E_X)$ which, for example, may be publicized in a "phone-book format" or via a "411-like service." If, as in the latter example, this distribution occurs over a public network, a digital authentication that $E_X$ comes from the center must be provided, for instance by using one of the existing digital signature schemes. Clearly the users must trust the center, as an untrustworthy center may enable a user Y to read the messages intended for user X by falsely claiming that $E_Y$ is X's encryption key. Thus, in ultimate analysis, the security of a PKC depends on the key-management center. Since setting up such a center on a grand scale requires a great deal of effort by society, the precise protocols the center must follow (and thus its properties) must be properly chosen.

Every advantage has a drawback, and public-key cryptography is no exception. Here a main disadvantage is that any such system can be abused; for example, by terrorists and criminal organizations who can now conduct their illegal business with great secrecy and yet with extreme convenience. It is thus our goal to develop a new technology that allows us to enjoy public-key cryptography while protecting society from the problems arising from its blind utilization.

## 3. Fair Public-Key Cryptosystems

### 3.1 The Basic Notion

Let *S* be a public-key cryptosystem. Informally speaking, and ignoring for the time-being some additional and important issues (see Subsection 3.3), we say that

> *S is a Fair Public-Key Cryptosystem --Fair PKC for short-- if it guarantees a special agreed-upon party (and solely this party!) under the proper circumstances envisaged by the law (and solely under these circumstances!) to understand all messages encrypted using S, even without the users' consent and/or knowledge.*

That is, the philosophy behind a Fair PKC is *improving* the security of the existing communication systems while *keeping the legal procedures* already holding and accepted by the society. In particular, we wish to design Fair PKCs so that the following proposition holds.

> **Proposition**: Let *C* be a ciphertext exchanged by two users in a Fair PKC *S*. Then, under the proper circumstances envisaged by the law, the proper third party will either
>
> 1) find the *cleartext* of *C* relative to *S* (whenever *C* was obtained by encrypting a message according to *S* ) or
>
> 2) obtain a (court-presentable) *proof* that the two users were not using *S* for their secret communication.

Of course, if using any other type of public-key cryptosystem were to be made *illegal*, Fair PKCs would be most effective in guaranteing both private communication to law-obeying citizens and law enforcement. (In fact, if a criminal uses a phone utilizing a Fair PKC to plan a crime, he can still be brought to justice by court-authorized line tapping. If he, instead, illegally uses another cryptosystem, the content of his conversations will never be revealed even after a court authorization for tapping his lines, but, at least, he will be convicted for something else: his use of an unlawful cryptosystem.) Nonetheless, as we shall discuss in section 4, Fair PKCs are quite useful even without such a law.

### 3.2 A General Construction

We shall now present, in a very *general* way, our prefered method for constructing Fair PKCs. We shall see in section 5 that this very general construction can be implemented in practice very efficiently for the best known PKCs.

Below, for concreteness of presentation, we shall use the *Government* for the special agreed-upon party, a *court order* for the circumstances contemplated by the law for monitoring a user's messages, and the *telephone system* for the underlying method of communication. We also assume the existence of a key-distribution center as in an ordinary PKC.

In a Fair PKC there are a fixed number of predesignated *trustees* and an arbitrary number of users. The trustees may be federal judges (as well as different entities, such as the Government, Congress, the Judiciary, a civil rights group, etc.) or computers controlled by them and especially set up for this purpose. Even if efforts have been made to choose *trustworthy* trustees, a Fair PKC does not blindly rely on their being honest. The trustees, together with the individual users and the key-distribution center, play a crucial role in deciding which encryption keys will be publicized in the system. Here is how.

Also for concreteness of exposition, assume that there are 5 trustees. Each user independently chooses his own public and private keys according to a given double-key system. Since the user himself has chosen both keys, he can be sure of their "quality" and of the privacy of his decryption key. He then breaks his private decryption key into five *special* "pieces" (i.e., he computes from his secret decryption key 5 special strings/numbers) possessing the following properties:

1) The private key can be reconstructed given knowledge of all five special pieces;

2) The private key cannot be reconstructed if one only knows (any) 4, or less, special pieces;

3) For $i=1,\ldots,5$, the $i$-th special piece can be *individually* verified to be *correct*.

***Comment***. *Of course, given all 5 special pieces, one can verify that they are correct by checking that they indeed yield the private decryption key. The difficulty and power of property 3 consists of the fact that each special piece can be verified to be correct (i.e., that together with the other 4 special pieces yields the private key) individually; that is, without knowing the secret key at all, and without knowing the value of any of the other special pieces! (How* *these special pieces can be generated is explained in later sections. Below we will show how they can be used.)*

The user then privately (e.g., in encrypted form) gives trustee $i$ his own public key and the $i$-th piece of its associated private key. Each trustee individually inspects his received piece, and, if it is correct, *approves* the public key (e.g., signs it) and safely *stores* the piece relative to it. These approvals are given to the key-management center, either directly by the trustees, or (possibly in a single message) by the individual user who collects them from the trustees. The center, which may or may not coincide with the Government, itself approves (e.g., it itself signs) any public key *approved by all trustees.* These center-approved keys are the public keys of the Fair PKC and they are distributed and used for private communication as in an ordinary PKC.

Since the special pieces of each decryption key are privately given to the trustees, an adversary who taps a user's communication line possesses the same information as in the underlying, ordinary PKC. Thus if this is secure, so is the Fair PKC. Moreover, even if the adversary were one of the trustees himself, or even a cooperating collection of any 4 out of five of the trustees, due to property 2, he would still have the same information as in the underlying ordinary PKC. Since the possibility that an adversary corrupts 5 out of 5 federal judges is remote, the security of the resulting Fair PKC is the same as in the underlying, ordinary one.

When presented with a court order, and only in this case, the trustees will reveal to the Government the pieces of a given decryption key in their possession. This enables the Government to reconstruct the given key. Recall that, by property 3, each trustee has already verified that he was given a correct piece of the decryption key in question. Thus, the Government is *guaranteed* that, *in case of a court order*, it will be given all correct pieces of any given decryption key. By property 1, it follows that the Government will be able to reconstruct any given decryption key if necessary.

### 3.3 Additional Important Issues

We consider the notion of a Fair PKC given so far bo be pretty *basic* because it does not address some additional important issues. In particular, consider the following desiderata.

**Time Bounded Court-Authorized Line-Tapping.**
In general, courts authorize line-tappings for a prescribed amount of time only. But, in our abstract construction, once the Trustees reveal their pieces of the secret key of party X to the Government, the Government's ability to understand X's communications is "turned on" for ever. If we wish, as we do, to improve on the *status quo* but keep our legal procedures, a mechanism must be found to "turn off" this ability at the prescribed time. Not to overload our basic construction of Fair cryptosystems, we postpone solving this important problem until Section 8. By contrast, unfortunately, this important issue is totally ignored by the key-escrow proposal of the Administration.

**Relying on Fewer Shares.**
Another important issue is the ability to rely on fewer shares in order to be able to reconstruct a secret key. So far, in fact, we have been assuming a "all-

or-nothing" approach; that is, a secret key can be reconstructed given all of its special pieces, and cannot be guessed at all given all of its pieces except one. This assume that our Trustee system will always work *perfectly*. Indeed, if a single Trustee, presented with a legitimate court order, does not contribute his own piece of X's secret key --either because he has lost it, or because the computer storing it has failed, or because of sabotage, or because he has been corrupted,-- then it will totally impossible for the Government to tap X's lines. This is a serious problem: any system that assumes too much reliability from people or their computers is *insecure*. Unfortunately, this problem too is ignored by the key-escrow proposal of the Administration. As we show in Subsection 8.2, however, it is possible for Fair cryptosystems to have two different thresholds $\alpha$ and $\beta$, $\alpha < \beta$, such that a secret key is totally impredictable from any $\alpha$ or less pieces, but easily reconstructable from any $\beta$ or more pieces. (Though one might always prefer setting $\beta=\alpha+1$, in practice, as we shall see in the appendix, having b<a may yield simpler algorithms for dealing with fewer shares.)

## 4. Questions and Answers About Fair PKCs

Before addresing the substantive technical questions of how Fair PKCs can be concretly constructed, let us consider some legitimate and broader questions.

Q *Are Fair PKCs less secure?*

A: No. Unless an adversary reads the private messages sent by the user to the 5 trustees (which can be prevented by encryption) or corrupts 5 out of 5 trustees --a rather unlikely event-- they provably provide just the same security as the underlying, ordinary PKC. (Only the Government, and in case of a court order, may have the cooperation of all 5 trustees.)

Q *Are Fair PKCs less efficient?*

A: No. Communication is exactly as efficient as in an ordinary PKC. The only differences are (1) when a public-key is registered, and (2) when a private key is, in a lawful manner, retrieved by the Government. Each user validates his public key only once. Thus only once does he need to give pieces of his private key to the trustees. Moreover, as we have seen in section 4, this step can be implemented by sending 5 short messages, one to each trustee. Second, the lawful reconstruction of a private key by the Government is essentially instantaneous once the five special pieces are obtained from the trustees. Collecting these five pieces electronically is no more cumbersome than issuing or checking a court order as it is needed in a lawful procedure. (As we shall see in Section 6, private-key reconstruction may just consist of receiving 5 short messages and one addition.)

Q *In a totalitarian system, what confidence can we have in a Fair PKC?*

A: Most probably, in a totalitarian system the trustees will be selected with rather different criteria. It is thus conceivable that all of them (whether individuals or organizations) may routinely conspire so as to reconstruct all private keys, destroying all confidence in the privacy of a Fair PKC. On the other hand, believing that ordinary PKCs may be the way to guarantee individual privacy during a dictatorship is quite *naive*. Outlawing any form of PKC will be among the first measures taken by any dictator. Indeed, public use of cryptography is a gift of democracy (and it is important that this gift cannot be turned against it). In fact, Fair PKCs are close in spirit to Democracy itself, in that power is not trusted to any chosen individual (read "trustee") but to a multiplicity of delegated individuals.

Q: *Aren't Fair PKCs the same as ordinary PKCs in which users are obliged to give the Government the private key corresponding to every public key?*

A: No. This deprives the individual of his right to privacy *a priori* and without any just cause. Someone who has not committed (nor is suspected to have committed) a crime should not be required to surrender his right to private communication to anybody, not even to the Government. And this is exactly what he would be obliged to do by revealing his own private key at the time of registering his public one with the key-management authority.

People consent that their right to privacy may be taken away under special circumstances, but do not agree to lose it in an automatic manner. Fair PKCs guarantee the users that they will keep exactly the same rights they currently have in a phone network, and with greater security. (In fact, due to technological advances --i.e., wireless networks-- eavesdropping ordinary phone conversations will become easier and easier for unauthorized parties.)

Q: *What is the difference between a Fair PKC and a PKC with a "hidden trapdoor" chosen by the Government?* [1]

A: There are three main differences:

1) A PKC with a hidden trapdoor is very dangerous: if an enemy finds it, the security of the entire system is compromised.

By contrast, in a Fair PKC, each user chooses his key independently. Thus even if a single user's key is compromised, this does not affect other users at all.

2) Society may never consent to using a PKC with a hidden trapdoor, since this is equivalent to asking the citizen to surrender their right to privacy even before being suspected of any wrong doing! (On the other hand, should a government maliciously ask its citizens to use a special type of PKC concealing the presence of a master secret key, things may get quite unpleasant if the existence of such a key is later discovered!)

3) PKCs with a hidden trapdoor may be weaker than ordinary PKCs, since in the former case the public and private keys must be chosen in a constrained way. In fact, enforcing the existence of a single master secret key for all public keys in the system is a very severe constraint in choosing the individual users' keys. Indeed, it is easy to speak of a

[1] A hidden trapdoor can be thought as a system master secret key.

system with a single master key, but it is also quite conceivable that any such cryptosystem may be easy to break.

By contrast, a Fair PKC, unless all trustees unlawfully collaborate, offers *the same* security of the underlying PKC. Even if 4 out of 5 trustees are compromised, the time that an adversary must invest for understanding anything about a message encrypted in a Fair PKC *provably equals* the time he needs to invest when the same message has been encrypted in the underlying ordinary PKC.

Q *Granted that Fair Cryptosystems protect Society and the individual. But what is their advantage if criminals do not use them for their communications?*

A: We must distingush two settings: First, when the use of any PKC which is not Fair is made illegal. Second, when all commercially available PKCs are Fair (e.g., because thay are the only ones to be standardized), even though non-Fair PKC are not illegal.

Setting 1 has a short answer: a criminal who uses a non-Fair PKC could be brough to justice at least on this charge (recall that Al Capone was convicted for tax evasion).

Let us now consider setting 2. First, note that this is the current setting: anyone in the U.S.A. can use any cryptosystem he or she chooses (though the market for encryption product has not yet reached its full potential). Still, if Society ensures, via standardization, that all *easily available* PKCs are Fair, there are big advantages to be gained.

1) Criminals will have difficulty in distributing their own keys.

In fact, they could not enjoy the convenience of a well-kept and well-publicized public file; that is, they could not call up anyone they want and have a secret conversation with her. They thus would need alternative, cumbersome, and secretive methods to exchange their own keys.

In other words, it is one thing that criminals go out of their way to avoid being controlled by the Society in presence of a court order, and a *very different thing* that the Society goes out of their way to provide criminals with this capability by setting up an ordinary PKC on a grand scale!

2) Besides difficulty in key distribution, criminals will have no convenient access to "alternative" cryptographic *products* which use their keys.

In fact, most products whose usefulness may be greatly enhanced by public-key cryptography --such as "secure" phones, "secure" faxes, etc.-- could become reasonably available, economic, reliable, and compatible, only if *mass produced;* that is, only after intensive engineering effort and big initial investments. Thus, if essentially only the criminals were to use non-fair cryptography, industry would not have sufficient interest in developing products incorporating such technology. (Else, the "criminal market" should have grown so much that we would have nothing more to worry about: civil society as we know it would have already ceased to exist.) Also, big and reputable companies would refrain anyway from manufacturing "questionable" products. Finally, even if a company were willing to manufacture products utilizing non-Fair PKCs, the list of its customers or any record of its sales would be excellent tips for the Police.

3) In an ordinary PKC, the Government is in a difficult position. Since it cannot understand any conversation at all, it has no way to distinguish even potential criminals from non-criminals (setting aside what criminals are saying). In a Fair PKC, instead, the Government can at least make this distinction. Assume that a Fair PKC is standardized, X is one of its users, and a court order authorizes the Government to listen to all messages addressed to X. If the Government is still unable to understand these calls, it means that X really uses a different cryptosystem, and thus intends not to be understood by the Government even in case of a court order. This may be crucial information, and information not available in an ordinary PKC.

4) If all commercially available cryptographic products (e.g., "secure" phones) were based on Fair-PKCs, there would be several advantages. True: a powerful criminal organization could succeed in having designed and produced phones made secure by a non-Fair PKC. This would, however, be less easy for isolated criminals; moreover, it would be most inconvenient for two or three people to get hold of "alternative" products just to discuss their FIRST crime.

5) In any case, punishing *abuse* is secondary with respect to enabling *legitimate use*.

Q: *Fair PKCs may strike a good balance between the needs of the Government and those of the citizens in a democratic country, but: is there any use of Fair PKCs for "less democratic" settings?*

A: Yes. Consider the case of a large organization (say, a private company) where there is a need for privacy, there is an established "superior" (say, a president), but not all employees can be trusted since there are too many of them. The need for privacy requires the use of encryption. Since not all employees can be trusted, using a single encryption key for the whole company is unthinkable. So is using lots of single-key cryptosystems, since this would generate enormous key-distribution problems. Having each employee use his own double-key system is also dangerous, since he might conspire against the company with great secrecy, impunity, and convenience. Obliging every employee to surrender his decryption key to the president is certainly more possible than in the public sector, since a private company need not to be too democratic an organization. But having all employees change their keys every time that a new president is nominated may be quite impractical.

Even in this context Fair PKCs may be of help. Again, key distribution will not be a problem. Each employee will be in charge of choosing his own keys, which makes the system more distributed and agile. While enjoying the advantages of a more distributed procedure, the company will retain an absolute control, since the president is guaranteed to be able to decrypt every employee's communications when necessary. There is no need to change keys when the president does, since the trustees need not to be changed. The trustees' storage devices (keeping their pieces of the employees keys) need not an overwhelming surveillance, since only compromising all of them will give an adversary any advantage.

Finally, Fair PKCs can be used as better secret sharing, since one has the guarantee that the secret will be reconstructed if all pieces (or the majority of them, depending on the implementation) will be made available.

## 5. Making Fair the Diffie-Hellman Scheme

Let us now explicitly exhibit a Fair PKC; actually, let us show how to make Fair the popular Diffie-Hellman PKC. Though this section requires some knowledge of number theory, it illustrates that Fair PKCs can be efficient and algorithmically simple.

Recall that, a bit differently than in other systems, in Diffie-Hellman's scheme each pair of users X and Y succeeds, without any interaction, in agreeing upon a common, secret key $S_{xy}$ to be used as a conventional single-key cryptosystem. Here is how.

*The Ordinary Diffie-Hellman PKC*

There are a *prime p* and a *generator* (or high-order element) $g$ common to all users.
User X *secretly* selects a random integer $Sx$ (for "secret") in the interval $[1,p-1]$ as his private key and publicly announces the integer $Px=g^{Sx}\ mod\ p$ as his public key. Another user, Y, will similarly select $Sy$ as his private key and announce $Py=g^{Sy}\ mod\ p$ as his public key. The value of their common and secret key is determined as $S_{xy}=g^{Sx.Sy}$ mod p. User X computes Sxy by raising Y's public key to his secret key mod p; user Y by raising X's public key to his secret key mod p. In fact

$$(g^{Sx})^{Sy}=g^{Sx.Sy}=Sxy=g^{Sy.Sx}=(g^{Sy})^{Sx}=\ mod\ p.$$

Notice that knowledge of a public key does not easily yield knowledge of its corresponding secret key. In fact, while it is easy, given $g$, $p$, and $a$, to compute $b=g^a\ mod\ p$, no efficient algorithm is known for computing, given $b$ and $p$, the $a$ such that $g^a=b$ mod $p$ when $g$ has high enough order. This is, in fact, the famous *discrete logarithm problem*. This problem has been used as the basis of security in many cryptosystems, and in the recently proposed U.S. standard for digital signatures. Our goal, however, is not to establish the security of the Diffie-Hellman scheme; it is proving that it can be transformed into a fair one. Again, to keep things as simple as possible we imagine that there are 5 trustees and that ALL of them should cooperate to reconstruct a secret key, that is, that ALL shares are needed to reconstruct a secret key. Relaxing this condition involves another idea and will be dealt with in section 5.

*A Fair Diffie-Hellman Scheme*
*(All-Shares Case)*

**Instructions for the users**
Each user X randomly chooses *5* integers $Sx1,...,Sx5$ in the interval $[1,p-1]$ and lets $Sx$ be their sum *mod p*. From here on, it will be understood that all operations are modulo $p$. He then computes the numbers

$$t1=g^{Sx1},...,t5=g^{Sx5} \text{ and } Px=g^{Sx}.$$

$Px$ will be user X's public key and $Sx$ his private key. The $ti$'s will be referred to as the *public pieces* of $Px$, and the $Sxi$'s as its *private pieces*. Notice that the product of the public pieces equals the public key Px. In fact,

$$t1\cdot ... \cdot t5=g^{Sx1}\cdot ... \cdot g^{Sx5}=g^{(Sx1+...+Sx5)}=g^{Sx}.$$

Let T1,...,T5 be the five trustees. User X now gives $Px$ and pieces $t1$ and $Sx1$ to trustee T1, $t2$ and $Sx2$ to T2, and so on. It is important that piece $Sxi$ be privately given to trustee $T_i$.

**Instructions for the trustees**
Upon receiving public and private pieces $ti$ and $Sxi$, trustee Ti verifies whether $g^{Sxi}=ti$. If so, it stores the pair $(Px,Sxi)$, signs the pair $(Px,ti)$, and gives the signed pair to the key-management center. (Or to user X, who will then give all of the signed public pieces at once to the key-management center.)

**Instructions for the key-management center**
Upon receiving all the signed public pieces, $t1...t5$, relative to a given public key $Px$, the center verifies that the product of the public pieces indeed equals $Px$. If so, it approves $Px$ as a public key, and distributes it as in the original scheme (e.g., signs it and gives it to user X.)

This ends the instructions relative to the keys of the Fair PKC. The encryption and decryption instructions for any pair of users X and Y are exactly as in the Diffie and Hellman scheme (i.e., with common, secret key $Sxy$). It should be noticed that, like the ordinary Diffie-Hellman, the Fair Diffie-Hellman scheme does not require any special hardware and is actually easy to implement in software.

**Why does this work?**
First, the privacy of communication offered by the system is the same as in the Diffie and Hellman scheme. In fact, the validation of a public key *does not compromise at all* the corresponding private key. Each trustee Ti receives, as a special piece, the discrete logarithm, $Sxi$, of a *random number*, $ti$. This information is clearly irrelevant for computing the discrete logarithm of $Px$! The same is actually true for any 4 of the trustees taken together, since any four special pieces are independent of the private decryption key $Sx$. Also the key-management center does not possess any information relevant to the private key; that is, the discrete logarithm of $Px$. All it has are the public pieces signed by the trustees. (The public pieces simply are *5* random numbers whose product is $Px$. This type of information is irrelevant for computing the discrete logarithm of $Px$; in fact, anyone could choose four integers at random and set the fifth to be $Px$ divided by the product of the first

four[1]. As for a trustee's signature, this just represents the promise that *someone else* has a secret piece. As a matter of fact, even the information in the hands of the center together with any four of the trustees is irrelevant for computing the private key *Sx*.) Thus, not only is the user guaranteed that the validation procedure will not betray his private key, but he also knows that this procedure has been properly followed because he himself has computed his own keys and the pieces of his private one!

Second, if the key-management center validates the public key *Px*, then the corresponding private key is guaranteed to be reconstructible by the Government in case of a court order. In fact, the center receives all *5* public pieces of *Px*, each signed by the proper trustee. These signatures testify that trustee Ti possesses the discrete logarithm of public piece *ti*. Since the center verifies that the product of the public pieces equals *Px*, it also knows that the sum of the secret pieces in storage with the trustees equals the discrete logarithm of *Px;* that is, user X's private key. Thus the center knows that, if a court order is issued requesting the private key of X, by summing the values received by the trustees, *the Government is guaranteed* to obtain the needed private key.

**Making Fair Other PKCs.**
The reader who wishes to see how *any* PKC can be made Fair can read section A1 of our technical appendix. Like for all general transformations, also this one will be quite inefficient. On the other hand, Section A2 shows that another specific PKC, the popular RSA scheme, can be made Fair in a reasonably efficient manner. This transformation, however, requires much more knowledge of number theory, and does not possess the algorithmic simplicity of our Fair Diffie-Hellman scheme. In particular, we wish to point out that the Diffie-Hellman PKC is very convenient from a law-enforcement point of view in that, if one reconstructs the secret key of a user, he will be able to decrypt both the outgoing and the incoming encrypted messages relative to that user. By contrast, in schemes such the RSA, only the incoming message traffic becomes intelligible once a secret key becomes known. This drawback (from the law-enforcement point of view) can be removed by adding some special hand-shake protocols to the original schemes.

## 6. Variants of the Basic Notion of a Fair PKC

As we have said, several variants of the notion of a Fair PKC are both possible and desirable. Three such variants are presented below in suffiecient detail, while others are only briefly mentioned.

### 6.1 Time-Bounded Court-Authorized Eavesdropping

As mentioned in Subsection 3.3, we now wish to prevent that, having received a court authorization to monitor the communications of a given user for a given interval of time, the agent doing this monitoring (say, the Police) may exceed its mandate and keep on tapping the suspected user's lines for a longer period of time. We discuss various strategies to accomplish this goal.

---

[1] The result would be integral because division is modulo p.

#### 6.1.1 Multiple Public-Keys

A very simple way to ensure time-bounded court-authorized line tapping consists of having each user choose a sufficient amount of matching public and secret keys, say one per month. Each public key will then be publicized specifying the month to which it refers. Someone who wants to send user X a private message in March, will then encrypt it with X's public March key. If this level of granularity is acceptable, the court may then ask the trustees to reveal X's secret keys for a prescribed set of months.

The disadvantage of this approach is that it requires a rather large "total public key," and it may be totally impractical if a fine granularity is desired.

#### 6.1.2 Tamper-Proof Chips

Simple and effective methods to ensure time-bounded court-authorized eavesdropping are possible by means of *secure* chips; these are special chips whose content cannot be "read from the outside," or tampered with in any way without destroying the entire chip with all its protected information. (Such chips are central to the Clipper Chip proposal.)

One such method is the following. Assume that, to monitor the communications of suspected users in response to a court order, the Police use secure chips -- call them the *Polchips*-- possessing an internal and thus untamperable clock as follows. Let there be a court order to tap user X's line from February to April. Then, each trustee will send the Polchip a digitally signed message consisting of his own share of user X's private key (encrypted so that only the Polchip will understand it). The Polchip can now easily compute X's secret key. Thus, if the Court sends to the Polchip a signed message consisting of, say, "decode, X, February-April", since the Polchip has an internal clock(or some other untamperable way to accurately keep track of time), it can easily decrypt all messages relative to X for the prescribed time period. Then, it will destroy X's secret key, and, in order to allow further line tapping, a new court order will be required.

A main advantage of this approach is its simplicity; it does, however, require some additional amount of trust. In fact, the citizens cannot check, but must believe, that each Polchip is manufactured so as to work as specified above.

#### 6.1.3 Algorithmically-Chosen Session Keys

In the multiple public-key method described above, each user selected and properly shared with the Trustees a number of secret keys of a PKC equal to the number of possible transmission "dates" (in the above example, each possible month). Within each specified date, the same public-secret key pair was used for directly encrypting and decrypting any message sent or received by any user. Time-bounded Fair PKCs, however, can be more efficiently achieved by using public keys only to encrypt session keys, and session keys to encrypt real messages (by means of a conventional single-key system). This is, in fact, the most common and efficient way to proceed.

Session keys are usually unique to each pair of users and date of transmission. Indeed, if each minute or second is considered a different date, there may be a different session key for every transmission between two users. In fact, the date may just be any progressive number identifying the transmission, but not necessarily related to physical time.

To achieve time-bounded court-authorized line tapping, we suggest to choose session keys *algorithmically* (so that the Trustees can compute each desired session key from information received when users enter the system), but *unpredictably* (so that, though some session keys may become known --e.g., because of a given court order-- the other session keys remain unknown).

The particular mechanics to exploit this approach is quite important, because not all schemes based on algorithmically selected session keys yield equally convenient time-bounded Fair PKCs.[1] An effective method is described in Section A3 of the technical appendix.

## 6.2 Relying on Fewer Shares

As mentioned in Subsection 3.3, we wish to prevent that, should a single piece of a secret key be missing, a court-authorized line tapping becomes impossible. Better said, we wish that the malicious collaboration of fewer than a prescribed number of Trustees should not enable anyone to compute even a single secret decryption key, while the honest collaboration of more than a given number (possibly different from the previous one) of Trustees should enable one to easily reconstruct a secret key.

To achieve these golas, the solution we present below increases a bit the original number of Trustees (in our case, from 5 to 15), but has the great advantage of being both conceptually and algorithmically very simple. Solutions that do not increase the number of Trustees are discuscussed in Section A4 of our technical appendix.

*THE SHARE REPLICATION METHOD.*
In this solution, each of the 5 trustees is replaced by a group of new trustees. For instance, instead of a single trustee $T_1$, there may be 3 trustees, $T_1{}^1$ $T_2{}^1$ $T_3{}^1$; each of these trustees will receive and check the same share of trustee $T_1$. Thus, it is going to be very unlikely that all 3 trustees will refuse to surrender their copy of the first share. This scheme is a bit "trustee-wasteful" since it requires 15 trustees while it is enough that an adversary corrupts 5 of them to defeat the scheme. (However, one should appreciate that defeating the share-replication scheme is not as easy as corrupting any 5 trustees out of 15, since it must be true that a trustee is corrupted in each group.) The scheme has, nonetheless, two strong advantages: (1) *Scalability*: denoting by n the number of trustee groups, the computational effort of the scheme grows polynomially in n, no matter what the group size is, and thus --if desired-- one can choose a large value for n; (2) *Repetitiveness*: if there are n trustee groups of size k

---

[1] For instance, a time-bounded FAIR PKC that required the Police to contact the Trustees specifying the triplet (X,Y,D) in order to understand X's communication to Y at time D (belonging to the court-authorized time interval), might be deemed inpractical. A better scheme may allow the Police to contact the Trustees only once, specifying only X, Y, and D1 and D2, in order to understand all the communications between X and Y at any date D in the time interval (D1,D2). Since, however, there may be quite many users Y to which the suspected user X talks to, also this scheme may be considered impractical.

each, one should only perform n "operations," in fact, each member of a trustee group gets a "xerox copy" of the same computation.

In the final paper we shall demonstrate that both methods can be optimized, but here let us instead move on to consider a far more important problem than efficiency.

### 6.3 Making Trustees Oblivious

There is another issue worth discussing. Namely, a trustee requested by a court order to surrender his share of a given secret key may alert the owner of that key that his communications are going to be monitored.

A technical solution to this problem is presented in Section A5 of the appendix. (The idea consist of having each Trustee give the shares in his possession to the trusted center from the very beginning, but encrypted. In case of a court order, however, the center has a way of obtaining from the Trustee the decryption of the share of the suspected user without revealing to him which this share is --and thus hiding the identity of the user.)

It should, however, be realized that, while solving a potential problem, this technique may introduce new and different problems. For instance, if the Trustees do not know which user's shares they are revealing, they cannot have a serious chance to consider the evidence against that user, which, presumably, contains the user's name. Thus, the danger exists that the obliviousness of the Trustees may allow one to obtain illegally the secret key of users against which no legitimate court order has been issued.

## 7. Clipper Chip and the Government's Key-Escrow System

### 7.1 A Quick Review of the Government's Proposal.

The Clipper Chip proposal is an attempt to turn conventional cryptosystems into Fair ones. Under the new proposal, users encrypt messages by means of secure chips (as defined in subsection 8.1.2). All these chips contain in their protected boundary the same classified encryption algorithm, *Skipjack*, and each chip possesses a unique identifier. To "initialize" chip x, two Trustees A and B independently choose a secret number (call ax the secret choice of Trustee A and bx that of Trustee B), and remember which secret choice they have made relative to x. These two numbers are then given (somehow) to a chip factory that computes their exclusive-or, cx, and stores it into the protected memory of the chip.[1] This ends the initialization of chip x. Thus after being initialized, each clipper chip possesses a secret key, whose value is at this point only known to the chip itself, though shares of it are stored with the two trustees. Since the chip is assumed to be tamper-proof, it can be handled and sold without any further precautions after being initialized. Assume now that user X has bought chip x, that user Y has bought an analogous chip y, and that the two users have *somehow* exchanged a common secret key Kxy. To privately send a message m to Y, X inputs m to chip x, which

---

[1] Hopefully, this delicate step will be handled in a proper manner; else some party may be enabled to learn or compute cx without any court authorization. The manner in which ax and bx are generated is also crucial.

will then use *Skipjack* to (1) encrypt Kxy with key cx, and (2) encrypt message m with key Kxy, and then send both ciphertexts to Y. In case of a court order for monitoring X's conversations, the two trustees will retrieve their respective secret numbers ax and bx, and reveal them to the Police, which will then combine them so as to compute cx, decode the first ciphertext with cx so as to compute Kxy, and finally decode the second ciphertext with Kxy so as to compute m.

### 7.2 Clipper Chip and Fair PKCs: a Brief Comparison

Both Fair PKCs and Clipper Chip aim at making encryption compatible with law enforcement. Both proposals envisage a group of *trustees* which are collectively, though not individually, trusted. Both proposals advocate that these trustees hold *guaranteed pieces* of the secret decryption key of every user in the system. The main difference between the two systems is the way in which the trustees enter in possession of their guaranteed pieces of a given decryption key. In the Clipper Chip, this process is TOP-DOWN and centralized, in Fair PKCs it is BOTTOM-UP and distributed. Indeed, in the Government proposal, for each user (chip) U, each trustee Ti selects a secret piece SUi, and the sum of these pieces, SU, is made the special secret key of U. (Thus each trustee is guaranteed to have correct pieces of SU.) In a Fair PKC, each user U chooses a public encryption key, PU, and a matching secret decryption key, SU. He then computes a special set of pieces of his own SU, and gives piece i to trustee Ti together with a proof that the piece is correct. (As we shall see, this bottom-up approach gives Fair PKCs greater flexibility.) A second main difference is that Clipper Chip recommends usage of a secret encryption algorithm, Skipjack, contained in the protected chip, while Fair PKCs allow use of any known public-key cryptosystem. In addition, Fair PKCs possess other advantages, some of which can be transfered to the Clipper Chip project. Let us now compare the two proposal on the following series of points.

1 *CONTROL*. Because f the top-down approach of the Government proposal, a Clipper-Chip user does not choose all the keys on which her privacy depends (nor does she choose her own encryption algorithm). By contrast, in a Fair PKC, because of its bottom-up approach, the user chooses all of her keys (and algorithms for that matter). Thus Fair PKC technology is much more respectful of the user, a necessary condition in my opinion for national acceptance of an encryption proposal.

2 *COST*. While Fair PKCs can be implemented either in (ordinary) hardware or software, the Clipper Chip must be implemented with *protected* hardware only, which would drive up the cost of any device using encryption.

3 *DURATION OF LINE TAPPING.* Though courts authorize the tapping of a given communication line only for a prescribed interval of time, once the relevant secret key of a suspected user U has been reconstructed, the Government proposal makes it very easy for the Police to exceed its mandate and keep on tapping the suspected user's lines for a longer period of time. By contrast, the Fair PKC proposal presents various techniques to guarantee time-bounded court-authorized line tapping. Some of these techniques can, however, be extended to work with the Clipper Chip.

4 *RELIABILITY.* In the Government proposal, if the records of a single Trustee are destroyed (e.g., by a fire or earthquake), then court-authorized line-tapping becomes impossible. By using Fair PKCs, instead, one can ensure that, say, 3 out of 5 pieces are enough to reconstruct a secret key (while any 2 or less are provably insufficient). Some of these techniques, however, can be adapted to work with Clipper Chip setting.

5 *COMPATIBILITY.* Assume that a Clipper Chip user U is both a citizen of the U.S.A. and an employee of a large corporation. His using the Clipper Chip guarantees the Government that, in case of a court-order, U's encrypted messages can be decrypted. There may be cases, however, (e.g., industrial espionage) in which also the corporation wishes to monitor U's conversations, but these cases may not coincide with the ones for which courts issue a line-tapping order. In these cases, asking the public trustees to reveal their pieces of U's decryption key to the corporation is unthinkable, since it violates their very mission. The corporation is in trouble and quite powerless. Indeed, this is one of Donn Parker's examples of *"Business Information Anarchy."*

By contrast, Fair PKCs guarantee an easy solution to this problem. Indeed, user U, knowing his own decryption key SU, may *spontaneously* (as long as he wishes to belong to the corporation) and easily break SU *again* into a different set of pieces, and hand them (with a new proof of authenticity) to the trustees of the corporation. Thus either set of trustees can reconstruct U's secret decryption key, each for their own good (but different) reasons. Indeed, it can be seen that Fair PKCs allow an arbitrary "hierarchy of accountability," and solve, in particular, all of the problems rightly raised in Parker's article. It is important to notice that, unless an enemy has all the shares of one set of trustees, having some of the shares of many sets is useless.

6 *SECRET-KEY EXCHANGE.* Two Clipper-Chip users may communicate in private only if they have established a common secret key beforehand. Clipper Chip, in fact, does not provide any help for two users who wish to establish a common secret key. This requires citizens to bear the cost of exchanging private keys at an interpersonal level. Efficiency concerns --or mere common sense-- will induce a market-oriented Government to provide businessmen with an ordinary public-key cryptosystem, **P**, to be used in conjunction with Clipper Chips.[1]
In this scenario, law enforcement is a hard task indeed. Assume, in fact, that the following sentence appears on a national daily paper (or becomes public knowledge somehow):

*"Whenever you (user X) are sent by a user Y a session key Kxy (encrypted with your public encryption key in **P**) and a ciphertext C, if decrypting C with your Clipper Chip and session key Kxy does not yield a meaningful plaintext message, then decrypt C with the DES-EDE algorithm and key Kxy"*

---

1 If the government decided not to provide any PKC, businessmen would most likely develop one at a private level.

Then, *in a single stroke*, all the criminals in the Country who can get hold of equipment running DES-EDE will be able to talk with each other with extreme *secrecy* (since court-authorized line tapping is only guaranteed for messages encrypted with Clipper Chip) and extreme *convenience* (since they have been linked --Society's courtesy-- by an expensive, country-wide, public-key cryptosystem).

In other words, Clipper Chip ensures that criminals cannot abuse of widely-distributed cryptographic products, but allows the abuse of the *public infrastructure* (e.g., the ordinary PKC) that makes effective these products. This is a major mistake. It arises from an old and fatal misconception: PRODUCTS are more valuable than INFRASTRUCTURE. In my opinion, the opposite is true. For very determined criminals, getting hold of "alternative" cryptographic products will be somewhat hard (which makes Clipper Chip a mild instrument of crime-deterrence), but not too hard and not too useful. More alarmingly, ALL potential criminals will benefit to a great extent from a nation-wide directory of public-encryption keys.

By contrast, Fair PKCs retain all the advantages of public-key cryptosystems, but avoid its dangers. Setting up a public file in grand style (e.g., providing a 411-like service that efficiently and reliably delivers tens of millions of public encryption keys) will be very costly to Society, and thus we must ensure that every encryption key contained in it cannot be used so as to avoid court-authorized line tapping. This is exactly what Fair PKCs achieve. And this is, in my opinion, the best way to extend to the field of encryption the proper system of "checks-and-balances" necessary in a democracy.

7 *AMBIGUITY*. Assume that, presented with a court order, the two trustees A and B reveal to the Police their pieces, respectively ax and bx, of the secret key of the chip, x, of a suspected user X. Further assume, however, that after reconstructing this key the Police is still unable to obtain the cleartext of the encrypted messages sent by X. The Police may interpret this as a proof that X does not use his clipper chip to encrypt messages. On the other side, user X may deny all this and say that, clearly, it is the trustees who have revealed wrong pieces to the Police. Who is right?
Fair PKCs are not affected by such ambiguities: the trustees reveal pieces that can be tested to be correct pieces of the secret key matching a public key chosen by the user. Thus, no malicious user can deny he encrypted messages with a different, not approved key. And this is a useful property, whether or not using "alternative" cryptosystems is made illegal.

8 *INTERNATIONAL ISOLATION.* Clipper Chip encrypts messages by means of a classified algorithm, Skipjack, whose circuitry is contained in the tamper-proof portion of Clipper Chip; as a consequence, a foreign business man who wishes to talk to a U.S. Clipper-Chip user must also buy a Clipper Chip, and thus accept that US trustees have pieces of his relevant secret key. This is unlikely to happen. What is more likely, however, is that the rest of the world will develop its own crypto-standards, thus making it easier for them to negotiate contracts, hold video-conferences, etc. at a distance and in private, leaving the U.S. cut off from the benefits that cryptography offers to business.

By contrast, Fair PKCs (1) allow private communication between any two people in the world, and (2) even without cooperation between different governments, guarantee the government of each individual country that each message sent to any of its citizens can be decoded under the circumstances provided by the (local) law.

9. *ANTI-INNOVATION.* The Clipper Chip provides a pre-determined solution to building a Fair cryptosystem. This *per se* is not negative, but mandating or otherwise influencing industry to make "universal" this *single* specific solution would be. Indeed, there would be little room, in the tight bounds of the Clipper Chip proposal, for technology suppliers to design the best algorithms and architectures to meet consumer needs.

   By contrast, Fair PKCs do not impose any particular algorithm, but a simple and clear constraint (verifiability of secret-key shares) that can be satisfied by any algorithm, without any additional penalties, in a pre-processing stage. This leaves still much room for invention and creativity in designing secure communication networks, ultimately leading to much more competitive, well-designed products.

## 8. An Announcement

Very recently, my colleague Tom Leighton and I have found a new type of cryptosystem that provides both Fair private-key cryptography and fair secret-key distribution [LM]. The method does not rely on public-key cryptography nor number theory. It is extremely secure, practical and can be effectively used with Clipper Chip if so wanted.

## 9. Final Remarks

Clipper Chip prevents that a diligently-designed encryption algorithm may fall in the wrong hands, and thus its wide adoption cannot directly harm national security. It may, however, harm both national security and law enforcement indirectly, by causing an unregulated public-key cryptosystem to handle the distribution of secret keys. Indeed, if the Government wants to control crime, the key-distribution infrastructure should be properly regulated. Fair PKCs may provide a technical and democratic way to regulate key distribtion, but first Society must agree on the need of such a regulation. Assuming this to be the case, many other political questions await for answers: Who should the Trustees be? How many should they be? For how long should line-tapping be authorized? Answering these questions well will require a debate as public, wide, and informed as possible.

## Acknowledgments

I wish to thank Allan Borodin, Dennis Branstad, Daniela Caruso Micali, Michael Dertouzos, John Deutch, Oded Goldreich, Shafi Goldwasser, Martin Hellman, Carl Keysen, Tom Leighton, Ron Rivest, and Phillip Rogaway for their valuable comments.

### References

[AwChGoMi] B. Awerbuch, B. Chor, S. Goldwasser and S. Micali. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In *Proceedings of the 26th Annual IEEE Symposium of Foundations of Computer Science.* IEEE, New York, 1986, pp. 383-395.

[Be] J. Benaloh. Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret. Advances in Cryptology --Proceedings of Crypto '86. Springer Verlag, 1986.

[BeGoWi] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Fault-Tolerant Distributed Computing. In *Proceedings of the 20th ACM Symposium of Theory of Computing.* ACM, New York, 1988, pp. 1-10.

[Bl] G. Blakley. Safeguarding Cryptographic Keys. In *AFIPS - Conference Proceedings.* NCC, New Jersey, 1979, Vol. 48 (June), pp. 313-317.

[BlMi] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *Siam Journal on Computing,* 1984, vol. 13 (Novenber), pp. 850-863.
Proceeding Version: FOCS 1982

[ChCrDa] D. Chaum, C. Crepeau, and I. Damgard. Multi-party Unconditionally Secure Protocols. In *Proceedings of the 20th ACM Symposium of Theory of Computing.* ACM, New York, 1988, pp. 11-19.

[De] D. Denning. To Tap or Not To Tap. In *Comm. of the ACM.* March 1993, Vol. 36, No. 3, pp. 25-44.

[DiHe] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Trans. Inform. Theory.* IT-22, 6 (Nov. 1976), IEEE, New York, pp. 644-654.

[Fe87] P. Feldman. A Practical Scheme for Non-Interactive verifiable Secret Sharing. In *Proceedings of the 28th Annual IEEE Symposium of Foundations of Computer Science.* IEEE, New York,1987, pp. 427-438.

[GoMi] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer Systems Science.* Academic Press, New York, Vol. 28 No. 2 (1984), pp. 270-299.

[GMW1] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design. In *Proceedings of the 27th Annual IEEE Symposium of Foundations of Computer Science.* IEEE, New York,1986, pp. 174-187.

[GMW2] O. Goldreich, S. Micali, and A. Wigderson. How To Play ANY Mental Game or A Completeness Theorem for Protocols with Honest Majority. In *Proceedings of the 19th Annual ACM Symposium of Theory of Computing.* ACM, New York, 1987, pp. 218-229.

[LM] T. Leighton and S. Micali. Secret-Key Distribution witout public-key cryptography. Presented at *Crypto 93*, Santa Barbara, CA, August 1993. Improved manuscript available from authors.

[RaBe] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *Proceedings of the 21st ACM Symposium of Theory of Computing.* ACM, New York, 1989, pp. 73-85.

[RSA] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystens. Comm. ACM 21, 2 (Feb. 1978), pp. 120-126.

[Sh] A. Shamir. How to Share a Secret. *Communications of the ACM.* ACM, New York, 1979, Vol. 22, No. 11 (Nov.), pp. 612-613.

# A. Technical Appendix

## A1. How to make Fair any PKC

In this section we want to show how to implement concretely the general (but quite abstract) construction of Section 3.2. Like for all things general, also this transformation is not too efficient.

Disregarding efficiency considerations, this section is devoted to the reader who, being superficially aware of concepts like "secret sharing" or "zero-knowledge," wishes to clarify what is their relationship with Fair PKCs. The reader who has never dealt with the above mentioned concepts may prefer to skip this section entirely; as for the expert in secure protocol theory, she may read just the following sketch.

### A1.1 A Sketch For The Expert.

Cutting corners, each user should (1) come up with a pair of matching public and private keys and give the trustees his chosen public key, (2) encrypt (by a different cryptosystem, even one based on a one-way function) his chosen private key, (3) give the trustees the just computed ciphertext and a zero-knowledge proof that the corresponding "decryption" really consists of the private key corresponding to the given public key, and (4) give the trustees shares of this decryption by means of a Verifiable Secret Sharing protocol that has the property of guaranteeing that the shared secret really is what was encryted in Step 2.

### A1.2 A More Informative Discussion

In expanding the above sketch for the non-expert in protocol design, we feel it is important to illustrate both similarities and differences between Fair PKCs and other related prior notions.

SECRET SHARING
As independently put forward by Shamir [Sh] and Blakley [Bl], secret sharing (with parameters n,T,t) is a cryptographic scheme consisting of two phases: in phase 1, a secret value chosen by a distinguished person, the *dealer*, is put in "safe storage" with n people or computers, the *trustees,* by giving each one of them a piece of information, a *share,* of the secret value. In phase 2, when the trustees pool together the information in their possession, the secret is recovered. In a secret sharing, this storage is *safe* only in two senses:

1 *Redundancy.*
Not all trustees need to reveal their shares in phase 2: it is enough that T of them do. (Thus the system tolerates that some of the trustees "die" or accidentally destroy the shares in their possession)

2 *Privacy.*
If less than t of the trustees accidentally or even intentionally divulge the information in their possession to each other or to an outside party, the secret remains unpredictable until phase 2 occurs.

However, secret sharing suffers of a main problem: *Assumed honesty*; namely,

> Secret sharing presupposes that the dealer gives the trustees correct "shares" (pieces of information) about his secret value. This is so because each trustee cannot verify that he has received a meaningful share of anything. A dishonest dealer may thus give "junk" shares in phase 1, so that, when in phase 2 the trustees pool together the shares in their possession, there is no secret to be reconstructed.

EXAMPLE (Shamir)
The following is a secret sharing scheme with parameters n=2t+1 and T=t+1.

> Let p be a prime >n, and let S belong to the interval [0,p-1]. Choose a polynomial P(x) of degree t by choosing at random each of its coefficients in [0,p-1], except for the last one which is taken to be equal to S , that is, P(0)=S. Then the n shares are so computed: S1=P(1),...,Sn=P(n). *Redundancy* holds since the polynomial P(x) can be interpolated from its value at any t+1 distinct points. (This, in turn, allows the computation of P(0) and thus of the secret.) *Privacy* holds since P(0) is totally undetermined by the value of P at any t distinct points X1 ... Xt different from 0 (in fact, any value v for P(0), together with the value of P at points X1 ... Xt uniquely determines a polynomial).

As it can be easily seen, if the dealer is dishonest, he may give each trustee a random number mod p. If this is the case, then (a) each trustee cannot tell that he has a junk share, and (b) in phase 2 there will be no secret to reconstruct. The consequence of this is that secret sharing is more useful in those occasions in which the dealer is certainly honest, for instance, because being honest is *in his own interest.* (A user that encrypts his own files with a secret key has a big interest in properly secret sharing his key with, say, a group of colleagues: if he accidentally looses it, he needs to reconstruct it!) Secret sharing alone, instead, cannot be too useful for building Fair Cryptosystems: we cannot expect that a criminal give proper shares of his secret key to some federal judges when the only purpose of his doing this is allowing the authorities, under a court order, to understand his communications!

VERIFIABLE SECRET SHARING
A closer connection exists between Fair PKCs and verifiable secret sharing (VSS) protocols. While the two concepts are not identical, a special type of VSS can be used to build Fair PKCs. As put forward by Awerbuch, Chor, Goldwasser, and Micali [CGMA], a verifiable secret sharing (VSS) scheme is a scheme that, while guaranteeing both the redundancy and the privacy property, overcomes the "honesty problem." In fact, in a VSS scheme each trustee can *verify* that the share given to him is genuine *without knowing at all the shares of other trustees or the secret itself.* That is, he can verify that, if T verified shares are revealed in phase 2, the original secret will be reconstructed, no matter what the dealer or dishonest trustees might do.

EXAMPLE (Goldreich, Micali, and Wigderson [GMW1])

> Assume that a PKC is in place and let Ei be the public encryption function of trustee i. Then, as in Shamir's scheme, the dealer selects a random polynomial P of degree t such that P(0)=the secret, and gives

each trustee the n-vector of encryptions E1(P(1)) E2(P(2))...En(P(n)). Trustee i will therefore properly decode P(i), but has no idea about the value of the other shares, and, consequently, whether these shares "define" a unique t-degree polynomial passing through them. The dealer thus proves to each trustee that the following sentence is true "*if you were so lucky to guess all decryption keys, you could easily verify that there exists a unique t-degree polynomial interpolating the encrypted shares.*" Since easily verifying something after a lucky guess corresponds to NP, the above is an "NP sentence." Since, further, the authors show that whole of NP is in zero-knowledge, the dealer proves the correctness of the sentence, in zero knowledge, to every trustee. This guarantees each trustee that he has a legitimate share of the secret, since he has a legitimate share of P, but does not enable him (or him and any other t-1 trustees) to guess what the secret is before phase 2.

VSS AND FAIR PKCs

Assume that each user chooses a secret/public key pair, and then VSS shares his secret key with some federal judges. Does this constitute a Fair PKC? Not necessarily. In a VSS scheme, in fact, the secret may be *unstructured*. That is, each trustee can only verify that he got a genuine share of some secret value, but this value can be "anything." For instance, if the dealer promises that his secret value is a prime number, in an unstructured VSS a trustee can verify that he got a genuine share of some number, but has no assurances that this number is prime.

Unstructured VSS is not enough for Fair PKCs. In fact, the trustees should not stop at verifying that they possess a legitimate share of a "generic" secret number: they should verify that the number they have a share of actually is the decryption key of a given public key! The GMW1 scheme, as described above, is an unstructured VSS, and thus unsuitable for directly building Fair PKCs. The same is true for other VSS schemes (e.g. the ones of Ben-Or, Goldwasser and Wigderson [BeGoWi]; of Chaum, Crepeau and Damgard [ChCrDa}; and of Rabin and Ben-Or [RaBe], just to mention a few).

Some VSS schemes are *structured*, that is each trustee can further verify that the secret value of which he possesses a genuine share satisfies some additional property. What this property is depends on the VSS scheme used. For instance, Feldman proposes a VSS in which, given an RSA modulus N and an RSA ciphertext E(m)= $m^e$ mod N (of some cleartext message m), the trustees can verify that they do possess genuine shares of the decryption of E(m) (i.e., of m). This scheme is attractive in that it is "non-interactive," but *cannot* be used to hand out in a verifiable way shares of the decryption key of a given public key. In fact,

> *the trustees have no guarantee that the decryption of E(m) actually consists of N's factorization.*

In other words, the trustees can verify that they have genuine shares of the decryption (m) of a ciphertext E(m), but m is *unstructured* (with respect to N's factorization and anything else).

CONSTRUCTING FAIR PKCs WITH A GENERIC VSS

Can a generic VSS scheme be transformed so as to yield Fair PKCs? The answer is YES, but at a formidable cost. All of the above mentioned VSS protocols can be "structured" so that the extra property verifiable by the trustees is that the dealer's secret actually is the decryption key of a given public key. In fact, this can be achieved as an instance of *secure function evaluation* between many parties as introduced by Goldreich, Micali, and Wigderson in a second paper [GMW2]. Such secure evaluation protocols are possible, though, more in theory than in practice in light of the complexity of the particular functions involved. In the case of the GMW1 VSS scheme, since the encryption of all the shares is publicly known, the transformation can actually be achieved by a simpler machinery: an additional zero-knowledge proof. But even in this case the computational effort involved is formidable. Essentially, one has to encode the right statement (i.e., the secret, whose proper shares are the decodings of these public ciphertexts, is the decryption key of this given public key) as a VERY BIG graph, 3-colorable if and only if the statement is true, and then prove, in zero-knowledge, that indeed the graph is 3-colorable. Not only are these transformations of a generic VSS to one with the right property computationally expensive, but they require INTERACTION (on top, if any, of the interaction required by the VSS scheme itself)! All these considerations may rule out constructing Fair PKCs this way in practice. Thus CUSTOM-TAILORED methods should be sought, whenever possible, to transform ordinary PKCs to Fair ones. This is our next goal.

## A2. Making Fair the RSA Scheme

In Section A1 we have demonstrated that any PKC can be made Fair by means of an interactive (and not so efficient) protocol for key registration. In Section 5, we have instead seen that the Diffie-Hellman scheme can be made Fair by a much simpler method, one that does not require any "talking back and forth." For completeness sake, given the popularity of the RSA scheme, I feel obliged to show here that it too can be made Fair by a non-interactive protocol. That is, I whish to show that, in order to ensure that each Trustee holds a verified piece of a user-chosen RSA secret key, it is enough that the user send a single message to each Trustee.

I wish to add, however, that our Fair RSA system, though more interesting from a mathematical point of view, does not possess the attractive simplicity of our Fair Diffie-Hellman scheme. Further, the latter scheme enjoys a big advantage from the point of view of law enforcement. Namely, in the Diffie-Hellman cryptosystem, once, in response to a court order, the secret key of a user X has been reconstructed, the Government can easily understand both the messages sent *to* X and those sent *by* X. By contrast, in the RSA scheme, if the secret key of a user X becomes know, only the messages sent to X become easily computable. To allow also the messages sent by X to become intelligible in case of a court order, one must complicate the RSA scheme by requiring a special common-key-agreement protocol prior to any encrypted conversation. An example of such a protocol is given in Subsection 8.1.3, but for now we will be content to show how to share a *standard* secret RSA key with the Trustees. Finally, let me note that our effort would be consirerably simplified if we were willing to make Fair not the standard RSA scheme, but some variant exhibiting its same security.

In the basic RSA PKC, the public key consists of an integer N which is the product of two primes and one exponent e (relatively prime with f(N), where f is Euler's totient function). No matter what the exponent, the private key may always be chosen to be N's factorization. Before we show how to make a Fair PKC out of RSA we need to recall some facts from number theory.

**Fact 1.** Let $Z_N^*$ denote the multiplicative group of the integers between 1 and N which are relatively prime with N. If N is the product of two primes N=pq (or two prime powers: $N=p^a p^b$), then

* a number s in $Z_N^*$ is a square mod N if and only if it has four distinct square-roots mod N: x, -x mod N, y, and -y mod N. (That is, $x^2=y^2=s$ mod N.) Moreover, for i,j ε {1,−1}, from the greatest common divisor of ix+jy and N, one easily computes the factorization of N. Also,

* one in four of the numbers in $Z_N^*$ is a square mod N.

**Fact 2.** On the integers in $Z_N^*$ is defined a function easy to evaluate, the Jacobi symbol, that evaluates to either 1 or -1. The Jacobi symbol of x is denoted by (x/N). The Jacobi symbol is multiplicative; that is, (x/N)(y/N)=(xy/N). If N is the product of two primes N=pq (or two prime powers: $N=p^a p^b$), and p and q are congruent to 3 mod 4, then, letting x, -x, y, and -y mod N be the four square roots of a square mod n, (x/N)=(-x/N)=+1 and (y/N)=(-y/N)=-1. Thus, because of fact 1, if one is given a Jacobi symbol 1 root and a Jacobi symbol -1 root of any square, he can easily factor N.

We are now ready to describe how the RSA cryptosystem can be made fair in a simple way. For simplicity we again assume that we have 5 trustees and that *all* of them must collaborate to reconstruct a secret key, while no 4 of them can even predict it.

*A Fair RSA Scheme*
*(All-Shares Case)*

**Instructions for the user**
A user chooses P and Q primes and congruent to 3 mod 4 as his private key, and N=PQ as his public key. Then he chooses, at random in $Z_N^*$, 5 integers whose Jacobi symbol equals 1, $X_1$ $X_2$ $X_3$ $X_4$ and $X_5$, and computes the values $X_i^2$ mod N for all i=1,...,5. (These values are called the public pieces of N, and the Xi's the private pieces.) Then, he gives each Trustee $T_i$ the modulus N and the private piece $X_i$. Finally, the user computes Z, the product of the 5 public pieces and itself a square mod N; extracts a square root, Y, of Z mod N whose Jacobi symbol is -1; and sends Y to the key-management center.

**Instructions for the trustees**
Trustee Ti stores $X_i$ and N, checks that $X_i$ has Jacobi symbol 1 mod N, and if this is the case, squares $X_i$ mod N and gives the key-management center his signature of $X_i^2$ mod N.

**Instructions for the key-management center**
The center first checks that (-1/N)=1 (thereby checking that for all x: (x/N)=(-x/N), which is partial evidence that N is of the right form). Upon receiving the valid signature of the public pieces of N and the Jacobi -1 value Y from the user, the center checks whether, mod N, the square of Y equals the product of the 5 public pieces. If so, the center is now guaranteed that it has a *split* of N. To make sure that it actually has the *complete factorization* of N, it must now perform the *missing procedure* (i.e., a procedure whose description we

temporarily postpone) to check that N is the product of two prime powers. If this is the case, it *approves* N.

Again, it should be noticed that the Fair RSA scheme can be conveniently implemented in software.

**Why does this work?**
The reasoning behind the scheme is the following. The trustees' signatures of the $X_i^2$'s (mod N) guarantee the center that every trustee Ti has stored a Jacobi symbol 1 root of $X_i^2$ mod N. Thus, in case of a court order, all these Jacobi symbol 1 roots can be retrieved. Their product mod N will also have Jacobi symbol 1, since this function is multiplicative, and will be a root of $X^2$ mod N. But since the center has verified that $Y^2 = X^2$ mod N, one would have two roots X and Y of a common square mod N; moreover, Y is different from X since it has a different Jacobi symbol, and is also different from -x, since (-x/N)=(x/N); in fact: (a) (-1/N) has been checked to be 1 and (b) the Jacobi symbol is multiplicative. Possession of such square roots, by Facts 1 and 2, is equivalent to having the factorization of N, *provided that N is a product of at most two prime powers*. That's why this last property has also been checked by the center before it approved N.

The reason that 4 (or less) trustees cannot factor N with the information in their possession is similar to the one of the discrete log scheme. Namely, the information in their possession solely consists of 4 random squares and their square roots mod N. This cannot be of any help in factoring N, since anybody could randomly choose 4 integers in $Z_N^*$ and square them mod N.

**The missing procedure**
The center can easily verify that N is not prime. It can also easily verify that N is not a prime power by checking that N is not of the form $x^y$, for x and y positive integers, y>1. In fact, for each fixed y one can perform a binary search for x, and there are at most $\log_2(N)$ y's to check, since x must be at least 2 if N>1. It is thus now sufficient to check that N is the product of at most 2 prime powers. Since no efficient algorithm is known for this task when N's factorization is not known, any such check must involve the user who chose N, since he will be the only one to know N's factorization. In the spirit of what we have done so far, we seek a verification method that is (1) *simple*, (2) *non-interactive*, and (3) *provably safe*. The key to this is the older idea of Goldwasser and Micali of counting the number of prime divisors of N by estimating the number of quadratic residues in $Z_N^*$. In fact, if N is the product of no more than two prime powers, at least one number in four is a square mod N, otherwise at most 1 in 8 is. Thus the user can demonstrate that N has at most two different prime divisors by computing and sending to the center a square root mod N for at least, say, 3/16 of the elements of a prescribed list of numbers that are guaranteed to be randomly chosen. This list may be taken to be part of the system. Requiring the user to give the square roots of those numbers in such a random sequence that are squares mod N does not enable the center -- or anybody else for that matter-- to easily factor N. To make this idea viable one would need some additional details. For instance, the trustees may be involved in choosing this public sequence so as to guarantee to all users the randomness of their elements; also the sequence should be quite long, else a user may "shop around" for a number N' that, though product of --say-- 3 prime powers, is such that at least 3/16 of the numbers in the sequence are

squares modulo it; and so on. In "practice" this idea can be put to work quite efficiently by one-way hashing the user's chosen N to a small "random" number H(N), where H is a publicly known one-way hash function, and then generating a sufficiently long sequence of integers S(N) by giving H(m) as a seed to a reasonable pseudo-random number generator. This way, the number sequence may be assumed to be random enough by everybody, since the user cannot really control the seed of the generator. Moreover, the sequence changes with N, and thus a dishonest user cannot shop around for a tricky N as he might when the sequence is chosen before hand. Thus, the sequence chosen may be much shorter than before. If a dishonest user has chosen his N to be the product of three or more prime powers, then it would be foolish for him to hope that roughly 1/4 of the integers in the sequence are squares mod N. The scheme is of course non-interactive, since the user can compute on his own H(N), the number sequence S(N), and the square roots mod N of those elements in S(N) that are quadratic residues, and then sends the center only N and the computed square roots. Given N, the center will compute on its own the same value H(N) and thus the same sequence S(N). Then, without involving the user at all, it will check that, by squaring mod N the received square roots, it obtains a sufficiently high number of elements in S(N).

## A.3 An Effective Method for Time-Bounded Eavesdropping Based on Algorithmically-chosen Session Keys.

**The high-level mechanics of our Suggestion**
In presence of a court order to tap X's lines beween dates D1 and D2, no matter how many dates there may be between D1 and D2, our method allows the Trustees to easily compute and give the Police a small amount of information, i=i(X,D1,D2), that makes it easy to tap X's lines in the specified time interval. The method consists of using a Fair PKC **F** together with a special additional step for selecting session keys for a conventional single-key cryptosystem **C**. In our suggested method, call it the ***(F,C)*** *method*, for any users X and Y, and any date D, there is a session key SXDY for enabling X to send a private message to Y at time D. Each user X is asked to provide the trustees not only with proper shares of his secret key in **F**, but also with *additional pieces of information* that enable them, should they receive a legitimate court order for tapping X between dates D1 and D2, to compute easily i(X,D1,D2) and hand it to the Police.

While the trustees can verify that they possess correct shares of X's secret key in **F**, we do not insist that the same holds for X's session keys. This decreased amount of verifiability is not crucial in this context for the following reasons. Assume that the Police, after receiving i(X,D1,D2) from the Trustees in response to a legitimate court order, are unable to reconstruct a session key of X during the given time interval. Then, this inability proves that X did not originally give the Trustees the proper additional pieces of information about his session keys. It is therefore quite justified that, in response to X's malicious action, the Trustees put together the verified pieces of information in their possession so as to reconstruct X's secret key in **F**. Consequently, from that point on, all messages sent to X via **F**, and in particular via the (**F**,**C**) system, will cease to be private. Moreover, the adoption of a proper "hand-shaking protocol" also ensures that the Police will understand all messages sent by X to any user who replies to him in the (**F**,**C**) system.

In sum, therefore, malicious users who want to hide their conversations from law-enforcement agents even in presence of a court order, cannot do so by taking advantage of the convenience of a nation-wide (**F**,**C**) system. They must go back to the cumbersome practice of exchanging common secret keys before hand, outside any major communication network. It is my firm opinion that the amount of illegal business privately conducted in this cumbersome way is estimated minuscule in comparison to the one that might be conducted via a nation-wide *ordinary* PKC.

**The Specifics Of Our Suggestion**

The hand-shaking protocol of our suggested (**F**,**C**) cryptosystem is the following. When X wants to initiate a secret conversation with Y at date D, she computes a secret session key SXDY and sends it to Y using the Fair PKC **F** (i.e., encrypts it with Y's public key in **F**). User Y then computes his secret session key SYDX and sends it to X after encrypting it with the received secret key SXDY (by means of the agreed-upon conventional cryptosystem **C**). User X then sends SYDX to Y by encrypting it with SXDY. Throughout the session, X sends messages to Y conventionally encrypted with SXDY, and Y sends messages to X via SYDX. (If anyone spots that the other disobeys the protocol the communication is automatically terminated, and an alarm signal may be generated.) Thus in our example, though X and Y will understand each other perfectly, they will not be using a common, conventional key. Notice that, if the Police knows SXDY (respectively, SYDX), it will also know SYDX (respectively, SXDY).

Assume now that the Court authorizes tapping the lines of user X from date D1 to date D2, and that a conversation occurs at a time D in the time interval [D1,D2] between X and Y. The idea is to make SXDY available to the Police in a convenient manner, because knowledge of this quantity will enable the Police to understand X's out-going and in-coming messages, if the hand-shaking has been performed, independently of whether X or Y initiated the call. To make SXDY conveniently available to the Police, we make sure that it is easily computable on input SXD, a master secret key that X uses for computing his own session key at date D with every other user. For instance, SXDY = H(SXD,Y), where H is a one-way function.

Since there may be many dates D in the desired interval, however, we make sure that SXD is easily computable from a short string, i(X,D1,D2), immediately computable by the Police from the information it receives from the Trustees when they are presented with the court order "tap X from D1 to D2." For instance, in a 3-out-of-3 case, if we denote by $i_j(X,D1,D2)$ the information received by the Police from Trustee j in response to the court order, we may set

$$i(X,D1,D2)= H( i_1(X,D1,D2), i_2(X,D1,D2), i_3(X,D1,D2)),$$

where H is a one-way (preferably hashing) function. Now, we must specify one last thing: what should $i_j(X,D1,D2)$ consist of? Letting $X_j$ be the value originally given to Trustee j by user X when she entered the system (i.e., X gives $X_j$ to Trustee j together with the j-th piece of her own secret key in the FAIR PKC **F**), we wish that $i_j(X,D1,D2)$ easily depend on $X_j$. Let us thus describe effective choices for $X_j$, $i_j(X,D1,D2)$, and SXD. Assume that there are $2^d$ possible dates. Imagine a binary tree with $2^d$ leaves, whose nodes have n-bit identifiers --where n=0,...,d. Quantity $i_j(X,D1,D2)$ is computed from $X_j$ by storing a value at each of the nodes of our tree. The value stored at the root, node Ne (where e is the empty word), is $X_j$. Then a *secure* function G is evaluated on input $X_j$ so as to yield two values, $X_j0$ and $X_j1$. The effect of G is that the value $X_j$ is unpredictable given $X_j0$ and $X_j1$. (For instance, $X_j$ is a random k-bit value and G is a secure pseudo-random number generator that, using $X_j$ as a seed, outputs 2k bits: the first k will constitute value $X_j0$, the second k value $X_j1$.) Value $X_j0$ is then stored in the left child of the root (i.e., it is stored in node N0) and value $X_j1$ is stored in the right child of the root (node N1). The values of below nodes in the tree are computed using G and the value stored in their ancestor in a similar way. Let $SX_jD$ be the value stored in leaf D (where D is a n-bit date) and $SXD=H(SX_1D,SX_2D,SX_3D)$. If D1 < D2 are n-bit dates, say that a node N *controls* the interval [D1,D2] if every leaf in the tree that is a descendent of N belongs to [D1,D2], while no proper ancestor of N has this property. Then, if $i_j(X,D1,D2)$ consists of the (ordered) sequence of values stored in the nodes that control [D1,D2], then

I. $i_j(X,D1,D2)$ is quite short (with respect to the interval [D1,D2]), and

II. For each date D in the interval [D1,D2], the value $SX_jD$ stored in leaf D is easily computable from $i_j(X,D1,D2)$, and

III. The value stored at any leaf not belonging to [D1,D2] is not easily predictable from $i_j(X,D1,D2)$.

Thus if each user X chooses her $X_j$ values (sufficiently) randomly and (sufficiently) independently, the scheme has all the desired properties. In particular,

1. user X computes SXD very efficiently for every value of D.

2. When presented with a court order to tap the line of user X between dates D1 and D2, each Trustee j quickly computes $i_j(X,D1,D2)$. (In fact, he does not need to compute all values in the $2^n$-node tree, but only those of the nodes that control [D1,D2].)

3. Having received $i_j(X,D1,D2)$ from every trustee j, the Police can, *very quickly* and *without further interaction with the Trustees,* compute (3.1)$SX_jD$ from $i_j(X,D1,D2)$ for every date D in the specified interval (in fact, its job is even easier since the SXiD's are computed in order and intermediate results can be stored)
(3.2) the master secret-session key SXD from the $SX_jD$'s, and
(3.3) the session key SXDY from SXD from any user Y talking to X in the specified time interval.

Note, however, that no message sent or received before or after the time-interval specified by the court order will be intelligible to the Police (unless a new proper court order is issued).

## A4. Additional Methods for Relying on Fewer Shares

The method described in Section 6.2 relied on fewer shares but increased the number of necessary trustees. Here we want to show, for the case of the Fair

Diffie-Hellman PKC, one can rely on the same number of Trustees, but increase the length of the messages sent by the user to the Trustees. Though this increase is exponential in the number of Trustees, the method is very effective if this number is small.

*THE SUBSET METHOD.*

Each Fair PKC described so far is based on a (properly structured, non-interactive) VSS scheme with parameters n=5, T=5 and t=4. It may be preferable to have different values for our parameters; for instance, n=5, T=3, and t=2. That is, any majority of the trustees can recover a secret key, while no minority of trustees can predict it at all. This is achieved as follows (and it is easily generalized to any desired values of n,T and t in which T>t). We confine ourselves to exemplifying our method in conjunction with the Diffie-Hellman scheme. The same method essentially works for the RSA case as well.

*The Subset Method for the Diffie-Hellman scheme*

After choosing a secret key Sx in [1,p-1], user X computes his public key $Px=g^{Sx}$ mod p. (All computations from now on will be mod p.) User X now considers all triplets of numbers between 1 and 5: (1,2,3), (2,3,4), etc.
For each triplet (a,b,c), he randomly chooses *3* integers *S1abc,...,S3abc* in the interval *[1,p-1]* so that their sum *mod p* equals Sx. Then he computes the 3 numbers

$$t1abc=g^{S1abc},\quad t2abc=g^{S2abc},\quad t3abc=g^{S3abc}$$

The *tiabc*' s will be referred to as *public pieces* of *Px*, and the *Sxiabc*' s as *private pieces*. Again, the product of the public pieces equals the public key Px. In fact,

$$t1abc \cdot t2abc \cdot t3abc = g^{S1abc} \cdot g^{S2abc} \cdot g^{S3abc} = \\ = g^{(S1abc + S2abc + S3abc)} = g^{Sx} = Px$$

User X then gives trustee Ta *t1abc* and *S1abc*, trustee Tb *t2abc* and *S2abc*, and trustee Tc *t3abc* and *S3abc*, always specifying the triplet in question.

Upon receiving these quantities, trustee Ta (all other trustees do something similar) verifies that $t1abc=g^{S1abc}$, signs the value (*Px*,*t1abc*,(a,b,c)) and gives the signature to the key management center.

The key-management center, for each triple (a,b,c), retrieves the values *t1abc t2abc* and *t3abc* from the signed information received from trustees Ta, Tb and Tb. If the product of these three values equals Px and the signatures are valid, it approves Px as a public key.

The reason the scheme works, assuming that at most 2 trustees are bad, is that all secret pieces of a triple are needed for computing (or predicting) a secret key. Thus no secret key in the system can be retrieved by any 2 trustees. On the other hand, when after a court order, at least 3 trustees reveal all the secret pieces in their possession about a given public key, the Government has all the necessary secret pieces for at least one triple, and thus can compute easily the desired secret key.

Finally, we would like to mention that, by making better use of number theory, Ray Sidney has observed that the work of Feldman [Fe87] can be used to rely on fewer shares within the Fair Diffie-Hellman PKC without increasing exponentially the size of the messages. Describing this method does require, however, substantially more number theory. We would also like to mention that, if one allows interaction between the Trustees and the user, then by using a proper VSS protocol in the strategy discussed in section A1, one may transform any PKC into a Fair one relying on any fewer shares and working in polynomial time no matter how many Trustees one wishes to have.

## A5. An Effective Method for Making Trustees Oblivious

The strategy exemplified below assumes that the Trustees can communicate to the center by means of a cryptosystem satisfying a suitable algebraic property: essentially, *random self-reducibility* as introduced by Blum and Micali [BlMi]. This Trustees-center cryptosystem needs not to coincide with the underlying Fair PKC, nor to be itself Fair. To clarify this point, below we assume that the Fair PKC (for which the Trustees need to be made oblivious) is our Fair Diffie-Hellman PKC, while the cryptosystem used by the Trustees to communicate with the center is the standard RSA PKC. For simplicity, we further assume the "all-shares" case for the underlying PKC.

*Oblivious and Fair Diffie-Hellman Scheme*
*(All-Shares Case)*

**The trustees' encryption algorithms**
Since RSA itself possesses a sufficient algebraic property, let us assume that all trustees use *deterministic* RSA for receiving private messages. Thus, let Ni be the public RSA modulus of trustee Ti and ei his encryption exponent (i.e., to send Ti a message m in encrypted form, one would send $m^{ei}$mod Ni.)

**Instructions for user U**
User U prepares his public and secret key, respectively Px and Sx (thus Px = $g^{Sx}$ mod p), as well as his public and secret pieces of the secret key, respectively ti and Sxi's (thus Px= t1· t2· ... ·t5 mod p and ti = $g^{Sxi}$ mod p for all i). Then he gives to the key-management center Px, all of the ti's and the n values Ui=$(Sxi)^{ei}$ mod Ni; that is, he encrypts the i-th share with the public key of trustee Ti.
(Comment: Since the center does not know the factorization of the Ni's this is not useful information to predict Sx, nor can it verify that the decryption of the n ciphertexts are proper shares of Sx. For this, the center will seek the cooperation of the n trustees, but without informing them of the identity of the user.)

**Instructions for the center/trustees**
The center stores the values tj's and Uj's relative to user U and then forwards Ui and ti to trustee Ti. If every trustee Ti responds to have verified that the decryption of Ui is a proper private piece relative to ti, the center approves Px.

**Instructions in case of a court order**

To lawfully reconstruct secret key Sx without leaking to a trustee the identity of the suspected user U, a judge (or another authorized representative) randomly selects a number Ri mod Ni and computes yi = Ri $^{ei}$ mod Ni. Then, he sends trustee Ti the value zi = Ui·yi mod Ni, asking with a court order to compute and send back wi, the *ei*-th root of zi mod Ni. Since zi is a random number mod Ni, no matter what the value of Ui is, trustee Ti cannot guess the identity of the user U in question. Moreover, since zi is the product of Ui and yi mod Ni, the ei-th root of zi is the product mod Ni of the ei-th root of Ui (i.e., Sxi) and the ei-th root of yi (i.e., Ri). Thus, upon receiving wi, the judge divides it by yi mod Ni, thereby computing the desired Sxi. The product of these Sxi's equals the desired Sx.