Translucent Cryptography — An Alternative to Key Escrow, and its Implementation via Fractional Oblivious Transfer*

Mihir Bellare† Ronald L. Rivest‡

February 18, 1996

Abstract

We present an alternative to the controversial "key escrow" techniques for enabling law-enforcement and national security access to encrypted communications.

Our proposal allows such access with probability p for each message, for a parameter p between 0 and 1 to be chosen (say, by Congress) to provide an appropriate balance between concerns for individual privacy, on the one hand, and the need for such access by law-enforcement and national security, on the other. For example, with $p = 0.4$, a law-enforcement agency conducting an authorized wiretap which records 100 encrypted conversations would expect to be able to decrypt (approximately) 40 of these conversations; the agency would not be able to decrypt the remaining 60 conversations at all.

Different values of p can be chosen for different situations, such as for export. Our proposal can be combined with other ideas, such as secret-sharing, to provide additional flexibility. Our scheme is remarkably simple to implement, as it requires no prior escrowing of keys.

We provide an efficient implementation of translucent cryptography. It is based on non-interactive oblivious transfer, as pioneered by Bellare and Micali [2], who showed how to transfer a message with probability 1/2. We provide means for non-interactive fractional oblivious transfer, which allows a message to be transmitted with any given probability p. Our protocol is based on the Diffie-Hellman assumption and uses just one El Gamal encryption (two exponentiations), regardless of the value of the transfer probability p.

This makes the implementation of translucent cryptography competitive, in efficiency of encryption, with current suggestions for software key escrow such as the fair Diffie-Hellman system [20], so that efficiency, at least, is not a barrier to its consideration.

---

* An earlier version of this paper, by the second author, was titled "Translucent Cryptography: An Alternative to Key Escrow," and was presented at the Rump session of Crypto 95. Patent pending with probability 1/2.

† Department of Computer Science & Engineering, Mail Code 0114, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: `mihir@cs.ucsd.edu`

‡ Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: `rivest@theory.lcs.mit.edu`

1 Introduction

Our nation is in the midst of an important and critical debate on cryptographic policy. The current administration seems committed to the idea that the government should be able to read encrypted communications to support law-enforcement or national security objectives, when appropriately authorized. (See NIST [21].) This position is highly unpopular with many (most?) citizens and with much of the business community.

The purpose of this paper is not to contribute to the political debate directly. (For the record, the views of the second author are strongly libertarian.) The reader is referred to Hoffman [17], Denning [10], or Micali [20] for some discussion of the issues involved. Rather, our purpose here is contribute as technologists by pointing out that there are other possible ways we might try to achieve an appropriate balance between individual privacy and government access to communications. Key-escrow is not the only game in town. Just as technology can produce or exacerbate a basic conflict, technology can also provide means for its solution.

1.1 Translucent cryptography

This paper introduces a new dimension along which debate can be framed and compromise can be considered: the probability p with which a particular message can be decrypted by the government. A fraction p of the messages sent from a user Alice to a user Bob will be decryptable by the government, and the remaining $1 - p$ fraction will not be decryptable by the government. (To recover messages from one user to another, the government would wiretap the communication between them. Of the recovered messages, it will be able to decrypt a p fraction.) Of course, the intended recipient of an encrypted message can always decrypt it; it is only the government that gets a "partial view." The sender of an encrypted message does not know whether or not that message will be decryptable by the government. A small value of p (say, $p = 0.02$) favors a libertarian viewpoint, while a large value of p (say, $p = 0.9$) favors law-enforcement.

In comparison, we see that debate about key-escrow is a difficult one because there is no "middle ground": either the government has access (if the keys are escrowed) or it does not (if the keys are not escrowed). With our proposal, values of p strictly between 0 and 1 form a "middle ground" where each side of the debate has some gain, and some loss. A value of p can be chosen that balances the relative concerns. Congress might pick the appropriate p.

The scheme is called "translucent" because it explores the space between "opaque" (strong encryption with no key escrow) and "transparent" (no encryption or encryption with key escrow).[1] With our translucent scheme, the government can decrypt *some* of the messages, but not all. Just as a translucent door on a shower stall provides some privacy, but not perfect privacy, translucent crypto provides some communications privacy, but not perfect privacy. In our scheme the degree of "translucency" can be controlled by varying p.

The value of p does not even need to be fixed once and for all, nor need it be the same for each kind of encryption equipment. The value of p might be chosen small today (say $p = 0.02$), and increased or reduced later as judged appropriate. Or, one could have one value of p for cellular phones and a different one for email encryption programs. Or, a larger value of p could be used in export versions of programs than is used for domestic versions. The value of p used is built into the encrypting device or program. It is possible for the government to measure the effective value of p used by an encrypting device or program, and so to monitor compliance with the overall scheme.

[1] Other adjectives we considered instead of "translucent" were "variable-opacity," "fractional-access," "partial-access," and "probabilistic-access." Translucent seemed the simplest choice.

Because a criminal does not know which messages are decryptable and which are not, he runs the risk every time he uses encryption that this particular message will be decrypted and will be used against him.

Our proposal also has the advantage, compared to key-escrow techniques, that there is practically no "set-up" required. Users and manufacturers do not need to register or escrow their cryptographic key information. More specifically, a manufacturer of cryptographic circuits does not have to secretly manufacture, record, and deliver to escrow agents the secret keys of each chip, as is the case for the "Clipper chip." Indeed, the chips can be all made identically in a non-secret manner. Analogously, there is no need for users of public-key cryptosystems to submit their private keys for escrowing; their private keys remain forever their own secrets. These are consequences of the fact that our scheme discloses only the message or session key to the government, not the long-term keys of the devices or of the users. The only set-up required is for the government to publish a list of public keys, and for Congress to pick an appropriate value(s) for p.

This proposal can combined with previously-known techniques to achieve other objectives, such as requiring more than one government agency to cooperate before any messages can be decrypted, or limiting the effective time period of a wire-tap warrant.

This proposal is hardly perfect. One can object to it on many fronts, both political (the "other side" of the debate gets to win a little, and could win more later if p changes) and technical (like key escrow, our scheme is easy to subvert by techniques such as double-encryption).

Nonetheless, this proposal will serve its purpose if it opens our imaginations a bit, enlarges our sense of the possible, and helps to bring a difficult national debate closer to a resolution we can all live with.

1.2 Fractional oblivious transfer

We suggest an implementation of translucent cryptography based on an implementation of *non-interactive fractional oblivious transfer*. The resulting translucent scheme is as efficient, in terms of encryption, as current suggestions for software key escrow. Specifically, we need one El Gamal encryption (two exponentiations), which is the same as the cost of encryption in the Diffie-Hellman system.

Our implementation is based on the non-interactive oblivious transfer primitive of [2]. We extend their scheme, which achieves transfer probability 1/2, to achieve transfer probability any fraction $p \in [0, 1]$, at no added encryption cost.

In any suggestion for technical solutions to the policy debate we have been discussing, efficiency is a key issue. Although a scheme cannot, of course, stand on efficiency alone, it can certainly fail due to its inefficiency. By providing an implementation of translucent cryptography which is competitive, in encryption efficiency, with implementations of key escrow, we have surmounted at least the first barrier to its discussion.

Furthermore, we suggest that our implementations of fractional oblivious transfer, described in Section 4, may be of independent interest.

We stress that our implementation of translucent cryptography based on non-interactive oblivious transfer will not incur any "extra flows." When Alice wishes to communicate with Bob, her only transmission is to Bob. (In particular, she doesn't communicate on-line with the government.) If the government wants to know something about what Alice is saying to Bob, it must wiretap their communications, and then it will be able to decrypt a fraction p of the messages it picks up.

2 Non-interactive oblivious transfer

Since our proposal uses non-interactive oblivious transfer techniques, we give the background for and sketch this technology in this section.

Rabin [24] was the first to introduce the notion of *oblivious transfer*, in which one party (Alice) can transfer a message to another party (Larry[2]) in such a way that:

- Larry receives the message with probability exactly 1/2.
- Alice does not know whether Larry received the message or not—that is, she is *oblivious* as to whether the transfer was successful or not.

Rabin introduced the notion of oblivious transfer to help solve the problem of "exchanging secrets," a problem also studied by Blum [4].

Protocols for oblivious transfer have been studied by Even, Goldreich, and Lempel [11, 12], Fischer, Micali, and Rackoff [14], Berger, Peralta, and Tedrick [3], Crépeau [6], and others [19, 9, 16, 1]. These protocols are *interactive*: they require the recipient, Larry, to actively participate in the protocol by sending messages to Alice. For our purposes, we need the oblivious transfer to be *non-interactive*: Larry should not have to send any messages in order to receive Alice's message with probability one-half. With non-interactive oblivious transfer, Larry needs merely to receive (or overhear) Alice's message in order to decrypt it with probability one-half.

The first non-interactive oblivious transfer protocol is due to Bellare and Micali [2]. Further protocols were given by De Santis and Persiano [8] and De Santis, Di Crescenzo and Persiano [7].

To make this paper concrete and self-contained, we describe the simplest proposal made by Bellare and Micali for implementing non-interactive oblivious transfer. Our proposal does not depend on the details of how non-interactive oblivious transfer is implemented, however, so that other implementation techniques may be used. The rest of this section may be skipped by those not familiar with number theory or those not wishing to get involved in the mathematical details.

An initial *global set-up phase* establishes the following three public values:

- a large global prime q (say at least 1024 bits in length),
- a generator g of the multiplicative group Z_q^*, and
- a value U such that no one knows the discrete logarithm of U (base g, modulo q). More precisely, computing U's discrete logarithm should be computationally infeasible for anyone.

We denote the global prime as q, since we are already using p to stand for something else. Bellare and Micali suggest ways that values for q, g, and U could be chosen. In our application, perhaps the ACLU could choose these values.

The second phase is *publication of public keys*. Like the global set-up phase, this phase needs to be done only once, no matter how many oblivious transfers will be performed. Larry publishes a pair of values (V, V'), where $V' = VU$, as his public key pair. Larry should know either the discrete logarithm of V, or the discrete logarithm of V'; he cannot know both. We say that V is a *good key* (for Larry) if Larry knows the discrete logarithm of V, otherwise we say that V is a *bad key* (for Larry).

[2]We explain the cast of characters: Alice and Bob are citizens, who may or may not be up to something. Larry works for a law-enforcement agency.

Can Larry cheat by publishing two public keys V and V' that are both good for him? If it is indeed the case that computing the discrete logarithm of U is computationally infeasible, then Larry can not successfully cheat, since anyone can check that $V' = UV$, and thus if both V and V' are good for Larry, Larry could easily compute the discrete logarithm of U:

$$\log(U) = \log(V') - \log(V) \pmod{q-1} .$$

Thus only one of the two public keys Larry published is good for Larry; he knows the discrete logarithm of only one of these keys.

In the final *communication phase*, we suppose now that Alice wishes to obliviously send Larry a message $s \in Z_q^*$. (We use s to denote the message, since later s will denote a session key in Alice's conversation with Bob.) Alice can do so by picking one of Larry's two public keys at random, and encrypting s using that public key and the ElGamal encryption algorithm [15], as follows (supposing that V was picked):

- Alice picks a value y from $\{0, 1, \ldots, q-2\}$ uniformly at random, and sends Larry the ciphertext

  $$E(s, V) = (c_1, c_2) = (g^y, sV^y) .$$

  (All values computed modulo q.)

- If (and only if) Larry knows the discrete logarithm x of V, he can compute s:

  $$s = c_2/c_1^x \pmod{q} .$$

Thus, Larry receives s with probability exactly $1/2$, since only one of his two public keys is good. The protocol is oblivious since Alice doesn't know which of Larry's keys is good.

Note that this protocol is non-interactive. Also note encryption takes two exponentiations. This is the same as in the Diffie-Hellman public key system. (There, Bob would have public key $V = g^x$ and private key x, and Alice would send him a message s by sending $E(s, V)$.)

The above protocol differs in presentation and inessential minor respects from that proposed by Bellare and Micali; see their paper [2] for other methods and discussion.

It is important to note that successive oblivious transfers are not independent: if Alice sends two successive messages using Larry's public key V, Larry either receives them both or receives neither of them. This property of non-interactive OT has often been pointed out in the literature, and has relevance to our application, as discussed later.

3 Translucent Cryptography

In the previous section we have explained how non-interactive oblivious transfer can be achieved, where the probability is $p = 1/2$ that Larry receives the message. In the next section, section 4, we explain how to achieve non-interactive *fractional* oblivous transfer can be acheived, where a wide range probabities p can be implemented. Before diving into the mathematics required to implement non-interactive fractional oblivous transfer, however, we explain in this section how non-interactive fractional oblivious transfer can be used to implement translucent cryptography. This is rather straightforward. The reader should, for the moment, accept our promise that we will explain how to implement non-interactive fractional oblivious transfer with a variety of values for p; this promise will be kept in section 4.

Assume that a probability p has been determined, and that the global quantities needed to set-up the non-interactive fractional oblivious transfer have been determined.[3] We also assume that Larry (the government) has published his public key(s), again according to the algorithm specified by whichever scheme we are using. Thus he can obtain a message sent via oblivious transfer with probability p.

The above computation and publication by Larry is the *only* set-up required by our translucent cryptography scheme; there is no need for each user to escrow shares of his private key, or for manufacturers to escrow shares of keys stored in each cryptographic device produced. Each cryptographic product can be made in an identical manner, embodying the quantities just described. In practice, each product would also presumably have a unique identifying serial number, so that its messages can be distinguished from those of other products. This number does not need to be secret.

How can a user Alice now send a message M in encrypted form to another user Bob, in such a way that Larry (who is authorized to eavesdrop on the message) can decrypt it with probability p?

First, Alice determines a "message key" (or "session key") key s in an arbitrary manner. The key s might, for example, be freshly generated, or might be the result of a prior agreement between Alice and Bob. Then, Alice computes, as a function of Larry's key, a string L which comprises the message she would send to transfer s to Larry under the p-NFOT scheme in use. (For example, if we are using the polynomial scheme of section 4 and Larry's public key is $(V_1, \ldots, V_m, W_0, \ldots, W_a)$ then Alice picks $i \in [m]$ at random and lets $L = E(s, V_i)$.) Now, Alice transmits a message to Bob consisting of the following fields:

(F1) The encryption of message M using a standard algorithm (e.g. DES) and the message key s.

(F2) Information, if necessary, that allows Bob to determine what secret message key s is being used.

(F3) The string L she computed above.

The third field, namely L, is the "LEAF" (Law Enforcement Access Field). With probability p this information allows Larry to determine the message key s, and thus to decrypt the first field to obtain the message m.

The second field would typically consist of the encryption of s under Bob's public key, as is done for example in Privacy Enhanced Mail [18]. Bob can reliably decrypt this field to obtain s, and thus to decrypt the first field to obtain the message m. In a variation, the session key s would be encrypted in a DES key known only to Alice and Bob. Or, this information might consist of a message that can be used in a Diffie-Hellman key-agreement protocol to establish s. There are a variety of methods by which Alice can let Bob know what s is, any of which can be used in our scheme.

The message might also contain the identifying serial number of Alice's cryptographic product. This could be in the clear, or be part of the information transferred obliviously to Larry in the third field.

To clarify this transmission, we stress that L is not sent to Larry; it is sent to Bob. Larry obtains it only if he wiretaps the line between Alice and Bob. There is no direct communication from Alice to Larry at any time.

We note that Bob can verify that Alice is following the translucent cryptography protocol properly, by checking that the LEAF is properly constructed. In this way a correct implementation

[3] For the schemes described in section 4, these quantities are denoted q, g, and U if one is using the binary scheme, or G, g and U if one is using the polynomial scheme, where no one can feasibly compute the discrete logarithm of U to the base g.

can refuse to work with "rogue" implementations that do not build proper LEAF's.

This completes our description of the basic translucent cryptography protocol.

4 Non-interactive Fractional Oblivious Transfer

We call an oblivious transfer scheme *fractional* if the probability p that Larry successfully receives the message may be chosen to be different from $1/2$. The only previous literature on fractional oblivious transfer schemes that we know of is by Brassard et al. [5], who discuss the special case of transferring one message out of a set of n messages. We now explain how to achieve non-interactive fractional oblivious transfer schemes for a variety of values for p.

Let's say that a p-NFOT is a Non-Interactive Fractional Oblivious Transfer protocol in which the transfer probability is p. Our goal is to design such protocols for given values of the probability $p \in [0, 1]$. We begin by noting simple solutions for certain values of p. Then we move on to the general case, and present two protocols.

4.1 Some simple special cases

A ONE OUT OF n NFOT. To obtain a simple form of fractional capability, it is easy to modify the basic scheme discussed above to provide "one of n" capability non-interactively. (That is, given n, we can design a p-NFOT with $p = 1/n$.) First, for technical reasons, we would work not over Z_q^* but over a multiplicative group of G of prime order q.[4] As before, g is a generator (now of G) and $U \in G$ is such that $\log_g(U)$ is both unknown and infeasible for anyone to compute. Larry publishes a list of values $(V_0, V_1, \ldots, V_{n-1})$ such that $V_i = V_0 U^i$, in such a way that only one of these keys is good for Larry. (See below for how.) Alice checks that indeed $V_i = V_0 U^i$ for $i = 0, \ldots, n-1$, then picks one of Larry's public keys at random, and uses it to encrypt the message to be sent to him.

To make his key, Larry picks $x \in Z_q$ at random and $i \in \{0, \ldots, n-1\}$ at random. He sets $V_i = g^x$, and then sets $V_j = V_i U^{j-i}$ for $j \neq i$. One can check that $V_j = V_0 U^j$ for all $j = 0, \ldots, n-1$.

On the other hand, no matter how Larry makes his key, he cannot know the discrete logs of two (or more) members of the list of group elements which comprises his key. For, say, he knew x_i, x_j such that $V_i = g^{x_i}$ and $V_j = g^{x_j}$ where $0 \leq i < j < n$. Dividing, we see that $U^{j-i} = g^{x_j - x_i}$. It follows that $\log_g(U)$ can be computed, as $\log_g(U) = (j-i)^{-1}(x_j - x_i) \bmod q$, where $(j-i)^{-1}$ represents the multiplicative inverse of $j - i$ in the field Z_q. (It is to ensure this inverse exists that we work in a group of prime order.)

A $n-1$ OUT OF n NFOT. Similarly for any n it is easy to obtain a p-NFOT with $p = (n-1)/n$. Larry publishes a list of values $(V_0, V_1, \ldots, V_{n-1})$ such that $\prod_{i=0}^{n} V_i = U$, in such a way that $n-1$ of these keys are good for Larry. (This is easy for Larry to do; the details are omitted here.) Alice checks the product constraint, then picks one of Larry's public keys at random, and uses it to encrypt the message to be sent to him. The product constraint implies that Larry cannot know the discrete logs of all the keys, so the transfer probability is $(n-1)/n$.

ARBITRARY p. Now our goal is to accomplish p-NFOT for an arbitrary, given value of $p \in [0, 1]$. We would like, ideally, to be as efficient as the above schemes, and use just one encryption. We do not accomplish this in our first scheme, the binary scheme of Section 4.2 below, where we use

[4] The order of Z_q^* is $q-1$ which is not prime. For more on groups of prime order, refer ahead to Section 4.3 where we use them again.

a number of encryptions proportional to the number of bits in the binary expansion of p. Then in Section 4.3 we present another scheme which requires only one encryption.

4.2 The binary scheme

Here is a way to extend the basic scheme to get a fractional scheme where the probability p can be any finite binary fraction $p = a/2^n$, where a is an integer in the range 1 to $2^n - 1$. (The cases $p = 0$ and $p = 1$ can be easily handled without any oblivious transfer; for $p = 1$ Alice merely needs to encrypt s with an additional public key known to be good for Larry.) In this solution Alice will use a number of encryptions depending on p to accomplish the transfer. (Specifically, $2n$ encryptions, which is $4n$ exponentiations.)

Let the n-bit binary expansion of $p = a/2^n = 0.a_1a_2\ldots a_n$, so that

$$p = \textstyle\sum_{i=1}^{n} a_i 2^{-i} \ .$$

We assume the values of n, a, and p are public knowledge, as are q, g, and U—the global set-up phase is the same as above.

Key setup. In the publication of public keys phase, Larry publishes a sequence of n pairs of public keys:

$$(V_1, V_1'), (V_2, V_2'), \ldots, (V_n, V_n') \ ,$$

where exactly one key in each pair is good for Larry. For each i, $1 \leq i \leq n$, Larry privately flips a coin to determine whether V_i or V_i' will be good for him, and proceeds to generate the i-th pair of keys as follows. If the i-th coin flip is "heads," he first randomly picks a x_i between 0 and $q - 2$, and sets

$$(V_i, V_i') = (g^{x_i}, Ug^{x_i}) \ ,$$

so that he knows the logarithm x_i of V_i. Otherwise he randomly picks a x_i' between 0 and $q - 2$, and sets

$$(V_i, V_i') = (g^{x_i'}/U, g^{x_i'}) \ ,$$

so that he knows the logarithm x_i' of V_i'. Note that $V_i' = UV_i$ in either case, which can be checked by anyone. Of course, Larry should not tell anyone which public keys are good for him.

Transfer. In the communication phase, Alice can non-interactively and obliviously transfer a message s to Larry so that he receives it with probability p, as follows. This will require sending a sequence of n triples

$$T_1, T_2, \ldots, T_n$$

to Larry, where each triple contains two values encrypted with Larry's keys V_i and V_i'. (In our application, we imagine that n probably need not be larger than about five to obtain satisfactory precision in the value of p, so that this sequence is actually quite short.)

First, Alice chooses a sequence of n keys K_1, K_2, $\ldots$, K_n as random values modulo q, and computes their running sums:

$$\begin{aligned} L_0 &= 0, \text{ and} \\ L_i &= K_1 + K_2 + \ldots + K_i \pmod{q}, \text{ for } i = 1, 2, \ldots, n \ . \end{aligned}$$

She also determines a value J_i for each i, $1 \leq i \leq n$:

$$J_i = \begin{cases} 0 & \text{if } a_i = 0, \\ s + L_{i-1} \pmod{q} & \text{if } a_i = 1 \ . \end{cases}$$

Each J_i is either "junk" (0, if $a_i = 0$) or a "jewel" ($s + L_{i-1}$, if $a_i = 1$).

Second, Alice chooses a sequence of n "random" bits $r_1, r_2, \ldots, r_n$. (We will return to the question as to how Alice might generate these bits later.)

Finally, Alice sends Larry a sequence of n triples, where the i-th triple T_i, for $1 \le i \le n$, contains r_i and the encrypted versions of J_i and K_i, and where the encryption is performed using Larry's public keys V_i and V_i' as follows.

- If $r_i = 0$, J_i is encrypted with V_i and K_i is encrypted with V_i'; the i-th triple is
$$T_i = (0, E(J_i, V_i), E(K_i, V_i')) \ .$$
- Otherwise (if $r_i = 1$) the public keys are switched: J_i is encrypted with V_i' and K_i is encrypted with V_i; the i-th triple is
$$T_i = (1, E(J_i, V_i'), E(K_i, V_i)) \ .$$

Since each triple contains r_i, Larry knows which way his public keys were used. For each i, Larry can decrypt either J_i or K_i; he knows which he can decrypt and which he cannot.

This completes our description of a rather straightforward way to implement non-interactive fractional oblivious transfer.

WHY THIS WORKS. To see that this scheme works as advertised, note that Alice sends Larry exactly n triples, and that Larry can decrypt exactly one ciphertext of each triple. On the i-th triple, if $a_i = 0$, Larry gets either junk (0) or a key (K_i). If $a_i = 1$, Larry either gets a jewel ($s + L_{i-1}$), or a key (K_i). Larry knows whether he gets junk, a jewel, or a key, since he knows a_i and r_i. Larry obtains s if and only if he gets $t-1$ keys followed by a jewel, for some t, $1 \le t \le n$. He can tell if he is able to obtain s or not. Since succeeding in position t happens only if $a_t = 1$, and then only with probability 2^{-t}, Larry receives s with probability exactly p, by equation (4.2).

ALICE'S RANDOM BITS r. Note that as long as Larry chooses which of each pair of public keys are good for him at random, then it does not matter whether or not Alice chooses the bits r_i randomly; Larry has a chance of exactly p of reading any particular message.

Similarly, as long as Alice chooses her bits r at random, then Larry will have a chance of exactly p of reading any particular message, even if Larry did not randomly decide which of each pair of public keys would be good.

However, we observe that Alice can in principle, if she wishes, choose her r bits to be identical or correlated from message to message. In some situations this might give her a perceived advantage, since this might allow Larry to read all of a sequence of messages, or none of them.

To help ensure that Alice uses appropriate randomness, one could require that Alice's random bits r be determined in some fixed manner, say by cycling sequentially through all possible values for r, or by hashing (taking a message digest of) the first field (the encrypted message m). It is easy for Larry to determine whether or not Alice is complying with this standard procedure. The second procedure is not perfect, since Alice can encrypt several variants of the same message, and only transmit those with desired r values, but this approach may not help her, inasmuch as she doesn't know which keys are good for Larry.

4.3 The polynomial scheme

We now propose a different scheme in which Alice needs only a single encryption (costing two exponentiations) in order to accomplish the transfer, regardless of the value of p.

Preliminaries. We consider a transfer probability of the form $p = a/m$ where a, m are integers satisfying $0 < a \leq m$. (In the binary scheme, $m = 2^n$ was a power of two. Here we won't make this restriction.) The scheme is based, as before, on the hardness of discrete logarithms, but this time in a group G of prime order $q > m$ for which the discrete logarithm problem is hard. There are many ways to get such groups. A simple, concrete implementation is to choose a prime $\rho = 2q + 1$ where $q > m$ is also prime, and let G be a subgroup of order q of Z_ρ^*. (Specifically, we can fix and publicize an element $\theta \in Z_\rho^*$ of order q, and let $G = \langle\theta\rangle$ be the sub-group generated by θ. Under this implementation, the arithmetic operations are all in Z_ρ^*, so have the usual costs.)

Notice that all non-trivial elements of G are generators of G. We let g be a randomly chosen generator of G, so $G = \{ g^i : i \in Z_q \}$. It is important for us that the index set Z_q is itself a field, which is why we chose G to be of prime order q. As before, we let $U \in G$ be an element for which $\log_g(U)$ is unknown. We let $\alpha_0 = 1 \in Z_q$. We also fix m *distinct* elements $\alpha_1, \ldots, \alpha_m$ of $Z_q^* - \{\alpha_0\}$. (It must be that $\alpha_0 = 1$. But it does not matter what $\alpha_1, \ldots, \alpha_m$ are as long as they are distinct, non-zero, and non-one, and we suggest the reader think of them as $2, 3, \ldots, m+1$. That $\alpha_0, \ldots, \alpha_m$ must be distinct is the reason we have $q > m$.)

The values p, a, m, ρ, q, g, and $\alpha_0, \ldots, \alpha_m$ are all fixed and public.

The idea. Before specifying the scheme, let us try to give a brief, informal overview of the ideas. Larry will form a public key $V_1, \ldots, V_m, W_0, \ldots, W_a$ consisting of $m + a + 1$ elements of G. The last $a + 1$ elements will be used only by Alice to verify that Larry's key is properly made. Letting $x_i = \log_g(V_i) \in Z_q$ for $i = 1, \ldots, m$, the key will be chosen so that:

(1) Larry knows a random, size a subset of $\{x_1, \ldots, x_m\}$.

(2) There exists a degree a polynomial $f(x) = f_0 + f_1 x + \cdots + f_a x^a \in Z_q[x]$ such that

  (2.1) $x_i = f(\alpha_i)$ for all $i = 1, \ldots, m$, and

  (2.2) *Larry does not know f.*

Furthermore, this will be done in such a way that Alice can check the property (2). Now if Larry does not know f then he cannot know *more* than a of the values $x_1, \ldots, x_m$ (otherwise he could interpolate to find the coefficients of f). Thus, in fact, he knows exactly a of these values. Now to accomplish the transfer, Alice can choose one key out of $V_1, \ldots, V_m$ at random, and use it as before. This calls, on the part of Alice, for only a single encryption.

The problem is how to set up the constraints we have discussed. Obviously we cannot have Larry choose f, since then he would know it. Instead, we make Larry specify $W_0, \ldots, W_a$ in some particular way, and then view the coefficients of the polynomial as *implicitly* specified by $f_i = \log_g(W_i)$ for $i = 0, \ldots, a$. Furthermore, we will ensure (and Alice will check) that $W_0 \cdot W_1 \cdots W_a = U$, which implies that Larry doesn't know all of $f_0, \ldots, f_a$, and hence doesn't know f. (In our scheme if Larry is honest he will in fact know the discrete logs of none of the W_i's.) Furthermore, Alice can verify item (2.1) above using a technique of Feldman [13] and Pedersen [23] used for verifiable secret sharing.

Larry will proceed by first specifying a random, size a subset of $V_1, \ldots, V_m$ in such a way that he knows the discrete logs of these a elements. Then, we will show how he can compute $W_0, \ldots, W_a$ by a linear algebraic technique. Finally, he will use these values to specify the remaining $m - a$ elements amongst $V_1, \ldots, V_m$. Let us now describe the scheme in full.

Key setup. Larry chooses at random a size a subset of $[m] = \{1, 2, \ldots, m\}$. This choice can be thought of as specifying an injective map $\pi\colon [a] \to [m]$, where $\pi(1), \ldots, \pi(a)$, all distinct, are the a

chosen indices. He now chooses elements $x_{\pi(1)}, \ldots, x_{\pi(a)} \in Z_q$ at random and sets

$$V_{\pi(l)} \;=\; g^{x_{\pi(l)}} \in G \qquad \text{for } l = 1, \ldots, a \,. \tag{1}$$

(This specifies a of the elements $V_1, \ldots, V_m$ in such a way that Larry knows their discrete logs. The other $m - a$ still need to be specified, in such a way that Larry doesn't know, and can't compute, their discrete logs.) Now Larry defines the $a+1$ by $a+1$ Vandermonde matrix

$$A \;=\; \begin{bmatrix} \alpha_0^0 & \alpha_0^1 & \cdots & \alpha_0^a \\ \alpha_{\pi(1)}^0 & \alpha_{\pi(1)}^1 & \cdots & \alpha_{\pi(1)}^a \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{\pi(a)}^0 & \alpha_{\pi(a)}^1 & \cdots & \alpha_{\pi(a)}^a \end{bmatrix} .$$

Since A is Vandermonde it is invertible. Larry computes its inverse

$$B \;=\; A^{-1} \;=\; \begin{bmatrix} \beta_{0,0} & \beta_{0,1} & \cdots & \beta_{0,a} \\ \beta_{1,0} & \beta_{1,1} & \cdots & \beta_{1,a} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{a,0} & \beta_{a,1} & \cdots & \beta_{a,a} \end{bmatrix} .$$

The arithmetic here is over the field Z_q. (Notice that in saying this inverse exists and can be computed we need the fact that Z_q is a field. This is why we choose G to be of prime order q.) Larry now sets

$$\begin{aligned} W_0 &= U^{\beta_{0,0}} \cdot \textstyle\prod_{l=1}^a V_{\pi(l)}^{\beta_{0,l}} \\ W_1 &= U^{\beta_{1,0}} \cdot \textstyle\prod_{l=1}^a V_{\pi(l)}^{\beta_{1,l}} \\ \vdots & \qquad\quad \vdots \\ W_a &= U^{\beta_{a,0}} \cdot \textstyle\prod_{l=1}^a V_{\pi(l)}^{\beta_{a,l}} \,, \end{aligned} \tag{2}$$

the arithmetic here being in G. (We will see that by doing this, Larry has implicitly chosen the polynomial $f(x) = f_0 + f_1 x + \cdots f_a x^a \in Z_q[x]$ where $f_i = \log_g(W_i)$. But Larry does not know $f_0, \ldots, f_a$.) Now Larry specifies the remaining V_i's as follows— he sets

$$V_i \;=\; \textstyle\prod_{j=0}^a W_j^{\alpha_i^j} \qquad \text{for all } i \in [m] \text{ that are not in the range of } \pi \,, \tag{3}$$

the arithmetic being in G. Finally, Larry outputs $(V_1, \ldots, V_m, W_0, \ldots, W_a)$ as his public key.

Properties of this key. To better understand what follows, it is worth saying something about what Larry accomplishes by the above steps. The following claim says that he is implicitly defining the polynomial $f(x) = f_0 + f_1 x + \cdots + f_a x^a \in Z_q[x]$ by the matrix equation Equation 4, and that his key is related to this polynomial as we would like.

Claim 4.1 Suppose Larry follows the key generation procedure described above. Define

$$\begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_a \end{bmatrix} = \begin{bmatrix} \beta_{0,0} & \beta_{0,1} & \cdots & \beta_{0,a} \\ \beta_{1,0} & \beta_{1,1} & \cdots & \beta_{1,a} \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{a,0} & \beta_{a,1} & \cdots & \beta_{a,a} \end{bmatrix} \begin{bmatrix} \log_g(U) \\ x_{\pi(1)} \\ \vdots \\ x_{\pi(a)} \end{bmatrix} , \tag{4}$$

the arithmetic being in Z_q, and let $f(x) = f_0 + f_1 x + \cdots + f_a x^a \in Z_q[x]$. Then

(1) $\log_g(W_j) = f_j$ for all $j = 0, \ldots, a$, and

(2) $\log_g(V_i) = f(\alpha_i)$ for all $i = 1, \ldots, m$, and, finally,

(3) $f_0 + f_1 + \cdots + f_a = \log_g(U)$.

The proof of this claim is in Appendix A. Note that from item (1) we have $W_j = g^{f_j}$, and thus from item (3) we have $W_0 \cdot W_1 \cdots W_a = U$, which is the product constraint that Alice will check.

Verification. Alice verifies the public key $(V_1, \ldots, V_m, W_0, \ldots, W_a)$ as follows. First, she checks the size, namely that it really consists of m elements of G followed by another $a+1$ elements of G. Then she checks two things—

$$U = W_0 \cdot W_1 \cdots W_a \tag{5}$$

$$V_i = \prod_{j=0}^{a} W_j^{\alpha_i^j} \quad \text{for all } i = 1, \ldots, m . \tag{6}$$

If these checks pass, she accepts the public key as valid.

One can check that Claim 4.1 implies that if Larry is honest then these checks do succeed. More important, however, is that even if Larry is not honest, this verification guarantees Alices that Larry will not receive the OT with probability *more* than p. Why this is true is discussed below.

We note that Alice has to perform this verification step only once, no matter how many messages she sends.

Transfer. As we have already indicated, to perform the p-NFOT, Alice picks $i \in [m]$ at random, and uses V_i as the key with which to encrypt her message $s \in G$. Namely, she picks $y \in Z_q$ at random and sends $E(s, V_i) = (c_1, c_2) = (g^y, sV_i^y)$, the operations being in G.

Efficiency. The key feature is that transfer needs only one El Gamal encryption (which is two exponentiations), regardless of the value of $p = a/m$. We pay for this in the size of the public file, which is $O(k(m+a))$ where $k = |\rho|$ is the security parameter. (In the binary scheme, it was $O(k \log_2(m))$.) But this is not too important. The public file is down-loaded once (or at not too frequent intervals) and stored by Alice on her machine. The time needed to compute a ciphertext and the size of the ciphertext don't depend on the size of this file.

Security for Alice. The verification is for Alice's security; it is supposed to guarantee her that even if Larry is dishonest, he won't get her data with probability more than p. So consider a Larry who tries to cheat. His goal is to somehow create the public key so that he ends up knowing $\log_g(V_i)$ for more than a values of $i \in [m]$. The following claim implies Larry cannot cheat in this way. To state it we first need some terminology. Given elements $W_0, \ldots, W_a$ of G, we define the *polynomial defined by* $W_0, \ldots, W_a$ as $f(x) = f_0 + f_1 x + \cdots + f_a x^a$ where $f_j = \log_g(W_j)$ for $j = 0, \ldots, a$. Now, the following claim says that if verification succeeds then exactly the same conditions as in Claim 4.1 hold with respect to the polynomial defined by Larry's public key, even if Larry had tried to cheat.

Claim 4.2 Suppose Alice's verification of key $(V_1, \ldots, V_m, W_0, \ldots, W_a)$ is successful, and let $f(x) = f_0 + f_1 x + \cdots + f_a x^a \in Z_q[x]$ be the polynomial defined by $W_0, \ldots, W_a$. Then

(1) $\log_g(W_j) = f_j$ for all $j = 0, \ldots, a$, and

(2) $\log_g(V_i) = f(\alpha_i)$ for all $i = 1, \ldots, m$, and, finally,

(3) $f_0 + f_1 + \cdots + f_a = \log_g(U)$.

The proof is in Appendix A. In consequence of item (3), Larry can know at most a of the values $f_0, \ldots, f_a$, not matter how he plays, because otherwise he would know $\log_g(U)$. Intuitively, this means he doesn't know f. But now, from item (2), it follows that Larry can know at most a of the values $\log_g(V_1), \ldots, \log_g(V_m)$. This, intuitively, means that Larry cannot receive the transfer with probability higher than $a/m = p$. Notice that Alice's security depends on the intractability of the discrete logarithm problem for Larry.

SECURITY FOR LARRY. We want to argue that we have security for Larry, meaning that Alice doesn't know which subset of a out of n keys is the one for which Larry knows the discrete logs.

Claim 4.3 Suppose Larry uses the procedure prescribed above to construct his public key $(V_1, \ldots, V_m, W_0, \ldots, W_a)$. Then the distribution on this key is the same as if the key were generated by the following experiment:

(1) Pick $f_0, \ldots, f_a \in Z_q$ at random subject to $f_0 + f_1 + \cdots + f_a = \log_g(U)$,

(2) Let $f(x) = f_0 + f_1 x + \cdots + f_a x^a \in Z_q[x]$,

(3) For $i = 1, \ldots, m$ let $V_i = g^{f(\alpha_i)}$,

(4) For $j = 0, \ldots, a$ let $W_j = g^{f_j}$, and

(5) Output $(V_1, \ldots, V_m, W_0, \ldots, W_a)$.

The proof of Claim 4.3 is in Appendix A. Now, clearly, presented with a key from this distribution, Alice has no idea of what Larry knows about the $\log_g(V_i)$'s, even if she can compute discrete logs.

Based on this, one can argue that there is no "key-choosing" strategy for Alice under which her transfer probability is reduced below p. By this we mean the following. Suppose that instead of using a random V_i as key, Alice chooses, somehow, probabilities $p_1, \ldots, p_m$ summing to 1, and transfers as follows— she picks $i \in [m]$ according to the distribution $\Pr[i = j] = p_j$ for all $j \in [m]$, and then uses V_i as the key. (If she is honest, $p_j = 1/m$ for all $j \in [m]$.) Then her transfer probability is still p, regardless of the values of $p_1, \ldots, p_m$.

As for the binary scheme, it may be simplest to specify that Alice's "random" choices are to be made in a specific manner, say by cycling through all values.

5 Discussion and Variations

SET-UP. Note that Alice needs *no* "set-up" to follow the translucent cryptography protocol. She does not need to be a registered user, have any private keys escrowed, etc.

EFFICIENCY. With the proposal of Section 4.3 described above, Alice needs to perform 2 modular exponentiations (one El Gamal encryption) in order to compute the desired LEAF. An implementation can, if it wishes, precompute future session keys and their associated LEAFs as a means of decreasing the latency in encrypting a new message.

THE VALUE OF p. The value of p that is effective is the value of p that is embedded in Alice's translucent cryptography implementation.

Different categories of equipment could have different probabilities p. For example, software and hardware that are exported could have $p = 1$, while domestic versions could have $p = 0.02$.

Larry can monitor whether or not Alice is using the correct value of p, by monitoring what fraction of the time he actually succeeds in getting s.

Warrants. To ensure that Larry must get a warrant in order to decrypt his allowed fraction of the translucent crypto, the value transmitted obliviously should be the message key s encrypted with the public key of Jerry (the judge), or his designated agent who can be available in real-time to decrypt LEAFs. This encrypted block could also include the ID of the software or hardware generating the message, if the search warrant is to be restricted to messages from a single source.

Multiple agencies and multiple probabilities The LEAF could easily contain messages for two or more agencies that need to cooperate to get the final message key. Larry might receive message key s_1 encrypted with his public key, and Louis (who works for another organization) might receive message key s_2, encrypted with his public key. The actual message key s might be the sum (or the exclusive-or) of s_1 and s_2.

Differing agencies could even receive the message key with different probabilities. The FBI might receive the message key with probability 0.02, whereas an escrow agent of the user's choice might receive the message key with probability 1.

Stewart Baker (in a private communication that was probably intended to tease the authors) suggested that law enforcement might find this proposal more attractive if it were implemented in a related variant, making 1% of the messages accessible to law enforcement (without even a warrant(!). Another 20% or so of the messages would become accessible if suspicious activity is detected in the first 1%, and the remainder would become available to law enforcement with a court order. It is straightforward to implement such a variation based on our ideas. (It is not so easy, fortunately, to get around the Constitution!)

Export. Of course, non-U.S. companies may object to Larry accessing their communications, whether this access was obtained through key escrow or through translucent cryptography. Translucent cryptography is likely to fare no better in an international market than key escrow fares.

On the other hand, it is easy, for example, for U.S. manufacturers to develop products (say for France) that give U.S. access with probability 0.5 and the French government access with probability 1.0. This would merely require the use of two LEAF fields, one for each government.

The opening problem. A weakness of our implementations, inherited from a weakness of non-interactive oblivious transfer, is that in some circumstances, if Larry does excercise his privilage and decrypt the fraction p of Alice's traffic to which he is entitled, Alice may learn some information about Larry's secret key which would enable her, in future, to decrease the probability that Larry recovers her messages. This happens if Larry not only decrypts, but also reveals *which* ciphertexts he decrypted. (As long as Larry keeps secret the decrypted information, nothing is revealed.) For this reason it may be desirable for Larry to have many public keys, with different keys used in different programs, different devices, or products produced in different months. Let us explain this issue by example.

Say we are using the polynomial scheme, and Larry's public key is $(V_1, V_2, V_3, W_0, W_1)$, and Larry knows $\log_g(V_1) = x_1$. (The transfer probability here is $p = 1/3$.) Thus, Larry's secret key consists of two things: a secret index, namely 1, saying which of the three keys is a "receiving" one for Larry, and the value x_1, which enables the actual receipt. Larry's security relies on the fact that Alice does not know his secret index; if she did, she could encrypt using only the other keys, and Larry would never be able to recover the message.

Now suppose Alice encrypts 5 messages, and her choices of keys are V_2, V_1, V_2, V_3, V_2. Suppose Larry decides to wiretap. He will obtain the second message. A priori Alice does not know which message Larry got. But suppose now she learns, somehow, that Larry got the second message. Then she knows that Larry knows $\log_g(V_1)$, because V_1 was the key she used in the second message.

Thus she has determined Larry's secret index. Now she can fool him; in future, she will never use key V_1.

How could Alice learn which ciphertext Larry decrypted? The issue is how wiretap information is used. We expect that often Larry wiretaps for his own information; the recovered plaintexts are not revealed to the public. In such a case, Alice, or other users, learn nothing about Larry's secret index. But suppose Larry needs to use the wiretap information, say as evidence in a court case. The plaintexts are then revealed, and, by their examination, Alice can determine which of her messages were decrypted. This tells her what is Larry's secret index.

The extent to which this is a problem thus depends on the extent to which Larry intends to publicize information obtained by wiretaps. Since this must happen to some extent, we need to mitigate its effects. Our suggestion, as indicated above, is that Larry have many public keys, with different keys used in different programs or devices at different times.

For the benefit of a reader familiar with non-interactive oblivious transfer (NIOT), let us add some historical notes and comparison. The underlying issue of revelation of the secret index of a recipient in a NIOT based on some action of the recipient arose, and was recognized, in the context of implementing non-interactive zero-knowledge based on NIOT [2]. There the problem was that if the sender learned that her proof had been rejected then the receiver's secret index would leak. The suggestion of [2] to overcome this was to change the public key when a proof was rejected. But this is not too practical, because the sender can *force* revelation by sending bad proofs. (This issue, and attacks based on it, have been discussed a few times in the literature.) In comparison, in translucent cryptography, there is much less of a problem, because it is much harder to force Larry to reveal which ciphertexts he decrypted. Thus, our suggestion above, that Larry have many different public keys, seems to provide an acceptable resolution to this "opening" problem in this context.

OTHER WAYS OF GETTING AROUND THESE SCHEMES. With sufficient work, these schemes, like other proposals, are easy to get around. Two particularly relevant references are Wyner's papers on the "wire-tap channel" [26, 22]. Superencryption also defeats this approach, of course; these sorts of "work-arounds" on the part of a user are problems common to any such proposal for government access to messages.

WHY NIOT? A reader may ask why NIOT is used at all. Specifically, how about the following instead? Let Larry publicize a public key of a conventional public key cryptosystem such as RSA, and let $\mathcal{E}$ denote encryption under this key. (Larry knows the corresponding decryption key.) When Alice is to send a message m to Bob, she picks, as before, a session key s, uses it to produce the first field (F1) as described in Section 3. The second field too is as before. She now lets s^* equal s with probability p, and 0 otherwise. She then lets the LEAF be $\mathcal{E}(s^*)$. Larry can access the LEAF, and has s a fraction p of the time.

This is certainly much simpler than NIOT. But the problem is that it puts greater trust in Alice. Alice could cheat very easily, and yet evade detection. For example, whenever m is an "important" message she would choose $s^* = 0$, and otherwise choose $s^* = s$, doing this in such a way that she chooses $s^* = s$ a fraction p of the time. Then Larry gets only the unimportant stuff, but, because he is getting a fraction p of the plaintexts, he can't really complain. In contrast, in NIOT, there is no key-choosing strategy for Alice which lowers the transfer probability below p.

All implementations, whether of key escrow or translucent cryptography such as we discuss, rely on some trust in Alice. The question is the *degree* of this trust. Our goal is to make it as hard as possible for Alice to cheat. As discussed above, there will always be ways around the schemes;

but let us not make it *too* easy.

Comparison with key escrow. The approach proposed has the following advantages over key-escrow schemes:

(A1) Set-up is particular easy with our scheme; there is no escrow procedure required of users or manufacturers. We feel that this is a very significant advantage of our proposal.

(A2) There are no escrow agents holding users' keys, who might be tempted (or ordered) to abuse users' privacy. In our scheme, the corresponding agents are those parties holding the private keys corresponding to the published public keys.

(A3) There is a firm upper bound on the extent to which law enforcement can encroach on individual privacy; a certain fraction of Alice's messages will be private from everyone except their intended recipients.

(A4) There is a firm lower bound on the extent to which cryptography will prevent authorized wiretapping from being effective; a certain fraction of Alice's messages will be wire-tappable (on the average).

(A5) The scheme contains a variable-access rate p that may be changed according to the specific use or the perceived risks.

(A6) Compliance with the scheme can be monitored.

(A7) The scheme can be easily elaborated or combined with other approaches to meet more detailed requirements.

Our scheme has the following possible disadvantages:

(D1) Law enforcement may be frustrated that when it has an authorized wiretap, it is not getting decryption of *all* of the messages. (Too bad; that is the nature of the compromise proposed here.)

(D2) Individuals may be frustrated that this scheme does not provide absolute privacy for their messages; law enforcement can read some fraction of their messages. (Too bad; that is the nature of the compromise proposed here.)

Related work. Upton [25] has suggested using *interactive* oblivious transfer as a replacement for key escrow. In his suggestion, every time Alice wishes to communicate with Bob, she must first communicate with Larry, engaging in an oblivious transfer protocol in which she transfers to Larry either the session key or a random string, she doesn't know which. But this means Larry must actively participate in every communications session, which creates some significant practical problems.

6 Open Questions

Can one build an efficient non-interactive fractional oblivious transfer scheme based on RSA or the Rabin function rather than on the Diffie-Hellman assumption?

7 Conclusions

We have presented a novel alternative to standard key escrow schemes, that may allow a generally acceptable compromise to be reached on a difficult issue of national cryptographic policy. We have proposed an efficient implementation of it, based on a primitive that may be of independent interest.

Acknowledgments

We thank Stewart Baker, Tony Eng, Rosario Gennaro, Oded Goldreich, and Burt Kaliski for helpful comments and suggestions.

References

[1] D. Beaver. How to break a "secure" oblivious transfer protocol. In R.A. Rueppel, editor, *Proc. EUROCRYPT 92*, volume 658, pages 285–296, 1993.

[2] Mihir Bellare and Silvio Micali. Non-interactive oblivious transfer and applications. In G. Brassard, editor, *Proc. CRYPTO 89*, pages 547–559. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 435.

[3] R. Berger, R. Peralta, and T. Tedrick. A provably secure oblivious transfer protocol. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Proc. EUROCRYPT 84*, pages 379–386. Springer-Verlag, 1985. Lecture Notes in Computer Science No. 209.

[4] M. Blum. How to exchange (secret) keys. *Trans. Computer Systems*, 1:175–193, May 1983. (Previously published in ACM STOC '83 proceedings, pages 440–447.).

[5] G. Brassard, C. Crepeau, and J.-M. Robert. Information theoretic reductions among disclosure problems. In *27th Annual Symposium on Foundations of Computer Science*, pages 168–173, Toronto, Ontario, Canada, 27–29 October 1986. IEEE.

[6] Claude Crépeau. Equivalence between two flavours of oblivious transfers. In Carl Pomerance, editor, *Proc. CRYPTO 87*, pages 350–354. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 293.

[7] A. De Santis, G. Di Crescenzo, and G. Persiano. Zero-knowledge arguments and public key cryptography. *Information and Computation*, 121(1):23–40, 1995.

[8] A. De Santis and G. Persiano. Public-randomness in public-key cryptography. In I.B. Damgård, editor, *Proc. EUROCRYPT 90*, volume 473, pages 46–62, 1991.

[9] B. den Boer. Oblivious transfer protecting secrecy. In I.B. Damgård, editor, *Proc. EUROCRYPT 90*, volume 473, pages 31–45, 1991.

[10] Dorothy E. Denning. Resolving the encryption dilemma: The case for the Clipper Chip. *Technology Review*, pages 48–55, July 1995.

[11] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. In R. L. Rivest, A. Sherman, and D. Chaum, editors, *Proc. CRYPTO 82*, pages 205–210, New York, 1983. Plenum Press.

[12] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28:637–647, 1985.

[13] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. 28th IEEE Symp. on Foundations of Comp. Science*, pages 427–438, Los Angeles, 1987. IEEE.

[14] M. Fischer, S. Micali, and C. Rackoff. Presentation made at Eurocrypt 84.

[15] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inform. Theory*, 31:469–472, 1985.

[16] L. Harn and H.-Y. Lin. An oblivious transfer protocol and its application for the exchange of secrets. In H. Imai, R.L. Rivest, and T. Matsumoto, editors, *Advances in Cryptology–ASIACRYPT '91*, volume 739, pages 312–320, 1993.

[17] Lance J. Hoffman, editor. *Building in Big Brother: The Cryptographic Policy Debate*. Springer-Verlag, 1995.

[18] Stephen T. Kent. Internet privacy enhanced mail. *Communications of the ACM*, 36(8):48–60, August 1993.

[19] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. 20th ACM Symp. on Theory of Computing*, pages 20–31, Chicago, 1988. ACM.

[20] Silvio Micali. Fair public-key cryptosystems. In Ernest F. Brickell, editor, *Proc. CRYPTO 92*, pages 113–138. Springer-Verlag, 1992. Lecture Notes in Computer Science No. 740.

[21] National Institute of Standards and Technology (NIST). *FIPS Publication 185: Escrowed Encryption Standard*, February 9, 1994.

[22] L.H. Ozarow and A.D. Wyner. Wire-tap channel II. In T. Beth, N. Cot, and I. Ingemarsson, editors, *Proc. EUROCRYPT 84*, pages 33–50. Springer-Verlag, 1985. Lecture Notes in Computer Science No. 209.

[23] T.P. Pedersen. Distributed provers with applications to undeniable signatures. In D.W. Davies, editor, *Proc. EUROCRYPT 91*, volume 547, pages 221–242, 1991.

[24] M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[25] Jimmy Upton. Unpublished comment made to Whit Diffie before Crypto 93, and mentioned by Diffie in the Crypto '93 rump session.

[26] A. D. Wyner. The wire-tap channel. *Bell Sys. Tech. J.*, 54:1355–1387, 1975.

A Proofs of Claims

Proof of Claim 4.1: From Equation 4 we have

$$f_j = \beta_{j,0}\log_g(U) + \textstyle\sum_{l=1}^{a}\beta_{j,l}x_{\pi(l)} \qquad \text{for } j = 0,\ldots,a\ . \tag{7}$$

Now from Equation 2 we have

$$\begin{aligned}
\log_g(W_j) &= \log_g\left(U^{\beta_{j,0}}\cdot\textstyle\prod_{l=1}^{a}V_{\pi(l)}^{\beta_{j,l}}\right)\\
&= \beta_{j,0}\log_g(U) + \textstyle\sum_{l=1}^{a}\beta_{j,l}\log_g(V_{\pi(l)})\\
&= \beta_{j,0}\log_g(U) + \textstyle\sum_{l=1}^{a}\beta_{j,l}x_{\pi(l)}\\
&= f_j\ ,
\end{aligned}$$

proving item (1) of the claim. (Here we used Equation 1, namely the fact that $V_{\pi(l)} = g^{x_{\pi(l)}}$ by definition.) Now, multiply both sides of Equation 4 by the matrix A, and use the fact that $AB = I$, to get

$$\begin{bmatrix} \alpha_0^0 & \alpha_0^1 & \cdots & \alpha_0^a \\ \alpha_{\pi(1)}^0 & \alpha_{\pi(1)}^1 & \cdots & \alpha_{\pi(1)}^a \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{\pi(a)}^0 & \alpha_{\pi(a)}^1 & \cdots & \alpha_{\pi(a)}^a \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_a \end{bmatrix} = \begin{bmatrix} \log_g(U) \\ x_{\pi(1)} \\ \vdots \\ x_{\pi(a)} \end{bmatrix} .$$

In other words,

$$\sum_{l=0}^{a} \alpha_0^l f_l = \log_g(U) \tag{8}$$

$$\sum_{l=0}^{a} \alpha_{\pi(j)}^l f_l = x_{\pi(j)} \qquad \text{for } j = 1, \ldots, a . \tag{9}$$

But recall that $\alpha_0 = 1$. Thus Equation 8 directly gives us item (3) of the claim. Furthermore, note that Equation 9 is the same as

$$f(\alpha_{\pi(j)}) = x_{\pi(j)} \qquad \text{for } j = 1, \ldots, a ,$$

which establishes item (2) for all i in the range of π. Now we must check item (2) for i not in the range of π. For these i we know that V_i is defined by Equation 3. Taking discrete logs of both sides of that equation we have

$$\log_g(V_i) = \log_g\left(\prod_{j=0}^{a} W_j^{\alpha_i^j}\right) = \sum_{j=0}^{a} \alpha_i^j \log_g(W_j) = \sum_{j=0}^{a} \alpha_i^j f_j = f(\alpha_i) ,$$

as desired. This completes the proof. ▮

Now, we would like to discuss the security. Refer to Section 4.3 for the definition of the polynomial defined by some elements of G.

Proof of Claim 4.2: Item (1) is a tautology. Taking discrete logs of both sides of Equation 6 gives

$$\log_g(V_i) = \log_g\left(\prod_{j=0}^{a} W_j^{\alpha_i^j}\right) = \sum_{j=0}^{a} \alpha_i^j \log_g(W_j) = \sum_{j=0}^{a} \alpha_i^j f_j = f(\alpha_i) ,$$

establishing item (2). Finally, taking discrete logs of both sides of Equation 5 proves item (3). ▮

Proof of Claim 4.3: Fix the map π chosen by Larry. Now let $f_0, \ldots, f_a \in Z_q$ be arbitrary subject to their sum being $\log_g(U)$. We argue that there is a unique choice of $x_{\pi(1)}, \ldots, x_{\pi(a)} \in Z_q$ such that Equation 4 holds, namely

$$\begin{bmatrix} x_{\pi(1)} \\ \vdots \\ x_{\pi(a)} \end{bmatrix} = \begin{bmatrix} \alpha_{\pi(1)}^0 & \alpha_{\pi(1)}^1 & \cdots & \alpha_{\pi(1)}^a \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{\pi(a)}^0 & \alpha_{\pi(a)}^1 & \cdots & \alpha_{\pi(a)}^a \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_a \end{bmatrix} . \tag{10}$$

To see that this choice makes Equation 4 hold, first note that since $\alpha_0 = 1$ and $f_0 + \cdots + f_a = \log_g(U)$ we have

$$\begin{bmatrix} \log_g(U) \\ x_{\pi(1)} \\ \vdots \\ x_{\pi(a)} \end{bmatrix} = \begin{bmatrix} \alpha_0^0 & \alpha_0^1 & \cdots & \alpha_0^a \\ \alpha_{\pi(1)}^0 & \alpha_{\pi(1)}^1 & \cdots & \alpha_{\pi(1)}^a \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_{\pi(a)}^0 & \alpha_{\pi(a)}^1 & \cdots & \alpha_{\pi(a)}^a \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_a \end{bmatrix},$$

and now multiplying both sides by B yields Equation 4. On the other hand the choice of Equation 10 is unique because multiplying both sides of Equation 4 by A recovers it.

This means that for any fixed π, any vector $f_0, \ldots, f_a$ with $f_0 + \cdots + f_a = \log_g(U)$ has the same probability of being defined by Larry's choices in his key construction process. Since the other quantities, namely $V_1, \ldots, V_m, W_0, \ldots, W_a$, are uniquely defined given $f_0, \ldots, f_a$, we have established Claim 4.3. ∎