Equitable Key Escrow with Limited Time Span

(or, How to Enforce Time Expiration Cryptographically)

Extended Abstract

Mike Burmester[1], Yvo G. Desmedt[2,1] and Jennifer Seberry[3]

[1] Information Security Group, Royal Holloway – University of London
Egham, Surrey TW20 OEX, U.K.
m.burmester@rhbnc.ac.uk

[2] Center for Cryptography, Computer and Network Security
Department of EE & CS, University of Wisconsin – Milwaukee
P.O. Box 784, WI 53201-0784, U.S.A.
desmedt@cs.uwm.edu

[3] Center for Computer Security Research, University of Wollongong, Australia,
jennie@uow.edu.au

Abstract. With equitable key escrow the control of society over the individual and the control of the individual over society are shared fairly. In particular, the control is limited to specified time periods. We consider two applications: time controlled key escrow and time controlled auctions with closed bids. In the first the individual cannot be targeted outside the period authorized by the court. In the second the individual cannot withhold his closed bid beyond the bidding period. We propose two protocols, one for each application. We do *not* require the use of temper-proof devices.

Key Words: key escrow, auctions with closed bids, time stamps.

1 Introduction

Key escrow has been proposed as a mechanism to protect society from individuals who use a communication system for criminal purposes [4, 24, 10] (an excellent survey of key escrow systems is given by D.E. Denning and D.K. Branstad in [11]). However key escrow can also be used to target innocent individuals. This potential targeting is a major factor which contributes to the social unacceptability of key escrow. ¿From the point of view of an individual, key escrow may restrict his/her privacy and give controlling power to society (Big Brother [8]), which may, in certain circumstances, abuse it. In a society oriented key escrow this power must be equally shared between the individual and society (for an analysis of fair cryptosystems see [24, 22]). Furthermore it must have a limited life span. Indeed a major objection to currently proposed key escrow schemes is that there is no effective time control. Once an order to recover a key by the escrow agents has been given, there is nothing to prevent the agents from abusing their power and decrypting all wire-tapped messages, far beyond the time specified by the Court order. Various scenarios can be envisaged in which a threat against a minority is indeed serious. While the Bellare–Goldwasser[3] scheme protects a majority against Big Brother, it does not protect a minority. For example, an extremist group aiming to take control of the government can wire-tap all communication of suspect dissidents, which would then be decrypted when the group took over control.

It is essential that the control of the escrow agents be limited to specified time periods, beyond which it should not be possible for the agents to recover the "old" private keys of a targeted individual. For this purpose we have chosen in our first application of equitable key escrow, to update the keys at regular intervals, and to make it infeasible to compute old keys from the new key. The escrow agents must destroy all the shares of the old keys with each updating. We can allow for a small number of corrupted agents who keep their old shares, but these should not be sufficient to reconstruct the keys.

Our second application of equitable key escrow is contract bidding. In this case it is the individual who may try to abuse society. To prevent a tender from being opened before the specified date, it is encrypted with an escrowed key. The bidder must have some control over the encryption otherwise one can envisage situations in which the escrow agents may collude with a corrupted receiving agent. This threat can be eliminated if the bidder pre-encrypts the bid with his/her own key. However the bidder may then withhold the key. There are several scenarios in which such a threat may be of concern. For example, if altered circumstances make the bid unprofitable, or loss making. In this case, it is "society" (the receiving office) which is threatened by the individual (the bidder). The solution we propose is to force the bidder to use a weak encryption key (a nice discussion on the use of weak keys is given in [28]). This imposes a time limit which should make it possible for the agents to recover the bid after the tender is opened. Two keys are used: a key for the bidder and an escrowed key. The pair of these keys can be regarded as an enlarged escrow key, in which the share of the bidder is her/his key while the shares of the agents are their old shares. In this way the bidder is included in all authorized sets.

Our goal in this paper is to design protocols which achieve equitable key escrow. For this purpose we combine the threshold scheme of Boyd [7], the El-Gamal threshold scheme of Desmedt–Frankel [14] and add time dependency. The organization of this paper has as follows. In Section 2 we present our first protocol for a time controlled key escrow system and discuss its security. In Section 3 we present a protocol for time controlled auctions with closed bids.

Notation and Background

Let p be a prime and $g \in Z_p$ an element of large order. All operations in Z_p are performed modulo p. For simplicity, and when there is no ambiguity, we drop the operator "mod$p$". We also write $x \in_R X$ to indicate that the element

x is selected uniformly at random from the set X, independently of all other selections.

The Diffie-Hellman [16] operator DH is defined by $\mathrm{DH}(g^x, g^y) = g^{xy}$. The problem of finding $\mathrm{DH}(g^x, g^y)$, given g^x, g^y, is believed to be hard, and is called the Diffie-Hellman problem. If g^x, g^y and $z \in Z_p$ are given, then the problem of deciding whether $z = \mathrm{DH}(g^x, g^y)$ is called the Diffie-Hellman *decision* problem. If this problem is hard then so is the Diffie-Hellman problem. The *symmetric* Diffie-Hellman problem (called the *squaring DH-oracle* in [23]) is the problem of finding $\mathrm{DH}(g^x, g^x)$ given g^x. This problem is as hard as the Diffie-Hellman problem under some reasonable conditions [23, Theorem 2]. The problem of deciding whether $z = \mathrm{DH}(g^x, g^x)$, given z, g^x is the symmetric Diffie-Hellman *decision* problem. If this problem is hard then so are the Diffie-Hellman problem, the Diffie-Hellman decision problem and the symmetric Diffie-Hellman decision problem. We will also consider the problem of finding elements with large order in Z_p. This is related to Problem C19 in the Adleman–McCurley list of open problems in Number Theoretic Complexity [1], and is considered to be hard.

2 Time controlled key escrow

For simplicity we focus on a basic ℓ-out-of-ℓ escrow system. We will discuss generalizations to other access structures later on.

Our system uses a Discrete Logarithm setting with prime modulus p and $g \in Z_p$ an appropriate element of large order. Initially, at time $t = 0$, the private key of the receiver, Bob, is $a \in_R Z_{p-1}$ and the public key is $y_0 = g^a \bmod p$. Bob shares his private key among ℓ escrow agents EA_i, $i = 1, 2, \ldots, \ell$.

In our basic model each agent gets a share $s_i \in_R Z_{p-1}$ $(i = 1, 2, \ldots, \ell - 1)$, and s_ℓ is such that $s_1 \cdot s_2 \cdots s_\ell = a \bmod (p-1)$. The main feature of our system is that the private key of Bob and its shares are updated at regular intervals without the need for interaction. At time t, the private key of Bob is updated to $a^{2^t} \bmod (p-1)$, the shares are updated to ${s_i}^{2^t} \bmod (p-1)$, and the public key is updated to $y_t = g^{a^{2^t}} \bmod p$. The agents EA_i compute the new shares by themselves, and *must* destroy the old shares. As a consequence, the escrow agents cannot decrypt a ciphertext which was encrypted with an old key at a later date, even if forced. We shall prove that the problem of decrypting encryptions with earlier keys is related to two problems: the problem of finding elements of large order in Z_p and the symmetric Diffie-Hellman decision problem. Both problems are believed to be hard (*cf.* [1, 23]).

We first describe our basic protocol in more detail. This combines ideas from [6, 7, 14, 26].

Setting

The parties involved: the sender Alice, the receiver Bob, a Court, the Law Enforcement Agency LEA, and the Escrow Agents EA_i, $i = 1, 2, \ldots, \ell$.
The parameters: A Discrete Logarithm setting is used. Bob chooses a prime p such that $p - 1$ has two large prime factors p_1, p_2, with $p_1 \equiv p_2 \equiv 3 \pmod 4$, so

$(-1 \mid p_1) = (-1 \mid p_2) = -1$ (p_1p_2 is a Blum integer [6]), and an element $g \in Z_p$ whose order is p_1p_2. Bob gives p, g to all the agents EA_i, $i = 1, 2, \ldots, \ell$, and to Alice.
Bob has a long term public key which is known to all parties concerned. This key is used for authenticating (signing) Bob's encryption keys and the parameters p, g, if required.

Set-up

Set $time := 0$.
Bob chooses his private key $a \in_R Z^*_{p-1}$ and finds ℓ shares s_i of it, $i = 1, \ldots, \ell$, by choosing $s_i \in_R Z^*_{p-1}$ for $i = 1, \ldots, \ell{-}1$, and taking $s_\ell = a{\cdot}(s_1 \cdots s_{\ell-1})^{-1}\mathrm{mod}(p-1)$.
The public key of Bob is $y_0 = g^a$. Bob publishes this key.
Then,

1. Bob gives privately to each agent EA_i, $i = 1, 2, \ldots, \ell$, the share s_i.
2. Bob publishes $z_1 := g^{s_1}, z_2 := g^{s_2}, \ldots, z_\ell := g^{s_\ell}$, and each agent EA_i checks that these are correct, that is that $z_i = g^{s_i}$, where s_i is its share. If any check fails then Bob has cheated and is reported to the LEA.
   Bob publishes $z_{1,2} := g^{s_1s_2}$, $z_{1,2,3} := g^{s_1s_2s_3}, \ldots,$ $z_{1,2,\cdots,\ell} := g^{s_1s_2\cdots s_\ell}\,(= y_0)$, and proves in zero-knowledge to the LEA that these are correctly constructed. That is, Bob proves that $z_{1,2,\ldots,k} = \mathrm{DH}(z_{1,2,\ldots,k-1}, z_k)$, for $k = 2, \ldots, \ell$, by using an interactive zero-knowledge proof for the Diffie-Hellman problem – an example of such a proof is given in Appendix A. If any of the proofs fails, then Bob has cheated and is reported to the LEA.

The protocol

Updating

At $time = t$
Each agent EA_i updates his share by squaring it, *i.e.*, the current share is ${s_i}^{2^t} \bmod (p-1)$, and then destroys the old share $({s_i}^{2^{t-1}} \bmod (p-1))$.
Bob updates his private key to $a^{2^t} \bmod (p-1)$ and publishes his public key $y_t := g^{a^{2^t}} \bmod p$. If necessary Bob proves to the LEA that this is correct by using an interactive zero-knowledge proof for the Diffie-Hellman problem (for example, the interactive proof given in Appendix A). That is, Bob proves that $y_t = \mathrm{DH}(y_{t-1}, y_{t-1})$.

Getting an escrowed key

1. Alice asks Bob for a new encryption key.
2. Bob sends Alice his public key which is authenticated with his long term key, $(p, g, y_t, \mathrm{sign}_{\mathrm{Bob}}(p, g, y_t))$.
3. If Bob's signature is valid then Alice sends Bob the encryption $\mathrm{ElG}(m) = (g^r, m{y_t}^r)$, $r \in_R Z^*_{p-1}$, of a message $m \in Z^*_p$ with key y_t.

4. If the Court has issued an order to recover the message, then the LEA will wire-tap the communication and send g^r to agent EA_1. The agents $\mathrm{EA}_1, \mathrm{EA}_2, \ldots, \mathrm{EA}_\ell$ then compute ${y_t}^r$ sequentially as follows: for $i < \ell$, each EA_i on receiving $g^{r \prod_{j=1}^{i-1} s_j^{2^t}}$ computes $g^{r \prod_{j=1}^{i} s_j^{2^t}} := (g^{r \prod_{j=1}^{i-1} s_j^{2^t}})^{s_i^{2^t}}$, which it sends to EA_{i+1}. Agent EA_ℓ then computes $(g^{r \prod_{j=1}^{\ell-1} s_j^{2^t}})^{s_\ell^{2^t}}$, which it sends to the LEA. Since this corresponds to ${y_t}^r$, the LEA can decrypt the ciphertext.

Security

Theorem 1. (Irreversible time) *If the symmetric Diffie-Hellman decision problem is hard and if finding elements of large order in Z_p is hard, then decrypting old ciphertext with new shares of the escrow agents is hard.*

Proof. (Sketch) At time $t-1$ the shares of the escrow agents are $s_i^{2^{t-1}}$ and the secret key of Bob is $a^{2^{t-1}}$. At time t the shares are updated to $s_i^{2^t}$, the old shares are destroyed, and Bob's secret key is updated to a^{2^t}. Observe that a^{2^t}, for $t > 1$, has 4 square roots in $Z_{p_1 p_2}$ of which only one is a quadratic residue, because of our restrictions on the primes p_1, p_2. It follows that there is only one primitive 2^u–th root of a^{2^t} in $Z_{p_1 p_2}$, $0 < u \leq t$, which is a quadratic residue.

We continue with the proof. Suppose that there is a polynomial time algorithm **A** which on input $p, g, z_1, z_2, \ldots, z_\ell,\ z_{1,2},\ z_{1,2,3},\ \ldots,\ z_{1,2,\ldots,\ell}$, the shares $s_1^{2^t}, s_2^{2^t}, \ldots, s_\ell^{2^t}$, the old shares of $(\ell-1)$ corrupted shareholders, Bob's long term key, the certificates $(p, g, y_i, \mathrm{sign}_{\mathrm{Bob}}(p, g, y_i))$, $i = 1, 2, \ldots, \ell$, and an old ciphertext (w_1, w_2), with $w_1 = g^r$, $r \in Z_{p-1}^*$, $w_2 = m y_{t-u}^r$, $m \in Z_p^*$, will output the message m. Then **A** can be used to compute $g^{r a^{2^{t-u}}}$ $(= w_2/m)$, since any of the other 2^u–th roots of $a^{2^{t-u}}$ will not produce the correct message. We now will use **A** to get an element in Z_p of large order. First we prepare an input for **A**.

Find an appropriate long term key for Bob. Take $b \in_R Z_{p-1}^*$ and choose $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_{\ell-1} \in_R Z_{p-1}^*$. Compute $\bar{s} := \bar{s}_1 \cdot \bar{s}_2 \cdots \bar{s}_{\ell-1}$, $\bar{s}_\ell := b \cdot (\bar{s})^{-1}$, and the public keys $\bar{y}_{t-u} = g^b$, $\bar{y}_{t-u+1} = g^{b^2}$, $\ldots$, $\bar{y}_t = g^{b^{2^u}}$. If $t = u$, we get ℓ shares $\bar{s}_i$ of b and the public keys $y_0 = g^b$, $\bar{y}_1 = g^{b^2}$, $\ldots$, $\bar{y}_t = g^{b^{2^t}}$. If $t > u$, take $b_0 \in_R Z_{p-1}^*$ and choose $\bar{\sigma}_1, \bar{\sigma}_2, \ldots, \bar{\sigma}_{\ell-1} \in_R Z_{p-1}^*$. Compute $\bar{\sigma} := \bar{\sigma}_1 \cdot \bar{\sigma}_2 \cdots \bar{\sigma}_{\ell-1}$, $\bar{z}_1 := g^{\bar{\sigma}_1}$, $\bar{z}_2 := g^{\bar{\sigma}_2}$, $\ldots$, $\bar{z}_{\ell-1} := g^{\bar{\sigma}_{\ell-1}}$, $\bar{z}_\ell := g^{b_0/\bar{\sigma}}$, $\bar{z}_{1,2} := g^{\bar{\sigma}_1 \bar{\sigma}_2}$, $\ldots$, $\bar{z}_{1,2,\ldots,\ell} := g^{b_0}$, and $\bar{y}_0 := g^{b_0}$, $\bar{y}_1 := g^{b_0^2}, \ldots$, $\bar{y}_{t-u-1} := g^{b_0^{2^{t-u-1}}}$. Observe that even though it is highly unlikely that the public key $\bar{y}_{t-u}$ is properly constructed when $t > u$ (that is, it is highly unlikely that $\bar{y}_{t-u} = \mathrm{DH}(\bar{y}_{t-u-1}, \bar{y}_{t-u-1})$, or that $g^{\bar{\sigma}_i^{2^t}} = g^{\bar{s}_\ell^{2^u}}$), it is hard for **A** to recognize this, if the symmetric Diffie-Hellman decision problem is hard.

Give as input to **A**: $p, g, \bar{z}_1, \bar{z}_2, \ldots, \bar{z}_\ell, \bar{z}_{1,2}, \ldots, \bar{z}_{1,2,\ldots,\ell}$, the shares $\bar{s}_1, \ldots, \bar{s}_{\ell-1}$, the public keys $\bar{y}_0 = g^b$, $\bar{y}_1 = g^{b^2}$, $\ldots$, $\bar{y}_t = g^{b^{2^u}}$, Bob's long term key, together with the certificates $(p, g, \bar{y}_i, \mathrm{sign}_{\mathrm{Bob}'}(p, g, \bar{y}_i))$, $i = 1, 2, \ldots, \ell$, and an "old" ciphertext $(\bar{w}_1, \bar{w}_2)$ encrypted at time $t-u$, with $\bar{w}_1 = g^{\bar{r}}$, $\bar{r} \in_R Z_{p-1}^*$, and

$\bar{w}_2 \in_R Z_p^*$. Algorithm **A** will output a message $\bar{m}$ such that $\bar{z}_2/\bar{m} = g^{\bar{r}d}$, where d is a 2^u–th root of b^{2^u} which is a quadratic residue in $Z_{p_1 p_2}$. However b was chosen at random in Z_{p-1}^*, so that with probability $3/4$ we get that $b \bmod p_1 p_2$ is not a residue in $Z_{p_1 p_2}$. Then with probability one half, $b - d$ is either a multiple of p_1 or a multiple of p_2. This means that $g^{\bar{r}d}/g^{\bar{r}b} = g^{\bar{r}(d-b)}$ has order p_1 or p_2. Consequently **A** can find an element in Z_p of large order.

Theorem 2. (Privacy) *A wire-tapper may try to decipher the ciphertext. This is as hard as the Diffie-Hellman problem.*

Proof. (Sketch) We show this by using the approach in [13]. Suppose that **B** is a polynomial time algorithm which on input: $p, g, z_1, z_2, \ldots, z_\ell$, $z_{1,2}$, $z_{1,2,3}$, $\ldots$, $z_{1,2,\ldots,\ell}$, the certificates $(p, g, \bar{y}_i, \text{sign}_{\text{Bob}'}(p, g, \bar{y}_i))$, $i = 1, 2, \ldots, \ell$, and the ciphertext $(\bar{w}_1, \bar{w}_2)$, $\bar{w}_1 = g^r$,$\bar{w}_2 = m\bar{y_t}^r$, will output m. Let $p, g, \bar{y}_t, \bar{w}_1$ be an instance of the the Diffie-Hellman problem. Construct $\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_\ell$, $\bar{z}_{1,2}, , \ldots, \bar{z}_{1,2,\ldots\ell}$, $\bar{y}_0, \bar{y}_1, \ldots, \bar{y}_t \in Z_p^*$, as in the previous case. Give this as input to **B** together with $(\bar{w}_1, \bar{w}_2)$, to get a "message" $\bar{m}$ such that $\bar{w}_2/\bar{m} = \text{DH}(\bar{y}_t, \bar{w}_1)$ $(= \bar{y}_t^{\bar{r}})$.
The rest can all be simulated because we have used zero-knowledge proofs.

2.1 Generalizations

Generalizing time controlled l-out-of-l key escrow systems to l'-out-of-l systems, is straightforward when using more complex secret sharing schemes over $Z_{p-1}^*(*)$. Secret sharing schemes that could be used for this purpose can be found in [15, 12, 2, 5], when using techniques such as those described in [17, 13]. Robustness can be achieved by using, for example [20, 19].

Other properties such as proactive secret sharing can also be achieved using [21, 18, 27].

3 Time controlled auctions with closed bids

We first consider a basic (additive) ℓ-out-of-ℓ escrow system, using a simple setting. Generalizations will be discussed later.

Our system uses a Discrete Logarithm setting with composite modulus $n = p_1 p_2$, where p_1, p_2 are appropriate large primes. The bidder, Alice, chooses n and $g_1, g_2 \in Z_n$ such that g_1 has large order whereas g_2 has a rather small prime order q. Alice has two public keys for encryption: $y_1 = g_1^{a_1} \bmod n$, $y_2 = g_2^{a_2} \bmod n$, where $a_1 \in_R Z_{\phi(n)}$, $a_2 \in_R Z_q$. The private key a_1 is shared among ℓ escrow agents EA_i, $i = 1, 2, \ldots, \ell$. The other is not shared. For this system the public key y_2 is weak and must be used *only once*. This key must be such that it can be recovered by an exhaustive search of the key space, but the time taken for this search should not be too short.[4]

[4] Since an exhaustive search is parallelizable, some kind of inherently sequential scheme may be used, such as the *time-lock puzzles* proposed in [28]. Our protocol can easily be adapted to allow for such schemes.

Alice "double" encrypts her contract bid m by using the keys y_1, y_2. Let $\mathrm{ElG}^2(m)$ be the encryption. Alice sends this to the receiving agent Bob. At completion she will reveal both secret keys a_1, a_2, from which Bob will get the tendered bid m. If Alice refuses to reveal these keys, then Bob informs the escrow agents who will enable a first decryption. This will make it possible for Bob to get an encryption $\mathrm{ElG}(m)$ of m with private key a_2. Bob then initiates a procedure to recover m, by exhaustively breaking this encryption. Bob can achieve this because the second key is relatively weak. A similar argument applies if a Court order is issued to the escrow agents to enable the decryption of $\mathrm{ElG}^2(m)$. The security issues of this protocol will be discussed in more detail later. We first describe the protocol more formally.

Setting

The parties involved: the bidder Alice, the receiving officer Bob, a Court, the Law Enforcement Agency LEA, and the Escrow Agents EA_i, $i = 1, 2, \ldots, \ell$.
The parameters: Both Alice and Bob have long term public keys which are known to each other. These keys are used for authentication (signing).
A Discrete Logarithm setting is used with a composite modulus n. Alice chooses $n = p_1 p_2$, a product of two large primes p_1, p_2, with $p_1 - 1 = 2qq_1$, $p_2 - 1 = 2qq_2$, q_1, q_2 primes, and q a rather small prime (say 140 bits).
Alice chooses $g_1 \in_R Z_n$ and $g_2 \in Z_n$ such that $\mathrm{ord}(g_2 \bmod p_1) = \mathrm{ord}(g_2 \bmod p_2) = q$. Here $\mathrm{ord}(g_2 \bmod p_1)$ is the order of g_2 in Z_{p_1} and $\mathrm{ord}(g_2 \bmod p_2)$ is the order of g_2 in Z_{p_2}. Consequently g_2 has order q in Z_n.

Set-up

Alice chooses $a_1 \in_R Z_{\phi(n)}$ and $a_2 \in_R Z_q$. The public key of Alice is $(n, q, g_1, g_2, y_1, y_2)$, where $y_1 := {g_1}^{a_1} \bmod n$, $y_2 := {g_2}^{a_2} \bmod n$.
Alice finds ℓ shares of a_1, by choosing exponents $s_i \in_R Z_{\phi(n)}$ for $i = 1, 2, \ldots, \ell - 1$, and taking $s_\ell = a_1 - (s_1 + s_2 + \ldots + s_{\ell-1}) \bmod \phi(n)$.

1. Alice gives privately to each agent EA_i, $i = 1, 2, \ldots, \ell$, the share s_i.
2. Alice publishes $z_1 := g^{s_1}, z_2 := g^{s_2}, \ldots, z_\ell := g^{s_\ell}$. Each agent EA_i checks that $z_i = g^{s_i}$ and reports failure to the LEA. The LEA checks that $g^a = z_1 \cdot z_2 \cdots z_\ell$. If any of the checks of the EA_i's fails or if the LEA's check fails then Alice has cheated, the tender is rejected, and appropriate actions are taken.

Sending an encrypted contract bid

1. Alice sends Bob the pair of her public keys authenticated with her long term key,
$$(n, q, g_1, g_2, y_1, y_2, \mathrm{sign}_{\mathrm{Alice}}(g_1, g_2, n, q, y_1, y_2)),$$
and the encrypted bid $\mathrm{ElG}^2(m) = (g_1^{r_1}, g_2^{r_2}, m y_1^{r_1} y_2^{r_2})$, where $m \in Z_n^*$ is the bid and $r_1, r_2 \in_R Z_n$.

2. If the parameters are in the appropriate fields, with q a small prime, if the order of g_2 is q, and if Alice's public keys are authenticated properly, then Bob accepts the tender and sends Alice a receipt $\text{sign}_{\text{Bob}}(\text{Alice}, \text{ElG}^2(m))$.

Opening a tender

When the tender is due to be opened, Alice sends Bob the private keys a_1, a_2. Bob checks these for correctness. If correct, $\text{ElG}^2(m)$ is decrypted to get the bid m, which is validated.
If Alice refuses to send her keys, the LEA is informed and initiates a procedure to recover m.

The Court recovers the bid

If the Court has issued an order to recover the bid, the LEA will wire-tap the communication and send g^{r_1} to the escrow agents who will compute $y_1^{r_1}$. From this the LEA can get $\text{ElG}(m) = ({g_2}^{r_2}, m{y_2}^{r_2})$. The key for this ciphertext is weak, so the LEA can recover m by brutal force. However, q has to be sufficiently large to prevent a conspiracy, as explained further on.

Security

The security of this system relies on the difficulty of factoring a number $n = p_1p_2$, p_1, p_2 primes, when a particular number $g \in Z_n$ is given, with a rather small prime order q. It is important that both $g \bmod p_1 \neq 1$ and $g \bmod p_2 \neq 1$. Otherwise, if say $g \bmod p_1 = 1$, then p_1 is a factor of $g - 1$ and it becomes easy to factor n by taking the $gcd(n, g-1)$. Observe that for $g = n - 1$ we have $q = 2$, but this trivial case is too small to be of any use for us.

Fair auction bidding

Alice may refuse to open her bid, on completion. Bob will inform the LEA and the Court will authorize the escrow agents to decrypt the ciphertext. The escrow agents will compute $y_1^{r_1}$ from which the LEA will get $\text{ElG}(m) = ({g_2}^{r_2}, m{y_2}^{r_2})$. The key for this ciphertext is weak, so the LEA can initiate a procedure to recover m by brutal force. (Note that q has to be sufficiently large, as we now explain.)

Conspiracy

The agents may be corrupted by the bidding officer Bob. They will recover $\text{ElG}(m) = ({g_2}^{r_2}, m{y_2}^{r_2})$, but if the key y_2 is not too weak they will not be able to recover the message in time. For this reason q cannot be too small.

Theorem 3. (Privacy) *A wire-tapper may try to decipher the bid m. This is as hard as breaking the Diffie-Hellman problem.*

Proof. (Sketch) Suppose that $\mathbf{A}$ is a polynomial time algorithm which on input: n, q, g_1, g_2, y_1, y_2, authenticated with Alice's long term key, $z_1, z_2, \ldots, z_\ell$, and

$(g_1^{r_1}, g_2^{r_2}, m{y_1}^{r_1}{y_2}^{r_2})$, will output m. Let $n, g_1, \bar{y}_1, {g_1}^{\bar{r}_1}$ be an instance of the Diffie-Hellman problem as in [13]. Take $\bar{s}_1, \ldots, \bar{s}_{\ell-1} \in_R Z_n$, and $\bar{s} = \bar{s}_1 + \ldots + \bar{s}_{\ell-1}$. Then let $\bar{z}_1 = g^{\bar{s}_1}$, …, $\bar{z}_{\ell-1} = g^{\bar{s}_{\ell-1}}$ and $\bar{z}_\ell = \bar{y}_1 g_1^{-\bar{s}}$. Find an appropriate long term key for Alice. Finally take $\bar{r}_2 \in_R Z_q$ and compute ${g_2}^{\bar{r}_2}$ and ${y_2}^{\bar{r}_2}$.

Give as input to **A**: $n, q, g_1, g_2, \bar{y}_1, \bar{y}_2$, authenticated with Alice's public key, $\bar{z}_1, \ldots, \bar{z}_\ell$, and $(g_1^{\bar{r}_1}, g_2^{\bar{r}_2}, \bar{w})$, where $\bar{w} \in_R Z_n$. Algorithm **A** will output $\bar{m}$, such that $\bar{w}/\bar{m} = \bar{y}_1^{\bar{r}_1}\bar{y}_2^{\bar{r}_2}$, from which we get $\mathrm{DH}(\bar{y}_1, {g_1}^{\bar{r}_1}) = \bar{y}_1^{\bar{r}_1}$.

Generalizations

Similar generalizations to those in Section 2.1 apply. (Although $\phi(n)$ is not public, techniques similar to those in [17, 13] will address this problem.)

Acknowledgement

The authors would like to thank an anonymous referee for pointing out the references [23, 28].

References

1. Adleman, L.M., McCurley K.S.: Open Problems in Number Theoretic Complexity. In: Johnson, D., Nishizeki, T., Nozaki, A., Wilf, H. (eds): Discrete Algorithms and Complexity, Proceedings of the Japan-US Joint Seminar (Perspective in Computing series, **15**. Academic Press Inc., Orlando, Florida (1986) 263–286
2. Alon, N., Galil, Z., Yung, M.: Efficient dynamic-resharing "verifiable secret sharing" against mobile adversary. In: Spirakis, P.G. (ed): Algorithms – ESA '95, Third Annual European Symposium, Proceedings (Lecture Notes in Computer Science 979). Springer-Verlag (1995) 523–537
3. Bellare, M., Goldwasser, S.: Verifiable partial key escrow. Proc. 4th ACM Conference on Computer and Communications Security (1997)
4. Beth, T.: Zur Sicherheit der Informationstechnik. Informatik-Spektrum, **13** (1990) 204–215
5. Blackburn, S.R., Burmester, M., Desmedt, Y., Wild, P.R.: Efficient multiplicative sharing schemes. In: Maurer, U. (ed): Advances in Cryptology – Eurocrypt '96, Proceedings (Lecture Notes in Computer Science 1070). Springer-Verlag (1996) 107–118
6. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM J. Comput. **15**(2) (1986) 364–383
7. Boyd, C.: Digital multisignatures. In: Beker, H., Piper, F. (eds): Cryptography and coding. Clarendon Press (1989) 241–246
8. Chaum, D.: Security without identification: transaction systems to make Big Brother obsolete. Commun. ACM, **28**(10) (1985) 1030–1044
9. Chaum, D.: Zero-knowledge undeniable signatures. In: Damgård, I. (ed): Advances in Cryptology, Proc. of Eurocrypt '90 (Lecture Notes in Computer Science 473). Springer-Verlag (1991) 458–464
10. Clipper. A proposed federal information processing standard for an escrowed encryption standard (EES). Federal Register, July 30, 1993.

11. Denning, D.E., Branstad, D.K.: A taxonomy of key escrow encryption systems. Communications of the ACM, **39**(3), (1996) 24–40
12. Desmedt, Y., Di Crescenzo, G., Burmester, M.: Multiplicative non-abelian sharing schemes and their application to threshold cryptography. In: Pieprzyk, J., Safavi-Naini, R. (eds): Advances in Cryptology – Asiacrypt '94, Proceedings (Lecture Notes in Computer Science 917). Springer-Verlag (1995) 21–32
13. De Santis, A., Desmedt, Y., Frankel, Y., Yung M.: How to Share a Function Securely. Proceedings of the twenty-sixth annual ACM Symp. Theory of Computing (STOC) (1994) 522–533
14. Desmedt, Y., Frankel, Y.: Threshold cryptosystems In: Brassard, G. (ed): Advances in Cryptology – Crypto '89, Proceedings (Lecture Notes in Computer Science #435). Springer-Verlag (1990) 307–315
15. Desmedt, Y.G., Frankel, Y.: Homomorphic zero-knowledge threshold schemes over any finite abelian group. SIAM Journal on Discrete Mathematics **7**(4) (1994) 667–679
16. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Trans. Inform. Theory, **IT–22**(6) (1976) 644–654
17. Frankel, Y., Desmedt, Y.: Parallel reliable threshold multisignature. Tech. Report TR–92–04–02, Dept. of EE & CS, Univ. of Wisconsin–Milwaukee, April 1992. ftp://ftp.cs.uwm.edu/pub/tech_reports/desmedt-rsa-threshold_92.ps.
18. Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Proactive RSA. In: Kaliski, B.S. (ed): Advances in Cryptology – Crypto '97, Proceedings (Lecture Notes in Computer Science 1294). Springer-Verlag (1997) 440–454
19. Frankel, Y., Gemmell, P., Yung, M.: Witness-based cryptographic program checking and robust function sharing. Proceedings of the Twenty-Eighth Annual ACM Symp. on Theory of Computing (1996) 499–508
20. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust and efficient sharing of RSA functions. In: Koblitz, N. (ed): Advances in Cryptology – Crypto '96, Proceedings (Lecture Notes in Computer Science 1109). Springer-Verlag (1996) 157–172
21. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive secret sharing. In: Coppersmith, D. (ed): Advances in Cryptology – Crypto '95, Proceedings (Lecture Notes in Computer Science #963). Springer-Verlag (1995) 339–352
22. Kilian, J., Leighton, T.: Failsafe key escrow, revisited. In: Coppersmith, D. (ed): Advances in Cryptology – Crypto '95, Proceedings (Lecture Notes in Computer Science #963). Springer-Verlag (1995) 208–221
23. Maurer, U.M., Wolf, Y.: Diffie-Hellman Oracles. In:. Kobliz, N. (ed): Advances in Cryptology – Crypto '96, Proceedings (Lecture Notes in Computer Science 1109). Springer-Verlag (1996) 268–282
24. Micali, S.: Fair public-key cryptosystems. In: Brickell, E.F. (ed): Advances in Cryptology – Crypto '92, Proceedings (Lecture Notes in Computer Science 740). Springer-Verlag (1993) 113–138
25. Ostrovsky, R., Yung, M.: How to Withstand Mobile Virus Attacks. Proceedings of the 10-th Annual ACM Symp. on Principles of Distributed Computing (1991) 51–60
26. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Davies, D.W. (ed): Advances in Cryptology, Proc. of Eurocrypt '91 (Lecture Notes in Com-

puter Science #547). Springer-Verlag (1991) 522–526

27. Rabin, T.: A simplified approach to threshold and proactive RSA. To appear in the Proceedings of Crypto '98.
28. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and time-release Crypto. http://theory.lcs.mit.edu/~rivest/publications.html (to appear).
29. Simmons, G.J., June 22–24, 1994. Observation made at the Workshop on Key Escrow.

A A zero-knowledge proof for the Diffie-Hellman problem

The zero-knowledge proof for the Diffie-Hellman problem in [9] is not adequate for our purpose, since it designed for the case when the group of the exponents has prime order. In our case the group of exponents has composite order. The following protocol will serve our purpose.

A zero-knowledge interactive proof for the Diffie-Hellman problem

Input: A prime p, $g \in Z_p$ of large order, $\alpha = g^a \bmod p$, $\beta = g^b \bmod p$, $\gamma = g^{ab} \bmod p$.

Repeat independently $t = \log p$ times the following subroutine:

1. The Prover selects exponents $x, y \in_R Z_{p-1}$ and sends to the Verifier: $\kappa_x = g^x \bmod p$, $\kappa_y = g^y \bmod p$, $\kappa_{xy} = g^{xy} \bmod p$, $\kappa_{ay} = g^{ay} \bmod p$, and $\kappa_{bx} = g^{bx} \bmod p$.
2. The Verifier sends the Prover a query bit $e \in_R \{0,1\}$.
3. If $e = 0$ the Prover sends x, y to the Verifier, and the Verifier checks that: $\kappa_x = g^x \bmod p$, $\kappa_y = g^y \bmod p$, $\kappa_{xy} = g^{xy} \bmod p$, $\kappa_{ay} = \alpha^y \bmod p$, and $\kappa_{bx} = \beta^x \bmod p$.
   If $e = 1$ the Prover sends $a' = a + x \bmod (p-1)$, $b' = b + y \bmod (p-1)$ to the Verifier who checks that: $g^{a'} \equiv \alpha \cdot \kappa_x \,(\bmod p)$, $g^{b'} \equiv \beta \cdot \kappa_y \,(\bmod p)$, and $g^{a'b'} \equiv \gamma \cdot \kappa_{ay} \cdot \kappa_{bx} \cdot \kappa_{xy} \,(\bmod p)$.
   If any of the checks fails, the Verifier halts and rejects the proof.

The Verifier accepts the proof of the Prover if all t rounds have been completed successfully.

Let $L = \{(p, \alpha, \beta, \gamma) \mid p \text{ prime}, \alpha = g^a \bmod p, \beta = g^b \bmod p, \gamma = g^{ab} \bmod p\}$. Then,

Theorem 4. *The protocol above is a perfect zero-knowledge proof of membership in L.*

Proof. (Sketch)
Completeness: Obvious.
Soundness: If the Prover can answer the queries $e = 0, 1$ then there exist a, b, x, y

such that $a' = a + x \bmod (p-1)$, $b' = b + y \bmod (p-1)$, with $\kappa_{ay} = \alpha^y \bmod p$, $\kappa_{bx} = \beta^x \bmod p$, $\kappa_{xy} = g^{xy} \bmod p$, and $\gamma \equiv g^{a'b'} \cdot \kappa_{ay}^{-1} \cdot \kappa_{bx}^{-1} \cdot \kappa_{xy}^{-1} \equiv g^{ab} \pmod{p}$.
Zero-knowledge: Let α, β, γ be given. Pick $a', b' \in_R Z_{p-1}$, and let $\kappa_x = g^{a'}/\alpha \bmod p$, $\kappa_y = g^{b'}/\beta \bmod p$. Then solve $\alpha^{b'} \equiv \gamma \cdot \kappa_{ay} \pmod{p}$, $\beta^{a'} \equiv \gamma \cdot \kappa_{bx} \pmod{p}$, $g^{a'b'} \equiv \gamma \cdot \kappa_{ay} \cdot \kappa_{bx} \cdot \kappa_{xy} \pmod{p}$, for the unknowns κ_{ay}, κ_{bx}, κ_{xy}, respectively.

This article was processed using the LaTeX macro package with LLNCS style