Equitable Key Escrow with Limited Time-Span (or, How to Enforce Time Expiration Cryptographically)

Mike Burmester, Yvo G. Desmedt, and Jennifer Seberry

Abstract— **With equitable key escrow mechanisms 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: limited time-span escrow and limited time-span 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/her closed bid beyond the bidding period. We propose two protocols, one for each application. These protocols do not require the use of tamper-proof devices. Furthermore, our limited time-span escrow scheme is proven secure (the security is reduced to two problems which are believed to be hard).**

Keywords— **Key escrow, auctions with closed bids, time stamps, complexity theory.**

I. Introduction

Key escrow has been proposed as a mechanism to protect society from individuals who use a communication system for criminal purposes [1], [2], [3] (an excellent survey of key escrow systems is given by D.E. Denning and D.K. Branstad in [4]). 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 [7]), which may, in certain circumstances, abuse it. In a society oriented key escrow system this power must be equally shared between the individual and society (for an analysis of fair cryptosystems see [2], [8]). Furthermore it must have a limited life span (see *e.g.* [5], [6]). Indeed a major objection to currently proposed key escrow schemes (*e.g.*, the Clipper [3]) 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 wiretapped 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[9] scheme protects a majority against Big Brother, it does not protect a minority. For example, an extremist group seeking to take control of the government may wiretap all communication of suspect dissidents, and decrypt it when it takes 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 key 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 key. This scheme is proven secure (the security is reduced to two problems which are believed to be hard).

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 (an interesting discussion on weak encryption is presented in [10]). 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.

Our goal in this paper is to design protocols which achieve equitable key escrow. The organization of the paper is as follows. In the rest of this section we discuss our model for escrow mechanisms and the failures we focus on. In Section II we present a basic protocol l-out-of-l limited time-span escrow system and discuss its security. This is then generalized to a k-out-of-l system in Section III. In Section IV we present a protocol for limited time-span auctions with closed bids.

M. Burmester is with the Information Security Group at Royal Holloway – University of London, Egham, Surrey TW20 OEX, UK. E-mail: m.burmester@rhbnc.ac.uk

Y. Desmedt is with the Center for Cryptography, Computer and Network Security, and at the Department of EE & CS, University of Wisconsin – Milwaukee, P.O. Box 784, WI 53201 Milwaukee,USA, and the Department of Mathematics, Royal Holloway – University of London, Egham, Surrey TW20 OEX, UK. E-mail: desmedt@cs.uwm.edu

J. Seberry is with the Center for Computer Security Research at the University of Wollongong, Australia. E-mail: j.seberry@uow.edu.au

A. 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 other selections.

The Diffie-Hellman [11] 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 *squaring* Diffie-Hellman problem [12] 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 [12, Theorem 2]. The problem of deciding whether $z = \mathrm{DH}(g^x, g^x)$, given z, g^x is the squaring Diffie-Hellman *decision* problem. If this problem is hard then so are the Diffie-Hellman problem, the Diffie-Hellman decision problem and the squaring Diffie-Hellman 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 [13], and is considered to be hard.

B. Model

We use the usual model for key escrow mechanisms (see *e.g.* [14], [8], [4]), in which the communication is encrypted with the encryption key escrowed to trusted agents. The escrow agents get shares of the encryption key and are trusted to safeguard their shares and to enable decryption when instructed by a law enforcement agency. The law enforcement agency and the escrow agents are trusted to adhere to the escrow protocol.

We shall focus on two types of system failure. In the first the adversary takes control of both the law enforcement agency and the escrow agents. Then we have total failure and the escrow mechanism does not serve any further useful purpose, at least as far as the honest parties are concerned. For limited time-span escrow we require that all messages encrypted prior to the last time update should be protected. That is we require *forward secrecy* [15].

One may argue that this feature is not really desirable for key escrow [16], since it makes it possible for criminals to exchange secret keys (*e.g.*, a one-time pad) through the escrow mechanism and then (after the next time update) pre-encrypt messages with perfect forward secrecy.

The problem here is not so much the pre-encryption of messages, which is possible with any key escrow system [4], but the fact that the escrow mechanism itself has been used to exchange pre-encryption keys. This weakness, however, is common to all escrow mechanisms, whether these have limited time-span or not. Indeed if the pre-encryption keys are exchanged before a court order for wiretapping is issued, then the law enforcement agency will not have access to them because it is not authorized to wiretap. In this context it is important to note that honest parties (*e.g.* the law enforcement agency) are bound by the rules of the escrow mechanism whereas dishonest parties are not. Escrow mechanisms with limited time-span will protect honest users from abuses by dishonest users. They will not prevent dishonest users from abusing the mechanism for their own purpose. A *determined* criminal can always bypass the escrow mechanism.

One may consider using tamper-proof trusted agents in limited time-span escrow to prevent abuses, by having the legitimacy of the court order chained through well certified procedures (*e.g.* each democratically elected government is authorized by the previous government) [16]. This should make it difficult for non-constitutional agents to enable decryption. There are two problems with this argument. The first is that it relies on tamper-proof devices which are not really secure, the second that it is based on a weak adversarial scenario. One cannot exclude the possibility that the adversary will take control by using constitutional means (a scenario strongly supported by recent history).

The goal of key escrow is to protect the confidentiality of honest user, while revoking it from the criminal *as far as this is possible*. Obviously there is a tradeoff between the requirements of honest users and the controls for criminals. However tightening the controls of the latter will only restrict the flexibility and rights of the former. As with all democratic processes the emphasis should be on the requirements of the honest users, not the controls for the criminal *at any cost to freedom* in society.

The second type of failure of escrow mechanisms concerns failure by some of the escrow agents (Section III-D and Section IV-B). This may disable the decryption procedure. We present equitable escow schemes which achieve robustness.

There are several other types of failure which make it possible for the adversary to bypass the escrow mechanism. We do not discuss these here because, as pointed out earlier, bypassing the system is an inherent weakness of escrow mechanisms (for the determined criminal), and it is therefore futile to try to prevent such attacks.

II. Limited time-span escrow: the basic protocol

For simplicity we start with a basic l-out-of-l escrow system. We will discuss generalizations to other access structures in the following section.

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, $i = 1, 2, \ldots, l$.

In our basic model each agent gets a share $s_i \in_R Z_{p-1}$ ($i = 1, 2, \ldots, l-1$), and s_l is such that $s_1 \cdot s_2 \cdots s_l = a \bmod (p-1)$. 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 $a^{2^t} = (a^{2^{t-1}})^2 \bmod (p-1)$, the shares are updated to ${s_i}^{2^t} = ({s_i}^{2^{t-1}})^2 \bmod (p-1)$, and the public key is updated to $y_t = g^{a^{2^t}} = \mathrm{DH}(g^{a^{2^{t-1}}}, g^{a^{2^{t-1}}})$. The agents EA_i compute the new shares by themselves, and *must* destroy the old shares $({s_i}^{2^{t-1}} \bmod (p-1))$. As a consequence, the escrow agents cannot enable the decryption of 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 (even if $(l-1)$ escrow agents did not delete their old shares) is related to two problems: the problem of finding elements of large order in Z_p and the squaring Diffie-Hellman decision problem. Both problems are believed to be hard (*cf.* [13], [12]).

We now describe our basic protocol in more detail. For this purpose we combine the multiplicative threshold scheme of Boyd [17], the ElGamal [18] threshold scheme of Desmedt–Frankel [19] and add time dependency using ideas from Blum-Blum-Shub [20]. For verification we adapt Pedersen scheme [21].

A. The protocol

A.1 Setting

The parties involved: the sender Alice, the receiver Bob, a Court, the Law Enforcement Agency LEA, and the Escrow Agents EA_i, $i = 1, 2, \ldots, l$.

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 $(-1 \mid p_1) = (-1 \mid p_2) = -1$ ($p_1 p_2$ is a Blum integer [20]), and an element $g \in Z_p$ whose order is $p_1 p_2$. Bob publishes p, g. The number l of agents must be polynomially bounded in the length of p.

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.

A.2 Set-up

Set $\mathit{time} := 0$.

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

Then,

1. Bob gives privately to each agent EA_i, $i = 1, 2, \ldots, l$, the share s_i.
2. Bob publishes $z_1 := g^{s_1}, z_2 := g^{s_2}, \ldots, z_l := g^{s_l}$, 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 $z_{1,2} := g^{s_1 s_2}$, $z_{1,2,3} := g^{s_1 s_2 s_3}, \ldots,$ $z_{1,2,\ldots,l} := g^{s_1 s_2 \cdots s_l} = (y_0)$, 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 $k = 2, \ldots, l$, by using an interactive zero-knowledge proof for the Diffie-Hellman problem – an example of such a proof is given in Appendix IV-B. If any of the proofs fails, then Bob has cheated and is reported to the LEA.
3. Set $\mathit{time} = 1$.

A.3 Updating

At $\mathit{time} = t$

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 a zero-knowledge proof for the Diffie-Hellman problem (Appendix IV-B). That is, Bob proves that $y_t = \mathrm{DH}(y_{t-1}, y_{t-1})$.

Remark 1: The updated shares ${s_i}^{2^t} \bmod (p-1)$ are shares of the updated key $a^{2^t} \bmod (p-1)$. This property follows from the fact that the mapping $x \to x^2$ of $Z^*_{p-1}(\cdot)$ is a (one-way) homomorphism which extends to a secret sharing homomorphism [22].

A.4 Getting an escrowed key and decrypting

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 wiretap the communication and send g^r to agent EA_1. The agents $\mathrm{EA}_1, \mathrm{EA}_2, \ldots, \mathrm{EA}_l$ then compute ${y_t}^r$ sequentially as follows: for $i < l$, each EA_i on receiving $g^{r \prod_{j=1}^{i-1} s_j^{2^t}}$ computes,
$$g^{r \prod_{j=1}^{i} s_j^{2^t}} := \left(g^{r \prod_{j=1}^{i-1} s_j^{2^t}}\right)^{s_i^{2^t}}, \qquad (1)$$
   which it sends to EA_{i+1}. Agent EA_l then computes $\left(g^{r \prod_{j=1}^{l-1} s_j^{2^t}}\right)^{s_l^{2^t}}$, which it sends to the LEA. Since this corresponds to the decrypting factor ${y_t}^r$, the LEA can decrypt the ciphertext.

Remark 2: The computation in Step 4 of the decrypting factor ${y_t}^r$ cannot be executed in parallel since this would involve solving the Diffie-Hellman problem.

B. Security

Theorem 1: **(Irreversible time)** If the squaring 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: We first note that, to date, the best algorithm to find elements of large order in Z_p is to first factor $(p-1)$, then find a generator g for Z^*_p (using the factorization of $(p-1)$) and finally raise g to an appropriate power. In our case the factorization of $(p-1)$ is not known.

Suppose that there is a polynomial time algorithm $\mathbf{A}$ which on input $p, g, z_1, z_2, \ldots, z_l$, $z_{1,2}$, $z_{1,2,3}$, $\ldots$, $z_{1,2,\ldots,l}$, the current shares[1] $s_1^{2^t}, s_2^{2^t}, \ldots, s_l^{2^t}$, the old shares of $(l-1)$ corrupted escrow agents, Bob's long term public key, the certificates $(p, g, y_j, \mathrm{sign}_{\mathrm{Bob}}(p, g, y_j))$, $j = 1, 2, \ldots, t$, 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 $\mathbf{A}$ can be used to compute $g^{r a^{2^{t-u}}}$ $(= w_2/m)$. We now will use $\mathbf{A}$ to get an element in Z_p of large order. First we prepare a (simulated) input for $\mathbf{A}$.

Note that $0 < u < t$ (since the public key y_0 is never used to encrypt). Assume that the dishonest escrow agents are $i_1, i_2, \ldots, i_{l-1}$ and let i_l be the honest escrow agent. Find an appropriate long term secret key for Bob. Choose $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_l \in_R Z_{p-1}^*$. Compute $b_0 = \bar{s}_1 \bar{s}_2 \cdots \bar{s}_l$, $\bar{z}_1 := g^{\bar{s}_1}$, $\bar{z}_2 := g^{\bar{s}_2}$, $\ldots$, $\bar{z}_l := g^{\bar{s}_l}$, $\bar{z}_{1,2} := g^{\bar{s}_1 \bar{s}_2}$, $\ldots$, $\bar{z}_{1,2,\ldots,l} := g^{\bar{s}_1 \cdots \bar{s}_l}$, and $\bar{y}_0 := g^{b_0}$, $\bar{y}_1 := g^{b_0^2}, \ldots, \bar{y}_{t-u-1} := g^{b_0^{2^{t-u-1}}}$.

Take $b \in_R Z_{p-1}^*$ and compute $\bar{s'}_{i_1} := \bar{s}_{i_1}^{2^{t-u}}$, $\bar{s'}_{i_2} := \bar{s}_{i_2}^{2^{t-u}}$, $\ldots$, $\bar{s'}_{i_{l-1}} := \bar{s}_{i_{l-1}}^{2^{t-u}}$. Compute $\bar{s} := \bar{s'}_{i_1} \cdot \bar{s'}_{i_2} \cdots \bar{s'}_{i_{l-1}}$, $\bar{s'}_{i_l} := b \cdot (\bar{s})^{-1}$, and the public keys $\bar{y}_{t-u} = g^b$, $\bar{y}_{t-u+1} = g^{b^2}$, $\ldots$, $\bar{y}_t = g^{b^{2^u}}$. Observe that even though it is highly unlikely that the public key $\bar{y}_{t-u}$ is properly constructed (that is, $\bar{y}_{t-u} = \mathrm{DH}(\bar{y}_{t-u-1}, \bar{y}_{t-u-1})$, it is hard for $\mathbf{A}$ to recognize this, if the squaring Diffie-Hellman decision problem is hard.

Give as input to $\mathbf{A}$: $p, g, \bar{z}_1, \bar{z}_2, \ldots, \bar{z}_l$, $\bar{z}_{1,2}, \ldots, \bar{z}_{1,2,\ldots,l}$, the "shares" $\bar{s'}_1^{2^u}, \ldots, \bar{s'}_l^{2^u}$, the "old shares" $\bar{s'}_{i_1}, \ldots, \bar{s'}_{i_{l-1}}$, Bob's long term public key, the certificates $(p, g, \bar{y}_j, \mathrm{sign}_{\mathrm{Bob}}(p, g, \bar{y}_j))$, $j = 1, 2, \ldots, t$, 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 $\mathbf{A}$ will output a message $\bar{m}$ such that $\bar{w}_2/\bar{m} = g^{\bar{r}d}$, 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 have that $b \bmod p_1 p_2$ is not a residue in $Z_{p_1 p_2}$. (Indeed for $t > 1$, a^{2^t} 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 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 $\mathbf{A}$ can find an element in Z_p of large order.

Note that the view of the communication between the escrow agents when computing the partial factors (1) can also be included in the simulation. Indeed a (simulated) view consists of $g^{\bar{r}}$, $g^{\bar{r} \bar{s'}_1^{2^u}}$, $\ldots$, $\bar{y}_t^{\bar{r}} = g^{\bar{r} \bar{s'}_1^{2^u} \cdots \bar{s'}_l^{2^u}}$, which can be computed from $g^{\bar{r}}$ and $\bar{s}'_1, \ldots, \bar{s}'_l$. Moreover if this communication is encrypted, then these encryptions can also be simulated. ∎

Theorem 2: **(Privacy)** A wiretapper with the help of up to $(l-1)$ escrow agents may try to decipher the ciphertext. This is as hard as the Diffie-Hellman problem.

Proof: We use the approach in [23]. Let $\mathbf{B}$ be a polynomial time algorithm which on input: $p, g, z_1, z_2, \ldots, z_l$, $z_{1,2}$, $z_{1,2,3}$, $\ldots$, $z_{1,2,\ldots,l}$, the certificates $(p, g, y_j, \mathrm{sign}_{\mathrm{Bob}'}(p, g, y_j))$, $j = 1, 2, \ldots, t$, and the ciphertext (w_1, w_2), $w_1 = g^r$, $w_2 = m {y_t}^r$, will output m. We shall show that $\mathbf{B}$ can be used to solve the Diffie-Hellman problem. Let $p, g, \bar{y}_t, \bar{w}_1$ be an instance of the the Diffie-Hellman problem. Construct $\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_l$, $\bar{z}_{1,2}, \ldots, \bar{z}_{1,2,\ldots,l}$, $\bar{y}_0, \bar{y}_1, \ldots, \bar{y}_{t-1} \in Z_p^*$, as in Theorem 1. Give this as input to $\mathbf{B}$ together with $\bar{y}_t, \bar{w}_1$ and $\bar{w}_2 \in_R Z_p^*$, a long term public key for Bob and appropriate certificates, to get a "message" $\bar{m}$ such that $\bar{w}_2/\bar{m} = \mathrm{DH}(\bar{y}_t, \bar{w}_1)$ $(= \bar{y}_t^{\bar{r}})$.
The interactive proofs of Bob can be simulated because we have used zero-knowledge proofs. ∎

III. Limited time-span escrow: the general case

A. Limited time-span escrow based on perfect hash families

There are several secret sharing schemes which can be used for limited time-span escrow (we refer the reader to [26], [28], [29], [30]). Here we present a k-out-of-l $(k < l)$ scheme which is based on perfect hash families.

To illustrate our technique we first describe a simple 2-out-of-l secret sharing scheme [26]. Then we show how to get a k-out-of-l scheme with $k > 2$, using the Kurosawa-Stinson interpretation [27], [30], of the scheme in [29].

Let $K(\cdot)$ be an Abelian group and $a \in K$ the secret. We number the participants i from 0 to $l-1$ and represent i in binary, *i.e.*, i corresponds to the bits $(i_1, \ldots, i_{\lceil \log_2(l) \rceil})$. For the shares, we choose $\lceil \log_2(l) \rceil$ independent uniformly random elements $r_d \in K$, $d = 1, 2, \ldots, \lceil \log_2(l) \rceil$. Shareholder i receives as sub-share $s_{i,d} = r_d$ if $i_d = 0$, and $s_{i,d} = a \cdot r_d^{-1}$ otherwise. So shareholder i receives $\lceil \log_2(l) \rceil$ sub-shares in total, and the therefore the length of his/her share is $\lceil \log_2(l) \rceil \cdot \mathrm{length}(\mathrm{key})$.

Suppose that shareholders i and j want to reconstruct the key $a \in K$. If $i \neq j$, then the binary representations of i, j will differ in at least one column d, *i.e.*, $i_d \neq j_d$. For this column, $a = s_{i,d} \cdot s_{j,d}$, so the decrypting factor for the limited time-span escrow system ${y_t}^r = g^{r \cdot s_{i,d} \cdot s_{j,d}}$ can be computed by i and j sequentially. This scheme may be regarded as an extension of a 2-out-of-2 scheme whose shares consist of subshares obtained by using the 2-out-of-2 scheme, determined by the binary representation of the index i, $i = 0, 1, \ldots, l-1$, of the users.

We now show how to get k-out-of-l schemes. Let A and A' be finite sets of cardinality l and l' respectively with $l' < l$, B a subset of A and F a set of functions from A to A'. A function $f \in F$ is called a *perfect hash for* B if its restriction to B is one-to-one. F is a *perfect hash family* (l, l', k) if for all subsets $B \subset A$ of order k there is at least one function $f \in F$ which is a perfect hash for B. Note that the binary representation which we considered in the basic 2-out-of-l scheme defines such a perfect hash family from the set A to $A' = \{0, 1\}$.

Suppose that F is a perfect hash family (l, l', k). Then given a k-out-of-l' secret sharing scheme over K we construct new k-out-of-l secret sharing scheme over K as follows. To distribute shares of a key $a \in K$ we use the share generation algorithm of the k-out-of-l' sharing scheme to

[1] In this proof, for simplicity, we drop the operator "mod$(p-1)$" (as well as "mod$p$").

get l' shares $s_{a'}$, $a' \in A'$, for each $f \in F$. Shareholder $i \in A$ gets for each $f \in F$, $s_{i,f} = s_{a'}$ as a subshare if $f(i) = a'$.

A set $B = \{i_1, i_2, \ldots, i_r\} \subseteq \{1, 2, \ldots, l\}$ of $r \geq k$ shareholders reconstructs the secret by first finding an $f \in F$ which is a perfect hash for B. Then B uses the reconstruction procedure of the k-out-of-l' scheme with the subshares $s_{i_j,f}$ which correspond to f. It follows that if the original k-out-of-l' sharing scheme can be used for limited time-span escrow then so can the k-out-of-l scheme. In particular we get a limited time-span k-out-of-l escrow scheme by using a k-out-of-k scheme from Section II. To decrypt a message, the escrow agents EA_j, $j \in B$, compute the partial factors

$$g^{r \prod_{j=1}^{d} s_{i_j,f}^{2^t}} := (g^{r \prod_{j=1}^{d-1} s_{i_j,f}^{2^t}})^{s_{i_d,f}^{2^t}}, \quad d = 1, 2, \ldots, k, \qquad (2)$$

sequentially, where $f \in F$ is a perfect hash for B. The rest is as in Section II-A.

B. Limited time-span escrow based on the extended Shamir scheme

We briefly sketch a k-out-of-l limited time-span escrow scheme which is based on the extended Shamir secret sharing scheme (for a detailed description of this scheme the reader is referred to [24]).

The set of secrets of the extended Shamir scheme is an Abelian group $K(+)$. Let q be a prime, $q \geq l + 1$, and $Z[u] = Z[x]/((x^q - 1)/(x - 1))$ the ring of all polynomials in u with integer coefficients taken modulo the cyclotomic polynomial $(u^q - 1)/(u - 1)$. The set of shares S_i, $i = 1, 2, \ldots, l$, is the $Z[u]$-module $M = Z[u] \otimes_Z K$, which is the tensor product [25] of $Z[u](+)$ and $K(+)$. M can be represented by the direct product $K^{q-1} = K \times K \times \cdots \times K$, with the internal module operation corresponding to vector addition. For the shares of $a \in K$ we choose a random polynomial $f(x)$ over M of degree $(k - 1)$ with $f[0, 0, \ldots, 0] = [a, 0, \ldots, 0]$. The key $a \in K$ may be viewed as an element of K^{q-1}. Then shareholder j receives $s_j = f(x_j)$, where $x_j = \sum_{i=0}^{j-1} u^i$. A set of $r \geq k$ shareholders $B = \{i_1, i_2, \ldots, i_r\} \subseteq \{1, 2, \ldots, l\}$ reconstructs the key by computing:

$$a = \sum_{j \in B} \mathrm{proj}_1 (y_{j,B} \cdot s_j),$$

where proj_1 is the projection of the elements of K^{q-1} onto their first coordinate, the sum is the internal module operation, the product is the external module operation, and the $y_{j,B}$ are the Lagrange interpolation coefficients, $y_{j,B} = \prod_{i \in B, i \neq j} (0 - x^i)(x^j - x^i)^{-1}$ [24].

For our application $K(+)$ is the multiplicative group $Z_{p-1}^*(\cdot)$, so addition corresponds to multiplication and $a = \prod_{j \in B} \mathrm{proj}_1 (y_{j,B} * s_j)$, with "$*$" corresponding to some form of exponentiation. The escrow agents EA_j, $j \in B = \{i_1, i_2, \ldots, i_r\}$ with $r \geq k$, can compute the decrypting factor ${y_t}^r$ as in the case of the l-out-of-l scheme, by first computing $y_{j,B} * s_j^{2^t}$, then taking $s_{j,B} = \mathrm{proj}_1 \left(y_{j,B} * s_j^{2^t}\right)$, and finally computing sequentially

$$g^{r \cdot \prod_{j=1}^{d} s_{i_j,B}} := (g^{r \cdot \prod_{j=1}^{d-1} s_{i_j,B}})^{s_{i_d,B}}, \qquad (3)$$

$d = 1, 2, \ldots, r$. This computation requires that the set of agents EA_j, $j \in B$, is known in advance. The rest of the protocol is as in Section II-A.

The length of the share of the extended k-out-of-l Shamir scheme is in general at least as large as $l \cdot \mathrm{length(key)}$ (as opposed to $\lceil \log_2(l) \rceil \cdot \mathrm{length(key)}$ for the scheme based on perfect hash families). For large l this scheme can be quite inefficient.

C. Security

Remark 1 and both Theorem 1 and Theorem 2 easily extend to the k-out-of-l escrow scheme based on perfect hash families or the k-out-of-l extended Shamir escrow scheme. Furthermore, these results extend to other k-out-of-l escrow schemes such as the one based on the sharing scheme in [26]. The proofs are essentially the same and are omitted.

D. Robustness

Robustness deals with the case when some of the escrow agents are faulty (possibly malicious, *i.e.,* Byzantine). Then the decrypting factor ${y_t}^r$ may be incorrect and the LEA will not be able to decrypt the messages of Bob. There are several ways to achieve robustness. For example we can use the approach in [31], [32]. Here we show how this can be done with the schemes in Section III when there is an upper bound on the number of faulty agents.

Suppose that the number of faulty agents is bounded by $k' < l$, and let $k = l - k'$. We use a k-out-of-l scheme, but require additionally that Bob proves that he has given the correct shares to the escrow agents, and that the escrow agents prove that all their computations during the execution of the escrow protocol are correct. Agents whose proof fails are deemed to be faulty and removed from the reconstruction list. Because of our upper bound on the number of faulty agents, the number of remaining non-faulty agents is still sufficiently large to enable decryption.

Let us explain in more detail how this is done for a k-out-of-l scheme which is based on perfect hash families. For robustness, we require that each escrow agent EA_i, $i = 1, 2, \ldots, l$ has an appropriate encryption key. In the the set-up phase (Section II-A.2), Bob publishes the encryption $c_{i,f} = \mathrm{enc}_{\mathrm{EA}_i}(s_{i,f})$, $z_{i,f} = g^{s_{i,f}}$ and $\mathrm{sign}_{\mathrm{Bob}}(c_{i,f}, z_{i,f})$, for each $f \in F$ and $i = 1, 2, \ldots, l$. Each agent EA_i decrypts the ciphertexts $c_{i,f}$, to get the shares $s_{i,f}$ and then checks that these are correct (that is, that $z_{i,f} = g^{s_{i,f}}$). If any check fails, the agent sends $s_{i,f}, c_{i,f}, z_{i,f}$ and Bob's signature the LEA which checks who cheated. The rest of the set-up is (essentially) as in Section II-A.2. That is, Bob publishes $z_{1,2;f} := g^{s_{1,f} s_{2,f}}$, $z_{1,2,3;f} := g^{s_{1,f} s_{2,f} s_{3,f}}, \ldots,$ $z_{1,2,\cdots,l;f} := g^{s_{1,f} s_{2,f} \cdots s_{l,f}}$ and proves that these are correctly constructed using a zero-knowledge proof of the Diffie-Hellman problem (Appendix IV-B). If any of these proofs fails then

Bob is dishonest and reported to the LEA. The rest is as in Section II-A.3.

In the decrypting phase (Section II-A.4) the remaining agents, say $B' = \{i_1, i_2, \ldots, i_r\}$, compute their partial factors sequentially as in (2), but also prove that these are correct, by using a zero-knowledge proof for the Diffie-Hellman problem. If any proof fails, the faulty agent is removed from the reconstruction set and the computation restarted without this agent.

This means that up to k' computations (the upper bound on the number of faulty processes) may have to be aborted. However, since the number of remaining agents is at least as large as the threshold, the correct decrypting factor will be computed.

A similar approach may be used for the k-out-of-l extended Shamir scheme (Section III-B). In this case the agents send $z_i^t := g^{s_i^{2^t}}$ to the LEA and prove that is correct (this involves $(q-1)$ proofs because z_i^t is a $(q-1)$-tuple in $(Z_{p-1}^*)^{q-1}$). The non-faulty agents must prove that their computations in (3) are correct.

E. Proactive limited time-span escrow

By using the approach in [34], [35], [36], [37] for proactive secret sharing we can get proactive limited time-span escrow.

IV. Limited time-span auctions with closed bids: the basic scheme

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

Our scheme 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, $i = 1, 2, \ldots, l$. 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.[2]

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 aspects of this protocol will be discussed in more detail later. We first describe the protocol more formally. For this purpose we use the additive threshold scheme of Boyd and Frankel [17], [38], and/or the ElGamal threshold scheme of Desmedt–Frankel [19] and use the concept of weak encryption (see *e.g.* [10]).

[2] Since an exhaustive search is parallelizable, some kind of inherently sequential scheme may be used, such as the *time-lock puzzles* proposed in [10]. Our protocol can easily be adapted to allow for such schemes.

A. The protocol

A.1 Setting

The parties involved: the bidder Alice, the receiving officer Bob, a Court, the Law Enforcement Agency LEA, and the Escrow Agents EA_i, $i = 1, 2, \ldots, l$.

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 of equal length, 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.

A.2 Set-up

Alice chooses $a_1 \in_R Z_{\phi(n)}$, $a_2 \in_R Z_q$. 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 $i = 1, 2, \ldots, l-1$, and taking $s_l = a_1 - (s_1 + s_2 + \cdots + s_{l-1}) \bmod \phi(n)$.

1. Alice gives privately to each agent EA_i, $i = 1, 2, \ldots, l$, the share s_i.
2. Alice publishes $z_1 := g_1^{s_1}, z_2 := g_1^{s_2}, \ldots, z_l := g_1^{s_l}$. Each agent EA_i checks that $z_i = g_1^{s_i}$ and reports failure to the LEA. The LEA checks that $y_1 = z_1 \cdot z_2 \cdots z_l$. 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.

A.3 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 and y_2 are both q, and if Alice's public keys are authenticated properly, then Bob accepts the tender and sends Alice a receipt $\mathrm{sign}_{\mathrm{Bob}}(\mathrm{Alice}, \mathrm{ElG}^2(m))$.

A.4 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, $\mathrm{ElG}^2(m)$ 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.

A.5 The Court recovers the bid

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

A.6 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 $g_2 \in Z_n$ is given, with a rather small prime order q. It is important that both $g_2 \bmod p_1 \neq 1$ and $g_2 \bmod p_2 \neq 1$. Otherwise, if say $g_2 \bmod p_1 = 1$, then p_1 is a factor of $g_2 - 1$ and it becomes easy to factor n by taking the $gcd(n, g_2 - 1)$. Observe that for $g_2 = n - 1$ we have $q = 2$, but this trivial case is too small to be of any use for us.

A.7 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 $\mathrm{ElG}(m) = ({g_2}^{r_2}, m{y_2}^{r_2})$. The key for this ciphertext is weak, so the LEA can initiate a procedure to recover m by brute force. (Note that q has to be sufficiently large, as we shall see in the following section.)

A.8 Conspiracy

The agents may be corrupted by the bidding officer Bob. They will recover $\mathrm{ElG}(m) = ({g_2}^{r_2}, m{y_2}^{r_2})$, 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 wiretapper collaborating with $(l-1)$ escrow agents may try to decipher the bid m. This is as hard as breaking the Diffie-Hellman problem.

Proof: For simplicity assume that the dishonest escrow agents are numbered from 1 to $l-1$. Suppose that **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, $z_1, z_2, \ldots, z_l$, 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 (we use the approach in [19], [23]). Take $\bar{s}_1, \bar{s}_2, \ldots, \bar{s}_{l-1} \in_R Z_n$, and $\bar{s} = \bar{s}_1 + \ldots + \bar{s}_{l-1}$. Then let $\bar{z}_1 = g_1^{\bar{s}_1}$, $\ldots$, $\bar{z}_{l-1} = g_1^{\bar{s}_{l-1}}$ and $\bar{z}_l = \bar{y}_1 g_1^{-\bar{s}}$. 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, $\bar{z}_1, \bar{z}_2, \ldots, \bar{z}_l$, 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 $\mathrm{DH}(\bar{y}_1, {g_1}^{\bar{r}_1}) = \bar{y}_1^{\bar{r}_1}$. ■

B. Generalizations and Robustness

The basic auction scheme can easily be generalized to a k-out-of-l limited time-span scheme by using the approach in Section III. There are two issues which need further clarification. The first one concerns the modulus $\phi(n)$ which is not public. The problem is that the shares that Alice gives privately to the escrow agent which are in $Z_{\phi(n)}$ should not reveal any knowledge other than that which follows from the secret key. It is shown in [39], [23] that this is indeed the case for the scheme in Section III-B (this follows from the fact that the extended Shamir scheme is simulatable), and hence for the scheme in Section III-A.

The second issue concerns the decrypting factor $y_1^{r_1}$. For auction schemes the group of exponents $Z_{\phi(n)}(+)$ is additive (and not multiplicative as in the case of escrow mechanisms in Sections III-B, III-A). It follows that the computation in (3) need not be sequential, but can be done in parallel.

For robustness Alice must prove that she has given the correct shares to the escrow agents, and the escrow agents EA_i must prove that they have computed their components $g^{r_1 \cdot s_i}$ of the decrypting factor $y_1^{r_1}$ correctly (that is that, $g^{r_1 \cdot s_i} = \mathrm{DH}(z_i, y_1)$). The approach in Section III-D for escrow schemes can easily be adapted for this purpose.

Acknowledgment

The authors would like to thank Moti Yung for helpful discussions.

References

[1] T. Beth, "Zur Sicherheit der Informationstechnik," *Informatik-Spektrum*, vol. 13, pp. 204–215, 1990.
[2] S. Micali, "Fair public-key cryptosystems," in *Advances in Cryptology – Crypto '92, Lecture Notes in Computer Science #740*, E.F. Brickell, Ed., 1993, pp. 113–138, Springer-Verlag.
[3] "A proposed federal information processing standard for an escrowed encryption standard (EES)," Federal Register, July 30, 1993.
[4] D.E. Denning and D.K. Branstad, "A taxonomy of key escrow encryption systems," *Commun. ACM*, vol. 39 (3) pp. 34–40, 1996.
[5] A.K. Lenstra and P. Winkler and Y. Yacobi, "A key escrow system with warrant bounds," in *Advances in Cryptology – Crypto '95, Lecture Notes in Computer Science #963*, D. Coppersmith, Ed., 1995, pp. 197–207, Springer-Verlag.
[6] J. He and E. Dawson, "A new key escrow cryptosystem," In: Dawson, E., J. Golic, J. (eds), in *Cryptography Policy and Algorithms, Lecture Notes in Computer Science #1029*, E. Dawson, J. Golic, Eds., 1996, pp. 105-114, Springer-Verlag.
[7] D. Chaum, "Security without identification: transaction systems to make Big Brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, 1985.
[8] J. Kilian and T. Leighton, "Failsafe key escrow, revisited," in *Advances in Cryptology– Crypto '95, Lecture Notes in Computer Science #963*, D. Coppersmith, Ed., 1995, pp. 208–221, Springer-Verlag.
[9] M. Bellare and S. Goldwasser, "Verifiable partial key escrow," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 1997.
[10] R.L. Rivest, A. Shamir, and D.A. Wagner, "Time-lock puzzles and time-release crypto," Tech. Rep.,

http://theory.lcs.mit.edu/ rivest/publications.html (to appear), 1998.

[11] W. Diffie and M.E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT–22, pp. 644–654, 1976.

[12] U.M. Maurer and Y. Wolf, "Diffie-Hellman oracles," in *Advances in Cryptology – Crypto '96, Lecture Notes in Computer Science #1109*, N. Koblitz, Ed., 1996, pp. 268–282, Springer-Verlag.

[13] L.M. Adleman and K.S. McCurley, "Open problems in number theoretic complexity," in *Discrete Algorithms and Complexity, Proceedings of the Japan-US Joint Seminar (Perspective in Computing series, Vol. 15)*, D. Johnson, T. Nishizeki, A. Nozaki, and H. Wilf, Eds., Kyoto, Japan 1986, pp. 263–286.

[14] Y. Frankel and M. Yung, "Escrowed encryption systems visited: Threats, attacks, analysis and designs," in *Advances in Cryptology – Crypto '95, Lecture Notes in Computer Science #963*, D. Coppersmith, Ed. 1995, pp. 135–155, Springer-Verlag.

[15] G.J. Simmons, Observation made at the Workshop on Key Escrow, June 22–24, 1994.

[16] M. Yung, Personal communication, October 1998.

[17] C. Boyd, "Digital multisignatures," in *Cryptography and coding*, H. Beker and F. Piper, Eds., 1989, pp. 241–246, Clarendon Press, 1986.

[18] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inform. Theory*, vol. IT–31, pp. 469–472, 1985.

[19] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Advances in Cryptology – Crypto '89, Lecture Notes in Computer Science 435*, G. Brassard, Ed., 1990, pp. 307–315, Springer-Verlag.

[20] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *Siam J. Comput.*, vol. 15, no. 2, pp. 364–383, 1986.

[21] T.P. Pedersen, "A threshold cryptosystem without a trusted party," in *Advances in Cryptology – Eurocrypt '91, Lecture Notes in Computer Science 547*, D.W. Davies, Ed., 1991, pp. 522–526, Springer-Verlag.

[22] Mike Burmester, "Homomorphisms of secret sharing schemes," in *Advances in Cryptology – Eurocrypt '96, Lecture Notes in Computer Science #1070*, U. Maurer, Ed., 1996, pp. 96–106, Springer-Verlag.

[23] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, "How to share a function securely," in *Proceedings of the twenty-sixth annual ACM Symp. Theory of Computing (STOC)*, May 23–25, 1994, pp. 522–533, Montréal, Québec, Canada. Full paper in preparation (available from authors when completed).

[24] Y.G. Desmedt and Y. Frankel, "Homomorphic zero-knowledge threshold schemes over any finite abelian group," *SIAM Journal on Discrete Mathematics*, vol. 7, no. 4, pp. 667–679, 1994.

[25] N. Jacobson, *Basic Algebra II*, W. H. Freeman and Company, New York 1989.

[26] Y. Desmedt, G. Di Crescenzo, and M. Burmester, "Multiplicative non-abelian sharing schemes and their application to threshold cryptography," in *Advances in Cryptology — ASIACRYPT '94, Lecture Notes in Computer Science 917*, J. Pieprzyk and R. Safavi-Naini, Eds., 1995, pp. 21–32, Springer-Verlag.

[27] K. Kurosawa, D. Stinson, Personal communication, June 1996.

[28] N. Alon, Z. Galil, and M. Yung, "Efficient dynamic re-sharing "verifiable secret sharing against mobile adversary," *Algorithms – ESA '95, Third Annual European Symposium, Lecture Notes in Computer Science #979*, pp. 523–537, 1990.

[29] S.R. Blackburn, M. Burmester, and Y. Desmedt, "Efficient multiplicative sharing schemes," in *Advances in Cryptology – Eurocrypt '96, Lecture Notes in Computer Science #1070*, U. Maurer, Ed., 1996, pp. 107–118, Springer-Verlag.

[30] Y. Desmedt, "Some recent research aspects of threshold cryptography," in *Information Security, First International Workshop ISW '97, Lecture Notes in Computer Science #1196*, M. Mambo E. Okamoto, E. Davida, Ed., 1997, pp. 158–173, Springer-Verlag.

[31] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust and efficient sharing of rsa functions," in *Advances in Cryptology – Crypto '96, Lecture Notes in Computer Science #1109*, N. Koblitz, Ed., 1996, pp. 157–172, Springer-Verlag.

[32] Y. Frankel, P. Gemmel, and M. Yung, "Witness based cryptographic program checking and robust function sharing," in *Proceedings of the twenty eighth annual ACM Symp. Theory of Computing, STOC*, 1996, pp. 499–508.

[33] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology – Crypto '86, Lecture Notes in Computer Science 263*, A. Odlyzko, Ed., 1987, pp. 186–194, Springer-Verlag.

[34] Y. Desmedt, S. Jajodia, "Redistributing secret shares to new access structures and its applications," Tech. Rep., George Mason University, July 1997, ftp://isse.gmu.edu/pub/techrep/97_01_jajodia.ps.gz

[35] A. Herzberg, S. Jarecki, S. Krawczyk, and M. Yung, "Proactive secret sharing," in *Advances in Cryptology – Crypto '95, Lecture Notes in Computer Science #963*, D. Coppersmith, Ed., 1995, pp. 339–352, Springer-Verlag.

[36] Y. Frankel, P. Gemmel, P.D. MacKenzie, and M. Yung, "Proactive RSA," in *Advances in Cryptology – Crypto '97, Lecture Notes in Computer Science #1109*, B.S. Kaliski, Ed., 1997, pp. 440–454, Springer-Verlag.

[37] T. Rabin, "A simplified approach to threshold and proactive rsa," in *Advances in Cryptology – Crypto '97, Lecture Notes in Computer Science #1462*, H. Krawczyk, Ed., 1998, pp. 89–104, Springer-Verlag.

[38] Y. Frankel, "A practical protocol for large group oriented networks," in *Advances in Cryptology – Eurocrypt '89, Lecture Notes in Computer Science #434*, J.-J. Quisquater and J. Vandewalle, Eds., 1990, pp. 56–61, Springer-Verlag.

[39] Y. Frankel and Y. Desmedt, "Parallel reliable threshold multisignature," Tech. Report TR–92–04–02, Dept. of EE & CS, Univ. of Wisconsin–Milwaukee, April 1992.

[40] D. Chaum, "Zero-knowledge undeniable signatures," in *Advances in Cryptology – Eurocrypt '90, Lecture Notes in Computer Science 473*, I. Damgård, Ed., 1991, pp. 458–464, Springer-Verlag.

[41] O. Goldreich, S. Micali, and C. Rackoff, "Knowledge Complexity of Interactive Proofs," *Siam J. Comput.*, vol. 18 (1), pp. 186–208, 1989.

Appendix

The zero-knowledge proof for the Diffie-Hellman decision problem in [40] is not adequate for our purpose, since it is 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.

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

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 $a' = x$, $b' = y$ to the Verifier, and the Verifier checks that: $g^x \bmod p = \kappa_{a'}$, $g^y \bmod p = \kappa_{b'}$, $g^{xy} \bmod p = \kappa_{a'b'}$, $\kappa_{ab'} \bmod p = \alpha^y$, and $\kappa_{ba'} \bmod p = \beta^x$.
   If $e = 1$ the Prover sends $a' = a + x \bmod (p-1)$, $b' = b + y \bmod (p-1)$ to the Verifier who checks that: $g^{a'} \equiv \alpha \cdot \kappa_x \pmod{p}$, $g^{b'} \equiv \beta \cdot \kappa_y \pmod{p}$, and $g^{a'b'} \equiv \gamma \cdot \kappa_{ay} \cdot \kappa_{bx} \cdot \kappa_{xy} \pmod{p}$.
   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 are completed successfully.

Let $L = \{(p, , g, \alpha, \beta, \gamma) \mid p \text{ prime}, \alpha = g^a \bmod p, \beta = g^b \bmod p, \gamma = g^{ab} \bmod p\}$. Then, *Theorem 4:* The protocol above is a perfect zero-knowledge proof of membership in L.

Proof: We use the model for proofs of membership in [41].

Completeness: Suppose that the Prover and Verifier follow the protocol. Then when $e = 0$: $g^{a'}, g^{b'}, g^{a'b'}, g^{ab'}$ and $g^{ba'}$ are of the correct form and the Verifier will accept. When $e = 1$: $g^{a'} = g^{a+x} = \alpha \cdot \kappa_x \,(\mathrm{mod} p)$ is of the correct form, and similarly $g^{b'}, g^{a'b'}$, and $g^{a'b'}$. So the Verifier will accept.

Soundness: Suppose that a Verifier V who follows the protocol accepts the proof of the prover with non-negligible probability. We must show that $(p, g, \alpha, \beta, \gamma) \in L$. First observe that if V accepts the proof then for some $(\kappa_x, \kappa_y, \kappa_{xy}, \kappa_{ay}\kappa_{bx})$ sent in Step 1, the prover must be able to answer both the queries $e = 0, 1$. For these, 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 $\gamma \equiv g^{a'b'} \cdot \kappa_{ay}^{-1} \cdot \kappa_{bx}^{-1} \cdot \kappa_{xy}^{-1} \equiv g^{ab} \,(\mathrm{mod} p)$. It follows that $(p, \alpha, \beta, \gamma) \in L$.

Zero-knowledge: We must show that the *view* $(\kappa_x, \kappa_y, \kappa_{xy}, \kappa_{ay}, \kappa_{bx}; e\,; a', b')$ of a possibly dishonest Verifier can be simulated in (expected) polynomial time. Let α, β, γ be given. Pick $a', b' \in_R Z_{p-1}$, and let $\bar{\kappa}_x = g^{a'}/\alpha \,(\mathrm{mod} p)$, $\bar{\kappa}_y = g^{b'}/\beta \,(\mathrm{mod} p)$. Then solve $\alpha^{b'} \equiv \gamma \cdot \bar{\kappa}_{ay} \,(\mathrm{mod} p)$, $\beta^{a'} \equiv \gamma \cdot \bar{\kappa}_{bx} \,(\mathrm{mod}\ p)$, $g^{a'b'} \equiv \gamma \cdot \bar{\kappa}_{ay} \cdot \bar{\kappa}_{bx} \cdot \bar{\kappa}_{xy} \,(\mathrm{mod} p)$, for the unknowns $\bar{\kappa}_{ay}, \bar{\kappa}_{bx}$, $\bar{\kappa}_{xy}$, respectively, to get the simulation $(\bar{\kappa}_x, \bar{\kappa}_y, \bar{\kappa}_{xy}, \bar{\kappa}_{ay}, \bar{\kappa}_{bx}; e\,; a', b')$. ∎

Remark 3: One may wish to use the non-interactive Fiat-Shamir signature [33] of the Diffie-Hellman proof (Appendix IV-B). This will simplify the protocol. However it will affect the proofs of both Theorem 1 and Theorem 2 in the extended case, since these require a simulation of the Diffie-Hellman proofs. For example, it will not be possible to simulate the Fiat-Shamir signature of the proof for $\bar{y}_{t-u} = \mathrm{DH}(\bar{y}_{t-u-1}, \bar{y}_{t-u-1})$ in Theorem 1, which is highly unlikely to be true. Similarly for the signature of the proof of $\bar{y}_t = \mathrm{DH}(\bar{y}_{t-1}, \bar{y}_{t-1})$ in Theorem 2.