RSA-Based Auto-recoverable Cryptosystems

Adam Young* and Moti Yung**

Abstract. The deployment of a "public-key infrastructure" (PKI) has recently started. Another recent concern in business and on the national level is the issue of escrowed encryption, key recovery, and emergency access to information (e.g., in the medical record area). Independent development of a PKI and an escrowed PKI (whenever required or desired) will pose a lot of constraints, duplication efforts and increased costs of the deployment. It will introduce inter-operability issues which will be hard to overcome. Thus, what we advocate here is a joint design of an escrowed PKI and a regular PKI.

In this work we develop an approach to such an integrated design. We give the first auto-recoverable systems based on RSA (or factoring), whereas the original auto-recoverable auto-certifiable schemes were based on Discrete Logarithm based keys. The security proof of our system assumes only that RSA is hard, while the original schemes required new specific discrete log based assumptions. We also put forth the notion of "generic" auto-recoverable systems where one can start with an unescrowed user key and then by simply doing "re-registration", change the key into an escrowed one. In contrast, in the original systems the user keys were tightly connected with the escrow authorities' key. Besides this novel (re)-registration procedure there are no changes or differences for users between a PKI and a generic auto-recoverable PKI.

1 Introduction

In this paper we develop an escrowed public key infrastructure that is as close as possible to infrastructures without escrow, and that can be used to escrow a subset of users. This is useful for numerous reasons. First, showing that an auto-recoverable escrowed PKI (i.e., one that is compatible with a regular PKI) can be based solely on the same functions that are used to implement a PKI (e.g., the RSA function [RSA78]), while at the same time not introducing new mathematical assumptions, makes the system more trustworthy. Second, due to the similarity of the escrowed system with unescrowed ones, much of the same software and practices for the former can simultaneously be applied to the later, making the composed solution more economical. Lastly, the coexistence of escrowed keys with unescrowed ones within the same infrastructure will minimize the use of escrow to that of the "required keys", and will enable operators of the PKI to comply with regulations and needs while allowing as much privacy as possible.

Enabling after the fact escrow where an unescrowed key can be used online and then can later be transitioned into "escrowed mode" is a facility which

* Currently: Computer Science, Columbia University. ayoung@cs.columbia.edu.
** Currently: CertCo NY, USA. moti@certco.com, moti@cs.columbia.edu

H. Imai, Y. Zheng (Eds.): PKC 2000, LNCS 1751, pp. 326–341, 2000.

seems useful. It combines on-line privacy with the ability to archive data for "future historical purposes" with controlled access. In this context information and keys can be securely deposited into a national archive. It gives maximal on-line privacy which is a must (for secrecy of operations on the national level). Yet, it should not prevent future access when the accessibility of the data becomes more important than outdated information secrecy. This separability of keys and escrow authorities also implies that the same key can be deposited with different escrow authorities.

The development of a system under the constraint of being "auto-recoverable" [YY98] enables the system to be software-based (i.e., user's programs can be distributed and operated in software without the need for tamper-proof hardware). Another inherited property due to these constraints is having a key escrow system with minimal overhead (i.e., in the PKI context, making it as close as possible to a regular PKI) while retaining flexibility (e.g., separating users from the inner workings of and access to, escrow agents). This enables the escrowed PKI to be used with the same ease as a typical unescrowed PKI. A list of privacy, abuse-freeness (no leakage), flexibility, and operational properties desired was given in [YY98].

The main challenge in implementing generic systems with a "drop-in replacement" property is basing it on general public-key (trapdoor) functions in a way which achieves certain verification, compliance, and security requirements, and having it such that the security totally relies on the hardness assumptions of the basic cryptographic functions. Along the way we also define the notion of a "zero-knowledge proof of knowledge in the random oracle mode" (we need it for our proofs and we did not find it in the literature). Additional properties we achieve for the system constructed here are:

1. **Employing RSA Keys:** The systems can be based on the most widely deployed system, i.e., RSA/factoring-based systems.
2. **Minimal User Overhead:** The only change for the user is additional information sent during key registration (or re-registration). Users need not re-key to escrow existing public keys.
3. **No Cascading Changes:** The users do not change their applications and systems software which employ cryptography (besides the registration procedure). Not even the cryptographic engine (e.g., an RSA based engine) need be changed.
4. **Separation of Users and Escrow Agents:** The escrow authorities are managed and constructed independently of the users (only their public key(s) need be known for (re)-registration, but not for the user's key generation).
5. **Independent User Keys:** The user's key is independent of any other key and is produced in much the same way as in an unescrowed PKI. The users employ the same basic crypto algorithms (key generation, encryption, etc.).
6. **Multiple Escrow Authorities** Users can register for escrow with multiple escrow authorities or with one of their own choosing, among other options.
7. **Granularity of Escrow:** When the escrow agents are activated, they may either open keys or open information encrypted under the keys. This enables

escrow within a specific context while keeping users' privacy otherwise. For national law enforcement applications, this feature is very important to allow minimization of escrow by avoiding revealing the private keys of receivers when senders are the ones under suspicion.

8. **Escrow/non-Escrow coexistence:** The same PKI can handle escrowed and unesrowed keys (which minimizes escrow to the keys needed to undergo the escrow process).
9. **Escrow Hierarchy:** A multi-level security system can be implemented where the escrow authorities at each level can access all of the information below in the hierarchy, and none of the rest of the information.

2 Background and Related Work

The definitions of the notions we employ are given in appendix A. In the appendix we also present the new definition of a zero-knowledge proof of knowledge in the random oracle model (which we need in our proofs and which was not available in the literature).

Micali used VSS to design public key cryptosystems with escrowed capabilities when he proposed the notion of Fair Public Key Cryptosystems [Mi92]. His system employs a "verifiable secret sharing protocol" to distribute users' keys to the escrow agents [CGMA,Fe85]. The problem with Fair PKC's is that first, every user must split his private key and interact with the escrow agents who then need to interact among themselves to approve a key and be convinced that it is escrowed properly. This is *not* a "minimal change" to a PKI (the entire initialization of the system is changed). Second and even more importantly, there are information leakage attacks on such systems. In fact, if used where each user's public key is publicly verifiable, the system requires the publication of an excessive amount of information; this can be abused to permit 'shadow public keys' to be published (which are unescrowed), a notion due to Kilian and Leighton [KL95]. The notion of having an escrow system be 'shadow public key resistant' is a very recent and important new problem to consider (naive attempts to associate auto-escrow with merely publicly verifiable encryptions or shared encryptions may result in similar information leakage flaws). It is hard to prove shadow public key freeness, but we can require that the public information in an escrowed system be the same as the public information in a regular PKI. Kilian and Leighton [KL95] suggested a correction to Fair PKC's which they call Fail-Safe Key Escrow to avoid shadow public key abuse, but their impractical solution requires even more protocol interaction than Micali's solution.

A more recent solution which attempts to minimize the impact on users when adding escrow and which limits information leakage is the notion of "auto-recoverable auto-certifiable" systems of [YY98], where users interact with CA's only to set up their keys and thereafter use the escrowed PKI as if it were a regular PKI. The system employed double decker exponentiation ([St96] and also [CS97,YY98,YY99]). The starting point model of the solution we present is also that of an auto-recoverable system. We present extended advantages of the notion.

Publicly Verifiable Secret Sharing (PVSS) is related (but not sufficient) for the above notion. A PVSS for sharing a composite of two primes was given in [FO98]. The solution encrypts the user's shares using RSA, and are thus not semantically secure encryptions. The scheme relies on a new modified RSA assumption. Also, when the shares are recovered, a factor of n is obtained, thus no mechanism for decrypting individual messages is given.

A number of recent suggestions have appeared after the initiation of this area in [YY98] and after the initial announcements of our results (e.g. [Man97]). Naturally, we welcome other works in this area. One work is another PVSS for composites which was given in [BT99]. This solution utilizes numerous SZK proofs and requires the use of a composite whose factorization is unknown to all of the parties involved. The solution has the property that it requires $O(2^{21})$ modular exponentiations to recover the shared secret (a factor) of the user, and requires $O(2^{80})$ tries to forge the PVSS proof. Shanks algorithm must be used to recover the shared secret. Like [FO98], the recovery algorithm reveals one of the factors of n, and no algorithm for decrypting individual messages is given; in addition, the encryption is not semantically secure. In [FPS99] an auto-recoverable solution for composites is promised. A cryptographic primitive called Diophantine Commitment (related to bounded range SZK proofs, see [CFT98]) is discussed. The solution is also based on [PS00]. From what is currently available [PS-L], we believe (but may be wrong) that the scheme proposed may not have all the properties we discuss herein, yet it provides very short certificates of recoverability which is a very elegant and useful property. Other schemes related to the original work [YY98] have appeared in [YY99,Ve00] where a scalable security for the recovery agent was provided, and in [Sch99] which simplified the original scheme's proofs.

It is worth while noting that numerous independent works have recently recognized the role of an off-line independent third party. The work on group signatures and ID-escrow [CS97,KP98], escrowed e-cash [CMS96,FTY96], registered mail [Mi92], and fair exchange regarding signatures [ASW98,Ch98], all employed mechanisms for verifying and transferring information to an off-line third party.

3 Basics of a Generic Auto-recoverable PKI

3.1 Defining an Auto-recoverable PKI

Assume that we have a public key function. This function has a key generation procedure GEN, which generates a public key K_1 and its associated private key portion K_2. Given the pair (K_1, K_2), when K_1 is registered in the public file, the user can run EMP where they employ the public key for encryption.

Our definition adds components that will be added to a PKI.

Definition 1. *A Generic Auto-Recoverable Cryptosystem contains the following algorithms (CER,VER,REC) such that:*

1. *CER is a publicly known poly-time probabilistic algorithm that takes the output of GEN, which is (K_1, K_2), and the keys of the escrow authority and generates the triple (K_1,K_2,P) which is left on the tape as output. Here K_2 is the randomly generated private key and K_1 is the corresponding public key. P is the transcript of the zero-knowledge proof of knowledge (its non-interactive version may be a poly-sized certificate that proves that K_2 is recoverable by the escrow authorities using P).*
2. *VER is a publicly known poly-time deterministic algorithm that takes (K_1,P) on its input tape and returns a boolean value. With very high probability, VER returns true iff P can be used to recover the private key K_2.*
3. *REC is a private poly-time deterministic algorithm that takes (K_1, P) as input and returns K_2 on its tape as output with overwhelming probability provided VER(K_1,P) is true (REC may be a distributed protocol among distributed entities).*
4. *It is intractable to recover K_2 given K_1 and P without REC.*

We say that CER is a valid certification protocol if:

- the protocol: (CER,VER) is a computational zero-knowledge proof of knowledge of K_2 on input K_1 (of size h) (we will describe the non-interactive random oracle based system, but an interactive procedure is also possible).
- In addition the following property holds:
  Transferability with error k: For any CER' if $p(h)$ is the probability that the CA accepts in VER on input K_1, P from CER', then REC is successful on the same input with probability at least $\max\{p(h) - k(h), 0\}$, for some $k < p$ (we are interested in negligible k).

Let us next explain in what sense the above is generic. In a regular PKI the following is done: (1) the CA publishes its parameters, (2) using GEN a user generates a key pair, accesses the CA, and registers the public key, (3) using EMP the users employ the system to send encrypted messages. In the generic escrowed PKI system, the CA parameters will include the Escrow Authority (EA)'s shared public key. In registration the user will execute GEN and will execute CER to add to the key the transcript P (or by interaction), the CA verifies it using VER. Otherwise, everything else is as in a regular PKI. Users send messages using EMP as before. When keys are taken out of escrow, the CA needs to cooperate and it supplies P to the EA which can then recover the private information using REC.

3.2 The Basic Structure of the System

To implement the system we will employ the following ingredients:

- General semantically-secure public key cryptosystem of the EA's.
- Public commitment schemes which are homomorphic, so that multiplications of commitment values result in a commitment of the sum of the plaintexts.
- The fact that in the number-theoretic public-key systems there exist homomorphic structure which enables homomorphic commitments.

3.3 System Initialization

The following are the cryptographic primitives that are used in our system by the escrow authorities. ENC is a semantically secure probabilistic public key encryption algorithm that takes three arguments, r, s, and E. Here r is a message to be encrypted, s is a randomly chosen string to make the encryption probabilistic, and E is a public key. Thus, $C = ENC(r, s, E)$ is the ciphertext of the message r. Let DEC be the corresponding decryption function. Thus, $r = DEC(C, D)$, where D is the private key corresponding to E. It could be that D is shared distributively, in which case $r = DEC(C, D_1, D_2, ..., D_m)$.

Semantic security is sufficient in our case (and there is no need for chosen ciphertext security), since the decryption will be performed to recover keys whose "ciphertext transcript" includes a proof of knowledge of the key by the encrypting party. Note that P proves message awareness.

The system is independent of the organization of the escrow agents as a distributed entity. In the case of m escrow authorities, they generate the shares $D_1, D_2, ..., D_m$ of their shared private key D, and they collaboratively compute their corresponding public key E. To this end, the notion of function sharing to enable threshold decoding can be employed [DDFY,FGY,GJKR,FGMY].

E is published as the key of the EA's. We may also allow a number of keys to be published where users choose the escrow authority key or keys to work with.

4 Generic Auto-recoverable Systems

4.1 Constructing the Certification and Recovery Mechanisms

We combine the notions of encryption and zero-knowledge proofs and add them on top of the key generation procedure. For efficiency of communication (which assures that the new scheme is embeddable in an old protocol), we employ the methodology of employing random oracles. This methodology was put forth by Bellare and Rogaway [BR94] to model the use of a cryptographic hash function. This notion was used originally by Fiat and Shamir in [FS86] and also Schnorr [Sc91], and was proved rigorously by Pointcheval and Stern [PS96]. The proof in the random oracle model validates the design and does not constitute a rigorous complexity-theoretic proof, however it typically gives efficient solutions. In our case it reduces interaction. Some weaknesses (not in the current use) are known, and validation via a random oracle is not universal, especially if the cryptographic functions themselves rely on the oracle [CGH98] – but here as well as in many other uses, separation exists. In any case, interactive registration variants of our constructions (which use only two full rounds) exist, and the penalty is only extra interaction. Rather than self-challenging by an oracle as in the descriptions below, a traditional challenge-response interaction takes place. Zero-knowledge proofs in the random oracle model were carefully defined in [BR94].

We will employ "split encryption" where a value is put in a number of places. Such a primitive and its demonstrated power have been used and shown

in various contexts: zero-knowledge proofs, e-cash and secret sharings., e.g., in [IY86,KMO89,CFN,FY93,Fe85,BG96], to give just a partial list.

4.2 RSA Based Systems

CER: Key Certification The user generates a product of two primes $n = pq$. Let d be the inverse of e mod $\phi(n)$ (this is necessary, since we need $Z_{\phi(n)}$ to be indistinguishable from Z_n for ZK). Here e and d are the public and private RSA exponents, respectively. We assume that before this protocol is engaged, P proves that n is the product of two primes. A protocol that proves that n is of the form $n = p^r q^s$, where p and q primes (congruent to 3 mod 4 where r and s are both odd) is given in [GHY] (resp. [GP87]). A protocol by Boyar et al. can be used to prove that n is square free [BFL91]. Taken together they assure that n is the product of two primes. There are non-interactive versions of such compliance proofs which we employ. In addition, care must be taken not to allow the exploitation of the "Desmedt subliminal channel" [De88] (as in [YY96]) to leak information. So, the upper half of the bits of n should be generated by a call to a random-oracle hash function and the preimage of this call should be given to the CA for verification.

4.3 Non-interactive Solution

In the protocol that follows, the prover uses an ideal hash function (assumed to be indistinguishable from a random oracle) to generate the values for t'_{i-1}. Thus, the underlying atomic protocol can be viewed as a protocol with a success probability of $\tau(n)/2n$, where $\tau(n)$ denotes the number of elements modulo n with order $\lambda(n)$. Let $k(m) = \omega(log\ m)$ be given. We will increase the size of the transcript by a factor of δ to achieve an error of at most $2^{-k(m)}$. In [LS] it was shown that if n is the product of two primes, then $\tau(n) > n/(3\ lnln\ n)$. Since we insist that P proves to V that n is the product of two primes, we can take $\delta = 12(lnln\ n)ln\ 2$ to insure that d is transfered in CER with overwhelming probability.

CER: Key Certification The following is how the non-interactive proof of knowledge transfer P is constructed:

1. P = (n), $t_0 = H(n)$ (H is a random hash function with range Z_n^*)
2. for $i = 1$ to $\delta k(m)$:
3. $\quad t'_{i-1} = H(t_{i-1})$
4. $\quad t_i = t'^{e}_{i-1}\ mod\ n$
5. for $i = 1$ to $\delta k(m)$ do
6. $\quad a_i \in_R Z_{\phi(n)}$, choose $s_{i,1}, s_{i,2}$ randomly for use in ENC
7. $\quad v_i = {t_i}^{a_i}\ mod\ n$
8. $\quad C_{i,1} = ENC(a_i, s_{i,1}, E)$, $C_{i,2} = ENC(d - a_i\ mod\ \phi(n), s_{i,2}, E)$
9. $\quad$ add $(v_i, C_{i,1}, C_{i,2})$ to the end of P

10. val = $H''(P)$ (H'' is a random oracle hash)
11. set $b_1, b_2, ..., b_{\delta k(m)}$ to be the $\delta k(m)$ least significant bits of val, where $b_i \in \{0,1\}$
12. for $i = 1$ to $\delta k(m)$ do
13. let $a_{i,1} = a_i$ and let $a_{i,2} = d - a_i \ mod \ \phi(n)$
14. add $z_i = (a_{i,j}, s_{i,j})$ where $j = 1 + b_i$ to the end of P

Thus, $P = (n, (v_1, C_{1,1}, C_{1,2}), ..., (v_{\delta k(m)}, C_{\delta k(m),1}, C_{\delta k(m),2}), z_1, ..., z_{\delta k(m)})$. Note that since $t_0 = H(n)$, the prover must commit to the composite before the oracle gives the prover the randomly chosen bases $t_1, t_2, ..., t_{\delta k(m)}$. Thus, a user is able to pick from a polynomial number of $(\delta k(m)+1)$-tuples $(n, t_1, t_2, ..., t_{\delta k(m)})$ with randomly chosen values for t when deciding on the public key to give to the CA. Note that a prover conditioning the transcript for a favorable tuple is the same situation as a prover conditioning a transcript for favorable challenge bits.

VER: Public Escrow Verification The verifier computes $t_1, t_2, ..., t_{\delta k(m)}$ himself based on n, and the verifier computes $b_1, b_2, ..., b_{\delta k(m)}$ in the same way as in the certificate generation process. The verifier checks that all of the values in P lie in the correct sets. For example, the verifier checks that the $t_i \in Z_n^*$, and that $a_{i,1+b_i} < n$ for $1 \leq i \leq \delta k(m)$. If any of these verifications fails, then the verifier concludes that the private key is not properly escrowed. For i ranging from 1 to $\delta k(m)$, the verifier verifies the following things:

1. $C_{i,1+b_i} = ENC(a_{i,1+b_i}, s_{i,1+b_i}, E)$
2. ${t_i}^{a_{i,1+b_i}} = (t'_{i-1}/v_i)^{b_i} {v_i}^{1-b_i} \ mod \ n$

The verifier concludes that the private key is escrowed as long as all the verifications pass and as long as both criterion are satisfied for $1 \leq i \leq \delta k(m)$.

REC: Key Recovery The escrow authorities recover the user's private key as follows. For i ranging from 1 to $\delta k(m)$, the authorities compute d'_i to be the sum of the plaintexts corresponding to $C_{i,1}$ and $C_{i,2}$. Let $K_i = ed'_i - 1$. The escrow authorities then utilize the well known Las Vegas algorithm that factors n given a multiple of $\lambda(n)$. This algorithm is run on K_i for each i.

4.4 Security

To prove the security, we will present the atomic version of the proof in the interactive case. We will then argue its security, and apply the Bellare-Rogaway reduction to prove the security of CER. Let $t' \in_R Z_n^*$ be an element chosen jointly by P and V. P and V compute $t = t'^e \ mod \ n$.

1. P chooses $a \in_R Z_{\phi(n)}$ and s_1,s_2 randomly for use in ENC
2. P computes $v = t^a \ mod \ n$
3. P computes $C_1 = ENC(a, s_1, E)$, $C_2 = ENC(d - a \ mod \ \phi(n), s_2, E)$
4. P sends (v, C_1, C_2) to V

5. V sends $b \in_R \{0,1\}$ to P
6. P sends $z = (w, s_{1+b}) = (a^{1-b}(d-a)^b \ mod \ \phi(n), s_{1+b})$ to V
7. V verifies that $C_{1+b} = ENC(w, s_{1+b}, E)$ and $t^w = (t'/v)^b v^{1-b} \ mod \ n$

Lemma 1. *The atomic 3-round interactive protocol is computational zero-knowledge.*

Proof. To prove that it is ZK, we will give a description of a poly-time simulator S^* that is capable of producing transcripts for the interactive version that are polynomially indistinguishable from a transcript derived from a prover and verifier in an interactive session. Let V^* be any polynomial-time probabilistic algorithm that a (possibly cheating) verifier uses to generate his challenges. S^* is defined as follows:

1. $t = t'^e \ mod \ n$
2. $b' \in_R \{0,1\}$, $w, w' \in_R Z_n$, choose s_1,s_2 randomly for use in ENC
3. $v = t^{w(1-b')}(t'/t^w)^{b'} \ mod \ n$
4. $C_{1+b'} = ENC(w, s_{1+b'}, E)$, $C_{2-b'} = ENC(w', s_{2-b'}, E)$
5. call V^* with input (v, C_1, C_2) obtaining challenge b
6. if $(b' = b)$ then return $(v, C_1, C_2, b, w, s_{1+b})$ else goto 2

Note that $w, w' \in_R Z_n$ in the simulation, since the simulator does not know $\phi(n)$. However $Z_{\phi(n)}$ and Z_n are statistically indistinguishable. Note that indistinguishability won't hold if there exists a line tapper T_N that when given $\{n, v^T, C_j^T, v^F, C_j'^F\}$ (in no particular order) can distinguish C_j^T as containing, for example, the plaintext $(d-w)^T$ from C_j^F containing $(d-w)^T$ with probability $> 1/2 + N^{-c}$. Here a^T indicates a value a that is in a transcript formed between a prover and a verifier, and b^F indicates a value b that is in a transcript forged by the simulator. The ciphertexts correspond to the unopened ciphertexts, and $j \in \{1,2\}$. Recall that semantic security implies that anything that can be learned about the plaintexts from these two ciphertexts can be learned without the ciphertexts, so the line tapper is unable to distinguish based on these values (even if T_N were given both plaintexts). Hence, semantic security is a sufficient condition. QED.

Note that adaptive chosen ciphertext security of ciphertexts is not needed in our system since users never see the decryptions of the values in the certificates (yet their proof achieves the notion via the formalism of a proof of knowledge). The interactive version of the CER protocol is a typical 3-round computational zero-knowledge proof. Completeness and soundness are straightforward. It differs from standard zero-knowledge proofs in that, in addition to a commitment value being sent in the first round (i.e., using v to commit to the base t logarithm of v mod n), the prover sends two semantically secure encryptions (which must be consistent with the commitment in v, and d). The first of these encryptions can be thought of as yet another commitment of the base t logarithm of v. The second encryption is a commitment to the base t logarithm of t'/v. Since these are both

semantically secure encryptions, they give as much information to a poly-time adversary as the adversary can compute himself without the encryptions. They are, in some sense, redundant commitments of v and t'/v (they are useful since they provide third party recoverability). In the second round, the verifier sends a randomly chosen bit to the prover, as in standard zero-knowledge proofs. In the third round, the verifier has to open either a (for v) or $d - a \ mod \ ord(t)$ (for t'/v), as in standard zero-knowledge proofs. The only difference is that exactly one of the two semantically secure encryptions (commitments) is also opened, and verified for consistency (to insure third party recoverability).

In section 5.2 of [BR94], a reduction is given that shows how to transform any three move atomic zero-knowledge proof with error probability 1/2 for $L \in NP$ into a non-interactive ZK proof in the random oracle model. The only significant differences here are that two semantically secure encryptions are sent in each round (this is secure for the same reason as in the DL version) and that the error probability of the non-interactive RSA atomic protocol is not 1/2. The fundamentals of the reduction and the proofs of soundness and zero-knowledge are unchanged. So, we have the following.

Lemma 2. *The non-interactive CER protocol for factoring based keys is sound and zero-knowledge in the random oracle model.*

Completeness of the non-interactive protocol for factoring based keys is straightforward. We will now prove that the non-interactive CER protocol for factoring based keys is a proof of knowledge in random oracle model. This will shed some light on the differences in the proof of soundness as compared to the soundness discussed in section 5.2 of [BR94]. See the appendix for a formalization of what it means for a non-interactive proof to be a proof of knowledge in the random oracle model. Note that this proof assumes that $b_1, b_2, ..., b_{\delta k(m)}$ are included in the transcript. This is a minor modification to the protocol, which clearly can be made since the verifier can verify this information.

Lemma 3. *The non-interactive CER protocol for factoring based keys constitutes a proof of knowledge with knowledge error at most* $2^{-k(m)}$.

Proof. It is easy to see that the non-triviality condition holds. We will now consider the validity condition. The common input is $\alpha = n$. We have that $x_i = (v_i, C_{i,1}, C_{i,2})$, $y_i = z_i$, $t = \delta k(m)$, and the b_i's in CER are the same as the b_i's in the definition. Suppose that P makes no extra oracle queries. In this case P fools V in round i with probability $(1 - \tau(n)/2n)$ and the probability that P fools V in all $\delta k(m)$ rounds is $(1 - \tau(n)/2n)^{\delta k(m)}$. The knowledge error $\kappa(\alpha)$ in this case equals $(1 - \tau(n)/2n)^{\delta k(m)}$. Note that n being the product of two different primes implies that $\tau(n)/n > 1/(3lnln\ n)$, and this implies that $(1 - \tau(n)/2n)^{\delta k(m)} \leq (1 - 1/(6lnln\ n))^{\delta k(m)}$. This can be shown to be at most $2^{-2k(m)}$ from the following fact:

For all real numbers t and θ, s.t. $\theta \geq 1$ and $|t| \leq \theta$, $(1 + t/\theta)^{\theta} \leq e^t$

Now, suppose that P makes $T(m)$ oracle queries (e.g., to try to fix the first several challenge bits to 0). It can be shown that the knowledge error is at most

$T(m)2^{-2k(m)}$, which for sufficiently long m is at most $2^{-k(m)}$. It follows that $p(\alpha)$ is at least $1 - 2^{-k(m)}$.

Let $T = P_{\alpha,\beta,r}(H)$ and $T' = P_{\alpha,\beta,r}(H')$ Consider the following knowledge extractor K. Suppose that in round i, b_i in T is 0 and b_i in T' is 1. K then adds the w_i in T to the w_i in T' to get d_c, a candidate decryption exponent. The operation of K when the bits are inverted is similar.

We will now give a lower bound on K's probability of extracting a witness from T and T'[1]. Assuming no extra oracle queries are made, with probability $1/2$ we have that b_i in T equals b_i in T', since the b_i's are chosen randomly for both T and T' (H and H' serve as honest verifiers). So, with probability $1/2$, the b_i's in each transcript differ. Also, both transcripts will use the same t_i's and with probability $\tau(n)/n$, t_i has maximal order. To see this recall that P for both transcripts is using: the same common-input α, the same auxiliary input β, and *the same random tape* r. Thus, with probability $1-\tau(n)/2n$ a decryption exponent is not extracted in round i. So, the probability that a decryption exponent isn't extracted in any of the $\delta k(m)$ rounds is $(1 - \tau(n)/2n)^{\delta k(m)}$, which is at most $2^{-2k(m)}$. Now consider the case where P makes $T(m)$ additional oracle queries. It can be shown that the probability that a decryption exponent isn't extracted is at most $T(m)2^{-2k(m)}$. For sufficiently long m this probability is at most $2^{-k(m)}$. Thus, the probability that a valid decryption exponent is extracted by K is at least $1-2^{-k(m)}$. Using this worst case value, and the worst case values for $p(\alpha)$ and the knowledge error, it follows that the inequality in the validity condition evaluates to $0 \geq -2^{-k(m)}$. Thus, the validity condition is satisfied. It follows that the non-interactive CER protocol is a proof of knowledge. QED.

Note that all the above steps are efficient (small polynomial overhead, which is reasonable for a one-time operation such as key (re)-registration). So, what we achieve is:

Theorem 1. *There exists a generic and efficient auto-recoverable auto-certifiable system, where the users' keys are based on the RSA/factoring system assuming only the security of RSA/factoring.*

5 Decrypting Individual Messages

In some settings, particularly in law-enforcement settings, it is important to provide the escrow authorities with the ability to decrypt individual messages of users without revealing the users escrowed private key. This is important, since the sender's private key should not be opened if the receiver is under suspicion; this issue is not dealt in systems like the Fair cryptosystems. This mode of operation is possible in our solution by making only minor changes in the system (to the organization of the escrow authorities).

Suppose for example that there are three escrow authorities with three separate semantically secure public encryption functions. The user in this system

[1] We won't derive an exact probability, since using $T(m)$ oracle queries, P may always try to make the first several t_i's not have order $\lambda(n)$ to trick the prover, for example.

generates three separate public keys, and then escrows each of them using the auto-recoverable auto-certifiable cryptosystem corresponding to each of the public encryption functions of the escrow authorities. The three public keys are verified and published. Suppose that the users use a hybrid cryptosystem to send confidential messages. The sender simply encrypts the session key using all three public keys. The escrow authorities can recover the message without any of them knowing all three of the corresponding private keys.

Alternate methods also exist for composite public keys. One idea is as follows. Now, suppose that there are three escrow authorities. The user can secret split d additively to get d_1, d_2, and d_3 in conjuction with the auto-recoverable auto-certifiable cryptosystem. It follows that with such additive values for d escrowed, the escrow authorities can perform shared decryption of a message encrypted with n without revealing the user's private key. What we then get is:

Theorem 2. *The generic RSA-based auto-recoverable cryptosystem presented here can be used to decode messages securely by the escrow agents, without revealing the private key of the receiver.*

6 Arbitrary Depth Escrow Hierarchy

Consider a three level key escrow system. The hierarchical tree consists of the escrow authorities at the top, subordinate escrow authorities at the middle level, and users at the bottom. The authorities at the top publish an escrowing public key E. Then each of the subordinate escrow authorities auto-certifies a unique public key using E. Finally, the users assigned to a given subordinate escrow authority generate public keys and auto-certify them using the escrowed public key of their corresponding subordinate escrow authority.

In the special case of the composite based auto-recoverable solution, it is necessary that the modulus at each level fit within the message space of ENC at the level above. We can therefore easily implement a depth 3, 4, or 5 hierarchy in a straightforward fashion.

Theorem 3. *The generic auto-recoverable cryptosystems for factoring/RSA based keys presented here can be used to efficiently implement an arbitrary depth hierarchical key escrow system.*

7 Smaller Certificate of Recoverability

Recall that a safe prime p is a prime of the form $p = 2p_1 + 1$ where p_1 is prime. The CA storage can be reduced as follows. We insist that the user prove that n is the product of two safe primes. This can be done using [CM99]. We remark that this proof can be made non-interactive. Also, in each iteration t_i is computed such that $J(t_i/n) = -1$ by making successive oracle queries. This fact is verified for each t_i by the verifier.

Having done this it then follows that each round in the certificate will transfer d with knowledge error negligibly larger than $1/2$. To see this note that the only

possible orders for t_i are $(2, 2p_1, 2q_1, 2p_1q_1)$ where $q = 2q_1 + 1$. This follows from the fact that t_i is checked by the verifier to be a quadratic non-residue mod n. If the order is $2p_1$ then [ML98] can be used to factor n (the same holds for $2q_1$). This can be seen from the fact that $gcd((t_i^2 \ mod \ n) - 1, n)$ is a non-trivial factor of n for such a t_i. If t_i has order $2p_1q_1$ then round i transfers $a + (d - a) \ mod \ \lambda(n)$. Thus, for all possible orders of t_i, d is transferable in round i (except if $ord_n(t_i) = 2$ which is an event having negligible probability). This combined with the knowledge error due to the challenge bit, achieves the stated knowledge error of 1/2. This implies that the number of iterations can be the same as in the NIZK proof in [BR94].

References

[ASW98] N. Asokan, V. Shoup, M. Waidner. Optimistic Fair Exchange of Digital Signatures. In *Advances in Cryptology—Eurocrypt '98*, pages 134–148. 329

[BFL91] J. Boyar, K. Friedl, C. Lund. Practical zero-knowledge proofs: Giving hints and using Deficiencies. In *Journal of Cryptology*, 4(3), pages 185–206, 1991. 332

[BG96] M. Bellare, S. Goldwasser. Encapsulated Key Escrow. manuscript, 1996. 332

[BR94] M. Bellare, P. Rogaway. Random Oracles are Practical In *ACM CCCS '94*. 331, 335, 338, 340, 341

[BT99] F. Boudot, J. Traore. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery In ICICS '99. 329

[Ch98] L. Chen. Efficient Fair Exchange of Verifiable Confirmation of Signatures. In *Advances in Cryptology—Asiacrypt '98*. 329

[CFN] D. Chaum, A. Fiat, M. Naor. Untraceable Electronic Cash. In *Advances in Cryptology—Crypto '88*, pages 319–327. 332

[CFT98] A. Chan, Y. Frankel, Y. Tsiounis. Easy Come - Easy Go Divisible Cash. In *Advances in Cryptology—Eurocrypt '98*, pages 561–575. 329

[CGH98] R. Canetti, O. Goldreich, S. Halevi. The Random Oracle Methodology, Revisited. In *ACM STOC '98*. 331

[CGMA] B. Chor, S. Goldwasser, S. Micali, B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In *FOCS '85*. 328

[CM99] J. Camenish, M. Michels. Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In *Advances in Cryptology—Eurocrypt '99*. 337

[CMS96] J. Camenish, U. Maurer, M. Stadler. Digital Payments Systems with Passive Anonymity Revocation Trustees. In *Esorics '96*. 329

[CS97] J. Camenish, M. Stadler. Efficient Group Signatures. In *Advances in Cryptology—Crypto '97*, pages 410–424. 328, 329

[De88] Yvo Desmedt. Abuses in Cryptography and How to Fight Them. In *Advances in Cryptology—CRYPTO '88*. 332

[DDFY] A. De Santis, Y. Desmedt, Y. Frankel, M. Yung. How to Share a Function Securely. In *ACM STOC '94*, pages 522–533. 331

[Fe85] P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In *FOCS '87*. 328, 332

[FGMY] Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung. Optimal Resilience Proactive Public Key Systems. In *FOCS '97*. 331

[FGY] Y. Frankel, P. Gemmell, M. Yung. Witness based Cryptographic Program Checking and Robust Function Sharing. In *ACM STOC '96*. 331

[FO98] E. Fujisaki, T. Okamoto. A Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications. In *Advances in Cryptology—Eurocrypt '98*, pages 32–46. 329

[FPS99] P. Fouque, G. Poupard, J. Stern. Recovering Keys in Open Networks. In IEEE ITW, 1999. 329

[FS86] A. Fiat, A. Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology—Crypto '86*, pages 186–194. 331

[FTY96] Y. Frankel, Y. Tsiounis, M. Yung. Indirect Discourse Proofs: Achieving Efficient Fair Off-Line Cash. In *Advances in Cryptology—Asiacrypt '96*. 329

[FY93] M. Franklin, M. Yung. Towards Provably Secure Efficient Electronic Cash. In *ICALP '93*. 332

[GHY] Z. Galil, S. Haber, M. Yung. Minimum-knowledge Interactive Proofs for Decision Problems. In *SIAM J. of Computing*, (4), pages 711–739, 1989. 332

[GJKR] R. Gennaro, S. Jarecki, H, Krawczyk, T. Rabin. Robust and Efficient Sharing of RSA. In *Advances in Cryptology—Crypto '96*. 331

[GP87] J. van de Graaf, R. Peralta. A simple and secure way to show the validity of your public key. In *Advances in Cryptology—Crypto '87*, pages 128–134. 332

[IY86] R. Impagliazzo, M. Yung. Direct Minimum-Knowledge Computations. In *Advances in Cryptology—Crypto '86*. 332

[KL95] J. Kilian, F.T. Leighton. Fair Cryptosystems Revisited. In *Advances in Cryptology—Crypto '95*, pages 208–221. 328

[KMO89] J. Kilian, S. Micali, R. Ostrovsky. Minimum-Resources Zero-Knowledge Proofs. In *FOCS '89*. 332

[KP98] J. Kilian, E. Petrank. Identity Escrow. In *Advances in Cryptology—Crypto '98*, pages 169–185. 329

[LS] M. Liskov, R. D. Silverman. A Statistical Limited-Knowledge Proof for Secure RSA Keys. Submited to the IEEE P1363 Working Group. Available at http://grouper.ieee.org/groups/1363/contrib.htm. 332

[Man97] A. Young, M. Yung. Manuscript related to the current work dated Sept. '97 available from the authors (preliminary version also submitted to Eurocrypt '99.) 329

[Mi92] S. Micali. Fair Public-Key Cryptosystems. In *Advances in Cryptology—Crypto '92*, pages 113–138. 328, 329

[ML98] W. Mao, C. H. Lim. Cryptanalysis of Prime Order Subgroups of Z_n^*. In *Advances in Cryptology—Asiacrypt '98*, pages 214–226. 338

[PY] P. Paillier and M. Yung. Self-Escrowed Public-Key Infrastructures. (Manuscript).

[PS96] D. Pointcheval, J. Stern. Security Proofs for Signature Schemes. In *Advances in Cryptology—Eurocrypt '96*. 331, 340, 341

[PS00] G. Poupard, J. Stern. Short Proofs of Knowledge of Factoring In These proceedings. 329

[PS-L] G. Poupard, J. Stern. Talks at Luminy, Oct. '99. 329

[RSA78] R. Rivest, A. Shamir, L. Adleman. A method for obtaining Digital Signatures and Public-Key Cryptosystems. In *Communications of the CACM*, 21(2), pp. 120–126, 1978. 326

[Sch99] B. Schoenmakers. A simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting. In *Advances in Cryptology—Crypto '99*, LNCS 1666, 148-164. 329

[Sc91] C. P. Schnorr. Efficient Signature Generation for Smart Cards. In Journal of Cryptology, 4 (3), pages 161–174, 1991. 331

[St96] M. Stadler. Publicly Verifiable Secret Sharing. In *Advances in Cryptology—Eurocrypt '96*,pages 190–199. 328
[Ve00] E. Verheul, Certificates of Recoverability with Scalable Recovery Agent Security. In These Proceedings. 329
[YY96] A. Young, M. Yung. The Dark Side of Black-Box Cryptography, In *Advances in Cryptology—Crypto '96*. 332
[YY98] A. Young, M. Yung. Auto-Recoverable and Auto-Certifiable Cryptosystems. In *Advances in Cryptology—Eurocrypt '98*. 327, 328, 329
[YY99] A. Young, M. Yung. Auto-Recoverable Cryptosystems with Faster Initialization and The Escrow Hierarchy. In *PKC '99*. 328, 329

A Appendix: Definitions

We employ the RSA function and the notion of semantic security in our proposed system. In public-key systems, the security of the encryptions of preimages of public one-way function values is equivalent to the notion of polynomial-security where a challenge of two messages is given, only one of which is the "real message", and where no one can tell which one is the actual message. In our applications however, we will encrypt a value for which there is a "public commitment" which is related to a public key. In such cases we could not withstand a challenge, since the public commitment is done via a one-way function of the message. However, semantic security still holds. Intuitively, this means that the added encryption does not help beyond what is known from the public commitment.

In [BR94], it was stated that a formalization of what it means for a NIZK proof to be a proof of knowledge was forthcoming. We know of no such formalization to date anywhere in the literature, so we will present a formalization here to prove security in the random oracle model.

To define a "proof of knowledge" in the random oracle model, one can apply the probabilistic notions in [PS96]. Informally, we define an extractor which invokes the prover on two transcripts T and T', which are initially the same but which are extended identically using random oracles which are only "polynomially" different. That is, the oracle entries used in extending T' are random and with high probability distinct from the corresponding entries in the oracle used to construct T. The forking argument of [PS96] can be cast into an extractor argument: a proof system is a proof of knowledge if there exists a knowledge extractor that when given access to a prover which constructs proofs in both extentions, is able to extract a witness with probability greater than or equal to the probability of P convincing V that P knows the witness minus the knowledge error.

Definition 2. *Denote by* $P_{\alpha,\beta,r}(H) = (\alpha, x_1, x_2, ..., x_t, b_1, b_2, ..., b_t, y_1, y_2, ..., y_t)$ *the message sent by machine* P *with common-input* α*, auxiliary-input* y*, and random input* r *when given access to random oracle* H*.* H *is random with the restriction that* $H(x_1, x_2, ..., x_t) = (b_1, b_2, ..., b_t)$*. Here* $x_1, x_2, ..., x_t$*,* $b_1, b_2, ..., b_t$*, and* $y_1, y_2, ..., y_t$ *are strings. The function* $P_{\alpha,\beta,r}$ *is called the transcript specification function of machine* P *with common-input* α*, auxiliary input* β*, random input* r*, and access to a random oracle.*

Definition 3. *Define the random oracle randomization operation, ROR, to be the following.* $ROR(H, x_1, x_2, ..., x_t) = H'$, *where* $H, H' \in 2^\infty$ *and* $x_1, x_2, ..., x_t$ *are strings.* H *and* H' *are identical random oracles except that* $H'(x_i) \in_R \{0,1\}^\infty$ *for* $1 \leq i \leq t$.

We can (w.v.h.p) choose the range values of H' randomly without causing conflicts in the entries which H' and H share since the set from which the range of a an oracle is drawn is uncountable (whereas the tables for H and H' are only countably infinite).

Definition 4. *Let* R *be a binary relation, and let* κ *be a function from the natural numbers to [0,1]. Denote by* $p(\alpha)$ *the probability that the machine* V *accepts on input* α, *when interacting with the prover specified by* $P_{\alpha,\beta,r}$. *Let the symbol* $\perp$ *denote failure to find* $\beta \in R(\alpha)$. *We say that a function* V *is a knowledge verifier for the relation* R *in the random oracle model with knowledge error* κ *if the following two conditions hold.*

1. *Non-triviality: There exists an interactive function* P *so that for every* $(\alpha, \beta) \in R$, *and for all* $H \in 2^\infty$, *all possible messages sent from* P *to* V *on common-input* α *and auxiliary input* β *are accepting.*
2. *Validity (with error* κ*): We say that the verifier* V *satisfies* validity with error κ *if there exists a probabilistic expected polynomial-time oracle machine* K *such that for every interactive function* P, *every* $\alpha \in L_R$, *for every* $H \in 2^\infty$, *it is the case that when* K *has access to* $P_{\alpha,\beta,r}(H)$ *and* $P_{\alpha,\beta,r}(H')$ *such that* $H' = ROR(H, x_1, x_2, ..., x_t)$, K *outputs an* $s \in R(\alpha) \cup \{\perp\}$, *and*

$$Pr\{K \text{ outputs an } s \in R(\alpha)\} \geq p(\alpha) - \kappa(|\alpha|).$$

We call such an oracle machine K a random oracle knowledge extractor. The reader may be wondering why we insisted on using the ROR operation in our definition, since no ROR operation was used in the definition of the forking lemma in [PS96]. In our opinion, this is a minor oversight in the formal definition of the forking lemma. To see this, note that the forking lemma assumes that the machine that replays the oracle machine A (A is a no-message attacker against the signature scheme) replays it with "a different" random oracle. Thus, it is not clear which signature algorithm should be used in practice, since the proof of security assumes access to both signature algorithms (i.e., the signature algorithm with the original oracle and the signature algorithm with the 'different' oracle), since clearly both cannot be used to sign a given message.

In the definition of a non-interactive proof being ZK in [BR94], it is made very clear that a random oracle completion operation, ROC, is used to insure that the oracles are *only polynomially different*. As such, we may use one oracle in practice, and the fact that the oracle may have polynomially many "faults" is intractable to detect. By faults we mean entries in the infinite table that are defined to have two values, when only one string from $\{0,1\}^\infty$ is allowed.