An Identity Escrow Scheme with Appointed Verifiers

Jan Camenisch
IBM Research
Zurich Research Laboratory
CH–8803 Rüschlikon
jca@zurich.ibm.com

Anna Lysyanskaya
MIT LCS
545 Technology Square
Cambridge, MA 02139 USA
anna@theory.lcs.mit.edu

Abstract. An identity escrow scheme allows a member of a group to prove membership in this group without revealing any extra information. At the same time, in case of abuse, his identity can still be discovered. Such a scheme allows anonymous access control. In this paper, we put forward the notion of an identity escrow scheme with appointed verifiers. Such a scheme allows the user to only convince an appointed verifier (or several appointed verifiers) of his membership; but no unauthorized verifier can verify a user's group membership even if the user fully cooperates, unless the user is completely under his control. We provide a formal definition of this new notion and give an efficient construction of an identity escrow scheme with appointed verifiers provably secure under common number-theoretic assumptions in the public-key model.
Keywords. Identity escrow, group signatures, privacy protection, formal model for group signatures.

1 Introduction

As digital communication becomes the preferred means of information exchange, it becomes ever easier for those of questionable motivation to mine the accumulated data. Under these circumstances, both the importance and the challenge of protecting the privacy of individuals grow considerably. A number of cryptographic protocols that limit the information dispersed from accumulated data have been proposed. These are, for instance, anonymous voting protocols [5, 30], anonymous payment schemes [6, 18], and credential systems [9, 16]. All these systems follow the principle of data minimization, i.e., a participant in the system can only learn as much information about the other participants as is necessary for the system to function properly.

In this context, group signatures [2, 12, 19] are an important building block. They allow a member of some group to sign anonymously on the group's behalf. Thus, a party receiving a signature can be sure that its originator is a member of the group, but receives no other information. However, in exceptional cases such as when the anonymity is misused and a legal dispute arises, a designated revocation manager has the power to reveal the unambiguous identity of the originator of the signature. At the same time, no one can misattribute a valid group signature. A concept dual to group signature schemes is that of identity escrow [32] schemes. They can be seen as group-member identification schemes with revocable anonymity. In fact, any group signature scheme can be turned into an identity escrow scheme and vice versa.

Group signatures can, for instance, be used by the purchasing department of a company to hide the internal structure of this department. All members of the department form a group, and sign all purchasing orders using group signatures. In case one day a sports car gets delivered instead of pencils, the department manager will be able to identify the culprit. Recently, group signature schemes were used to realize an anonymous credential system [9]. Here, being a member of some particular group meant possessing a particular credential. Hence, ownership of a credential can be proved anonymously. Other applications include bidding [20], electronic cash [33], and anonymous fingerprinting [7].

Group signature/identity escrow schemes with appointed verifiers, as proposed in this paper, go a step further: here a group member can prove his membership only to an appointed verifier but not to anyone else. There can be several different appointed verifiers for each member. This property of not being able to convince non-appointed parties is similar to receipt-freeness in electronic voting schemes, where a voter must not be able to prove to anyone how she voted, which is required to hinder vote-buying. We stress that this is different from the situation with so-called confirmer signatures [17] or designated-verifier proofs [31], where although signatures (resp., proofs) can only be verified by a designated party, the signer (resp., prover) would have the power to issue a signature (resp., proof) that is universally verifiable.

Appointed verifiers are useful for many applications of group signature and identity escrow schemes. As an example, consider a bank that issues a credential stating that the customer is eligible for a small business loan. The bank might want to have a guarantee that the customer cannot use this credential in order to obtain a better loan from a competing bank; or to use the loan money for something other than the business for which it was granted. Or, consider the purchasing department scenario outlined above. Naturally, different members of the department are authorized to conduct different kinds of transactions. Using a group signature scheme with appointed verifiers allows the department manager to ensure that employees can only order from the companies they are authorized. Finally, consider their use in a credential scheme. It is natural that for some types of credentials the user should not be able to show them to anyone except the intended verifier. This can be useful in preventing abuse of credentials as well as in controlling who can get to know to whom the credentials were issued.

Let us loosely outline how our identity escrow (group signature) scheme with appointed verifiers is constructed. To this end we first explain how efficient and provably secure group signature schemes [2, 12] are realized. The public key of the group can be viewed as a public key of a suitable signature scheme. The group manager holds the secret key corresponding to this public key. To become a group member, a user chooses, as membership secret key, an element of a certain (algebraic) group. The user's identifier is computed as a one-way function of this key, for example through exponentiation in a group where computing the discrete logarithm is conjectured to be hard. The group manager signs (certifies) this identifier and sends back the signature to the new group member. This signature is the user's group membership certificate. To convince a verifier of her group membership, a user proves in zero-knowledge that she knows a membership certificate and the corresponding membership secret key. In case of a group signature scheme, this proof is turned into a signature scheme using the so-called Fiat-Shamir heuristic [26]. An identity escrow scheme constructed in this way is provably secure as long as the underlying signature scheme is secure. The corresponding group signature scheme is provably secure in the random oracle model. The challenge in designing an efficient identity escrow or group signature scheme is finding a signature scheme for the group manager and a format for membership secret keys and corresponding identifiers such that the proof of membership is efficient.

To extend such a scheme to an identity escrow system with appointed verifiers, we will have the group manager split the group membership certificate into two pieces. The first piece will be handed over to the user. The second piece will be encrypted under the appointed verifier's public key. It will be easy to fake a tuple that looks like the first piece of the membership certificate and the encryption of the second piece. Only the appointed verifier, under whose public key the encryption is carried out, will be able to verify that a given ciphertext corresponds to the second piece of a user's certificate. Together, the two pieces constitute an unforgeable group membership certificate. To prove group membership to the appointed verifier, the user could prove possession of his piece of the membership certificate as before, and then give the verifier a blinded version of the encrypted piece.

An adversary in this system can try to induce some verifier to accept an invalid user; or he can try to make it look as though some honest user participated in a shady transaction; or he can conduct a shady transaction and then try to avoid anonymity revocation; or he can try to convince another adversary, who is not an authorized verifier, that some user is a group member. We provide a formal definition of security against such attacks. For the first time, a formal model for identity escrow schemes along the lines of an ideal world specification, is given. These new identity escrow/group signature specifications are more rigorous than the ones that exist to date. As they are similar to the definitions from the multi-party computation literature [13, 14, 37], they integrate with this literature better than previous specifications did, and so such properties as composability of protocols can be better understood in this framework (but we do not address them here). Finally, we formally define the appointed-verifier property, i.e., the property that no proof system (A, B) exists in which A is a group member, B is not the appointed verifier, and yet A acts as a prover and B as a verifier for the statement that A is a group member, and the gap between the completeness and soundness of the system is non-negligible. Cryptographic problems of this flavor have not been sufficiently explored. While receipt-free voting is a relatively well-studied example [5, 30], no formal definition of receipt-freeness has been given, and it is not well understood what gap between completeness and soundness for the adversary-verifier in receipt-free voting is satisfactory. Thus, we are the first to explore this in a formal way and to obtain a scheme that satisfies our strong and relatively natural definition.

We prove that, under the strong RSA assumption, the decisional composite residuosity assumption, and the decisional Diffie-Hellman assumptions, our scheme is secure and has the appointed verifier property.

2 The Model

In this section, we define an ideal identity escrow scheme with appointed verifiers. Here, an ideal trusted third party takes care of the proper functionality of the system. Our model captures all the properties of previous ones (without appointed verifiers) in a natural way. We then define what it means for a real system to match this specification. We define the system with one group and one revocation manager; extending it to multiple ones is straightforward. Extending the model to group signatures can be done as well.

The Ideal System. The ideal system, the functionality of which is ensured by an ideal trusted party T, is as follows:
Ideal parties: The trusted party T, the group manager M, a set of users $\mathcal{U}$, a set of verifiers $\mathcal{V}$, and the anonymity revocation manager R.
Ideal communication: All communication is routed through T. If the sender of a message wishes to be anonymous, he requests that T not reveal his identity to the recipient. Finally, a sender of a message may request that a session, i.e., a block of messages, be established between him and the recipient. This session then gets a session id *sid*.
Ideal operations for a general identity escrow scheme:

Join. This operation is a session between a user U and the group manager M. M tells T that it wants user U to become a member of the group. The user confirms that he wants to be a member. Upon receiving this messages from M and U, T sends a key K_U to U for further transactions related to his group membership; he also notifies M of the success of the transaction.

Authenticate. This operation is a session between a user and a verifier V. The user must send a tuple (K,*sid*,V,*con*) to T, where K denotes a key, *sid* denotes a session id, V is the name of the verifier, and *con* is the condition under which the identity of the participating user can be established. T verifies that K is a key that corresponds to some group member (not necessarily the user from whom the request originates). If so, T tells the verifier V that the user with whom the verifier has session *sid* running is a member of the group. V then either accepts or rejects, and forwards his reply to T. (If T receives no reply that is equivalent to rejecting.) T then notifies the user of the verifier's output.

Identify. This operation is a session between the revocation manager R and the verifier V. V submits a tuple (*sid*, *con*) to T and to R. R asks T to confirm that *sid* was an Authenticate operation with revocation condition *con*. Then R may ask T to reveal to R the identity of the user who participated in session *sid*. Finally, R may ask T to reveal the user's identity to V.

Ideal operations for an appointed-verifier identity escrow scheme:

Join with appointed verifier. This operation is a session between a user U and the group manager M. As a result, M tells T that user U's membership can be confirmed to verifier V. The user receives a key K_U from T for further transactions related to authenticating his group membership to V.

Authenticate to appointed verifier. This is the same as in the general scheme, except that T will only carry this out with the appointed verifier V.

Convert. This operation is between a user and the appointed verifier V. V tells T that the user is now authorized to demonstrate group membership to other verifiers. T notifies the user of that fact.

Authenticate. This is the same as in the general scheme, except that T will only carry this out if the user is authorized to demonstrate group membership to all verifiers.

Identify. This is the same as in the general scheme.

Inputs and outputs of the ideal players: The ideal players are interactive probabilistic Turing machines. Prior to initiating a transaction, a player receives an input that tells it to do so. These inputs are produced externally. At the end of the lifetime of the system, each player outputs a list of interactions in which this player has participated and their outcome (success/failure).

The Real System. We make the following assumptions on the communication in the real-system: We are in the public-key model, i.e., each user has carried out a proof of knowledge of his secret key at the beginning of the lifetime of the system. It is possible to establish a session between an anonymous user and a verifier (in practice, this can be achieved by a so-called mix-network [15] or by onion-routing protocols [29]. The information transmitted over a channel cannot later be retrieved by some physical means (i.e., it does not stick around in routers and caches). This is necessary to make sure that one cannot demonstrate that one sent or received a given message. This can also be achieved in conjunction with the methods to get anonymous communication, e.g., by requiring the hosts to delete all processed data. The real system is implemented by cryptographic protocols.

Security vs. Appointed-Verifier Property. The usual way of defining security of a real system is to restrict the power of the real-world adversary to the power of an adversary that controls the same set of players in the ideal system. Security in this sense is exhibited by providing a simulator that translates the real-world adversary into one in the ideal world. Here, in addition to providing security in this sense, we have to also allow for the case where there are two adversaries, such that one is trying to convince the other of his relationship with other players. Therefore, two security properties must be satisfied.

Protecting the Honest Players. First, we have to guarantee simulator-based security for the honest parties.

The ideal-world (resp., real-world) adversary is a probabilistic polynomial-time Turing machine that can control some subset of ideal (resp., real) parties and participate in transactions on their behalf. In addition, the adversary controls the environment, i.e., he either explicitly gives input to other players as to the transactions to be carried out, or he specifies the probability distribution on these inputs.

At the end of the lifetime of the system, each player outputs the entire list of interactions in which this player has participated and their outcome (success/failure).

Let the ideal system be called *IS*, and its cryptographic implementation be called *CS*. Let $p = \texttt{poly}(k)$ be the number of players in the system with security parameter k. Let Z_i denote the output of the i-th player. In the real world, a public-key infrastructure has been securely set up (i.e., each party has produced a public key and proved knowledge of the corresponding secret key). Let P denote its public information; let a denote the collection of dishonest players' secret keys. (In case we are working in the absence of the public-key model, these are empty strings.) An identity escrow scheme is secure if the adversary $\mathcal{A}$ cannot distinguish whether he is interacting with the real-world honest players, or if in fact the system is implemented in the ideal world (so all the honest players are shielded because T protects them) and he is just interacting with a simulator. More formally, with "$D_1(1^k) \stackrel{c}{\approx} D_2(1^k)$" denoting the computational indistinguishability of the distributions D_1 and D_2:

Definition 1 (Secure identity escrow scheme). *CS is* secure *if there exists a simulator $\mathcal{S}$ (ideal-world adversary) such that for all interactive probabilistic polynomial-time real-world adversaries $\mathcal{A}$, for all sufficiently large k, we have:*

- *In the IS, $\mathcal{S}$ controls the same set of players as $\mathcal{A}$ does in CS.*
- *The inputs given by $\mathcal{S}$ to the ideal-world players are identical to those given by $\mathcal{A}$ to the real-world players.*
- *For all P,*

$$(\{Z_i^{CS}(1^k, P, s_i)\}_{i=1}^{p}, \mathcal{A}(1^k, P, a)) \stackrel{c}{\approx} (\{Z_i^{IS}(1^k, P, s_i)\}_{i=1}^{p}, \mathcal{S}^{\mathcal{A}}(1^k, P, a)) \ ,$$

where $\mathcal{S}$ is given black-box *access to $\mathcal{A}$.*

Comparison with previous models. It is easy to see that this ideal model captures the requirements *correctness*, *anonymity*, *unlinkability*, *traceability*, *exculpability/framing*, and *coalition-resistance* of previous models (e.g., [2]), i.e., that the trusted party T ensures them.

No Benefits for Dishonest Players that Mistrust Each Other. Informally, an identity escrow scheme is *appointed-verifier* if only the appointed verifier can be persuaded that a user is a member of the group. A formal definition is more complex. Formally, we have two adversaries, $\mathcal{A}$ and $\mathcal{B}$, and $\mathcal{A}$ tries to convince $\mathcal{B}$ that some player A it controls is a group member, even though $\mathcal{B}$ does not control the appointed verifier V. The appointed verifier property of the scheme makes it impossible for any proof system $(\mathcal{A}, \mathcal{B})$, where $\mathcal{A}$ acts as prover and $\mathcal{B}$ as verifier, to have a non-negligible gap between the completeness and the soundness of the system. However, in defining this property, we have to take into account that (1) $\mathcal{B}$ can apply to V to tell him whether a given user is a group member; and (2) $\mathcal{B}$ can become convinced of the truth of the statement by means that are independent on the system's implementation: for example, if A is the only user in the system, and V flashes a green light every time it recognizes a group member. Thus, a formalization of the appointed verifier property is bound to be technically involved.

The approach we will take to defining it is as follows: we will require that for any $\mathcal{A}$, there exists an efficient $\mathcal{D}$ such that whenever $\mathcal{A}$ can convince $\mathcal{B}$ that A has group membership with appointed verifier V, $\mathcal{D}$ can convince $\mathcal{B}$ of the same statement without access to group manager's M's messages pertaining to the corresponding *Join* operation. We will call $\mathcal{D}$ *the deceiver*, because it can deceive any verifier $\mathcal{B}$. However, $\mathcal{D}$ is not responsible if $\mathcal{B}$ has other ways, implementation-independent, of getting convinced. That is why, in the definition, we need an additional efficient machine, $\mathcal{F}$, called *the filter*, which sets up the relevant group membership on behalf of A, but shields $\mathcal{D}$ from this information. $\mathcal{F}$ guarantees that group manager M and verifier V have the same view whether $\mathcal{A}$ has a valid membership certificate or one faked by $\mathcal{D}$. Intuitively, if $\mathcal{B}$ cannot distinguish whether he is talking to $\mathcal{A}$, or to the deceiver $\mathcal{D}$, but can still tell whether or not A is a group member with appointed verifier V, then $\mathcal{B}$'s way of telling is implementation-independent, and arises from the way other parties, such as M and V, behave. We now proceed to formalize this idea.

Let $\mathcal{A}$ and $\mathcal{B}$ be the two adversaries, modeled by probabilistic polynomial-time interactive Turing machines. Let *ES* denote an event sequence in the cryptographic identity escrow scheme. We write C^{ES} for a machine C to denote the fact that these events may be scheduled one-by-one, maybe even by an adversary. Let P denote the public information of the public-key infrastructure. Let a denote the set of secret keys of the players controlled by $\mathcal{A}$. Let a' be an additional input to $\mathcal{A}$. By $A \in \mathcal{A}$ we denote that A is a player controlled by adversary $\mathcal{A}$. Let $L \subseteq \{(A, V) \ : \ A \in \mathcal{A}, V \notin \mathcal{A}, V \notin \mathcal{B}\}$ be a list of user-verifier pairs that is given as a challenge to $\mathcal{B}$. We say that such a list L is *good* for *ES*, $\mathcal{A}$ and $\mathcal{B}$, if in the sequence of events specified by *ES*, for all $(A, V) \in L$, V never performs the *Convert* and *Identify* operations for A and $\mathcal{B}$ and V have not engaged in the *Authenticate with appointed verifier* protocol in which V accepted such that a subsequent *Identify* operation, if carried out, will point to A.

Let $\mathcal{F}^{ES}(1^k, P, L, a, \mathit{mode})$, $\mathcal{D}^{\mathcal{A},\mathcal{F}}(1^k, P, L, a, \mathit{mode})$ be interactive Turing machines. The *mode* part of their input specifies their behavior as follows: There are two modes of operation, the *real* mode and the *fake* mode. In the *real* mode, $\mathcal{F}$ passes the messages received from *ES* on to $\mathcal{D}$, which in turn passes them on to $\mathcal{A}$. If $\mathcal{A}$ sends any messages to $\mathcal{B}$, $\mathcal{D}$ faithfully passes them.

In the *fake* mode $\mathcal{F}$ behaves as follows: If a session *sid* is a *Join with appointed verifier* of user $A \in \mathcal{A}$ for verifier V, where $(A, V) \in L$, then $\mathcal{F}$ does not pass $\mathcal{A}$'s messages for *sid* to M, and does not forward M's replies to $\mathcal{A}$ for this *sid*. Instead, $\mathcal{F}$ carries out the *Join* operation himself, on behalf of $\mathcal{A}$, possibly guided by additional input from $\mathcal{D}$. It then notifies $\mathcal{D}$ whether this *Join* was successful. If a session *sid* is an *Authenticate to appointed verifier* between A and V such that $(A, V) \in L$ and the corresponding *Join* has taken place, then $\mathcal{F}$ does not pass $\mathcal{A}$'s messages for *sid* to V, and does not forward V's replies to $\mathcal{D}$ for this *sid*. Instead, $\mathcal{F}$ carries out the *Authenticate* operation himself, on behalf of $\mathcal{A}$, possibly guided by additional input from $\mathcal{D}$. It then notifies $\mathcal{D}$ whether this *Authenticate* was successful. For all other sessions, $\mathcal{F}$ just passes all the messages to and from $\mathcal{D}$.

In the *fake* mode, $\mathcal{D}$ behaves as follows: For a session *sid* of *Join with appointed verifier* for user A and verifier V where $(A, V) \in L$, $\mathcal{D}$ will create fake messages and send them to $\mathcal{A}$ in place of the group manager's messages. For a session *sid* of *Authenticate to appointed verifier* $\mathcal{D}$ will decide whether this session is between user A and verifier V, $(A, V) \in L$. In case it is, $\mathcal{D}$ notifies $\mathcal{F}$, and possibly sends it additional information. $\mathcal{D}$ will then create messages to $\mathcal{A}$ in place of V's responses. For all other sessions, $\mathcal{D}$ passes all the messages to and from $\mathcal{A}$.

We stress that $\mathcal{D}$ does not have the ability to reset $\mathcal{B}$.

Definition 2 (Appointed verifier property). *An identity escrow scheme has the appointed-verifier property if there exist polynomial-time algorithms $\mathcal{D}$, $\mathcal{F}$ as described above, such that for all probabilistic polynomial-time (in their first input) adversaries $\mathcal{A}$,$\mathcal{B}$, for all P, a', b, for all sequences of events in the system ES, and for all good lists L,*

$$\mathcal{D}^{\mathcal{A}(1^k,P,a,a'),\mathcal{F}^{ES,\mathcal{B}^{ES}(1^k,b)}(1^k,P,L,a,real)}(1^k,P,L,a,real) \overset{c}{\approx}$$
$$\mathcal{D}^{\mathcal{A}(1^k,P,a,a'),\mathcal{F}^{ES,\mathcal{B}^{ES}(1^k,b)}(1^k,P,L,a,fake)}(1^k,P,L,a,fake)$$

3 High-Level Presentation of our Construction

First, a public-key infrastructure is set up in which each user has a secret key x and, based on this secret, an identifier $\tilde{h}^x$, where $\tilde{h}$ is a generator of some group G. Other players in the system have their public keys set up as follows: The group manager's public key is a modulus $n = pq$ such that $p = 2p' + 1$ and $q = 2q' + 1$, and p,q, p' and q' are all prime numbers, and five quadratic residues modulo n, denoted $(a_0, a_1, a_2, a_3, a_4)$. (The length of n depends on the size of the group G.) Each verifier has a public key for the Paillier cryptosystem. A revocation manager R for this scheme will have a Cramer-Shoup public key in G. The specifics of how these keys are set up are described in Section 5.1.

For a user with secret key x, a group membership certificate for an appointed verifier V, will be a quin-tuple (s, Z, c, u, e) such that each of these values lies in the correct integer interval, $u^{2e} = (a_0 a_1^s a_2^x Z a_4^c)^2$ holds, and c is the encryption of the value $\log_{a_3} Z \bmod n$ under V's public key. We show that such a certificate is hard to forge under the strong RSA assumption [3, 11, 23, 27, 28] and the assumption that computing discrete logarithms modulo a modulus of this form is hard. On the other hand, if c is not an encryption of $\log_{a_3} Z \bmod n$, then this certificate is easy to forge (Lemma 3). As V is the only entity that can check this, under the assumption that the Paillier cryptosystem is semantically secure, this is the first key step towards obtaining the appointed verifier property (the other key step is discussed at the end of this section). The fact that c is included in the certificate implies security for the verifier against adaptive attacks even though the Paillier encryption scheme as such is not secure against these attacks[1]. This membership certificate is issued via a protocol (between the user and the group manager), that does not allow the group manager to learn x and s, but only $\tilde{h}^x$ and $a_1^s a_2^x \bmod n$. This protocol is described in detail in Section 5.2.

To prove group membership to V, the user blinds c to obtain c', and blinds Z to obtain Z' in such a way that, if c is the encryption of $\log_{a_3} Z$, then c' is the encryption of $\log_{a_3} Z'$. This is why we use the Paillier cryptosystem: the additive homomorphism property of the Paillier scheme is crucial for this step. c' and Z' are given to the verifier. Further, the user proves knowledge of a tuple (x, s, c, Z, u, e, r) such that (s, Z, c, u, e) is a group membership tuple for key x, and r is the randomizer used to blind (c, Z) to obtain (c', Z'). In addition, to enable anonymity revocation, the user provides an encryption E of his identifier $\tilde{h}^x$ under the anonymity revocation manager's public key and proves that E is a valid encryption of an identifier that is based on the same x as the group membership certificate. These proofs are done using efficient statistical zero-knowledge discrete-logarithm-based proofs of knowledge. The fact that these proofs are zero-knowledge and that the user blinds c and Z give us anonymity for the user. These proofs are described in detail in Section 5.3. Finally, the verifier checks that (1) c' is an encryption of $\log_{a_3} Z'$, and (2) the user carried out the proofs correctly. If so, the verifier accepts.

To convert an appointed-verifier membership certificate into a universally verifiable membership certificate, the appointed verifier reveals $\log_{a^3} Z'$ to the user. Under the strong RSA assumption and the hardness of discrete logarithms modulo n, the resulting tuple, (x, s, z, c, u, e) is hard to forge (cf. full version of this paper [10]).

Let us finally discuss the second key element to achieve the appointed verifier property: requiring a user to verifiably encrypt, under her own public key, some of the secrets she uses in the *Authenticate to appointed verifier* protocol. This is necessary as, in essence, the definition for this property requires that no matter how adversary $\mathcal{A}$ behaves, and no matter how often and when $\mathcal{A}$ and $\mathcal{B}$ exchange messages, there is nothing $\mathcal{A}$ can convince $\mathcal{B}$ of that $\mathcal{D}$ (in fake mode) would not be able to convince him of either. Running in fake mode requires $\mathcal{D}$ to

---

1 This step resolves the following paradox: On the one hand, we want the encryption scheme to be malleable, so that the user can successfully blind the ciphertext c. On the other hand, we want it to be secure against adaptive attacks by malicious users. Thus c is created by the group manager.

know a great deal about the internal information of $\mathcal{A}$. Traditionally, this would be realized by allowing $\mathcal{D}$ black-box access to $\mathcal{A}$ and the ability to rewind it. However, as we allow message exchanges between $\mathcal{A}$ and $\mathcal{B}$ at arbitrary times, arbitrarily interleaved with other executions, this is not possible as it would require $\mathcal{D}$ to have black-box access to other players as well (in particular those controlled by $\mathcal{B}$). Thus, $\mathcal{D}$ must somehow contain a knowledge extractor that does not rewind $\mathcal{A}$. $\mathcal{D}$ will instead extract what it needs to know from the verifiably encrypted secrets. Thus, we need the public-key model: in this model, $\mathcal{A}$ and, as a consequence, $\mathcal{D}$, will receive as input the secret keys of all the players controlled by $\mathcal{A}$.

4 Preliminaries

4.1 Proof Protocols and Corresponding Notation

We use notation introduced by Camenisch and Stadler [12] for the various proofs of knowledge of discrete logarithms and proofs of the validity of statements about discrete logarithms. For instance,

$$PK\{(\alpha,\beta,\gamma) : y = g^{\alpha}h^{\beta} \ \wedge \ \tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\gamma} \ \wedge \ (u \le \alpha \le v)\}$$

denotes a "*zero-knowledge* P*roof of* K*nowledge of integers* α*,* β*, and* γ *such that* $y = g^{\alpha}h^{\beta}$ *and* $\tilde{y} = \tilde{g}^{\alpha}\tilde{h}^{\gamma}$ *holds, where* $v < \alpha < u$*,*" where $y, g, h, \tilde{y}, \tilde{g}$, and $\tilde{h}$ are elements of some groups $G = \langle g \rangle = \langle h \rangle$ and $\tilde{G} = \langle \tilde{g} \rangle = \langle \tilde{h} \rangle$. By convention, the Greek letters denote quantities the knowledge of which is being proved, while all other parameters are known to the verifier. Using this notation, a proof-protocol can be described by just pointing out its aim while hiding all details.

It is important that we use protocols that are *concurrent* zero-knowledge. They are characterized by remaining zero-knowledge even if several instances of the same protocol are run arbitrarily interleaved [24, 25]. Damgård [24] shows that so-called Σ-protocols (this includes all the PK's discussed above) can easily be made concurrent zero-knowledge in many practical scenarios, including the public-key model. We assume throughout that the latter technique is used with all PK's.

4.2 Proving that a Commitment Contains a Paillier Encryption

Our scheme requires a proof that some value e is a Paillier encryption [34, 35] of a value x that the prover knows, under a given Paillier public key (g, n), and a similar proof where the ciphertext e is not given as input to the verifier; instead only a Pedersen commitment [36] to ciphertext e is given. Protocols for carrying out the former proof have been realized [21]. The latter proof is, to the best of our knowledge, not found in the literature and is constructed as follows:

Let (g, n) be the public key of Paillier's encryption scheme. Assume that we are given a group $\hat{G} = \langle \hat{g} \rangle = \langle \hat{h} \rangle$ of order n^2. Let E be the commitment to a ciphertext, i.e., $E = \hat{g}^{e}\hat{h}^{z}$ where $e = g^{x}r^{n} \mod n^2$. Using the protocol denoted

$PK\{(\alpha,\beta,\gamma) : E = \hat{g}^{g^\alpha \beta^n} \hat{h}^\gamma\}$ the prover can convince the verifier that E is a commitment to a Paillier encryption of some value she knows. The protocol is as follows.

1. The prover chooses $r_1 \in_R \mathbb{Z}_n$ and $r_2, r_3 \in_R \mathbb{Z}_{n^2}$, computes $t = \hat{g}^{g^{r_1} r_2^n} \hat{h}^{r_3}$ and sends t to the verifier.
2. The verifier chooses a $c \in_R \{0,1\}$ and sends c to the prover.
3. The prover computes $s = r_1 - cx \mod n$, $u = r_2/r^c \mod n^2$, and $v = r_3 - czg^s u^n \mod n^2$ and sends s and u to the verifier.
4. The verifier checks whether $t = \hat{g}^{g^s u^n} \hat{h}^v$ if $c = 0$ and whether $t = E^{g^s u^n} \hat{h}^v$ otherwise.

It is easy to see that the proof is correct and honest-verifier zero-knowledge proof of knowledge.

4.3 Verifiable Encryption

Verifiable encryption [1, 8], is a protocol between a prover and a verifier such that as a result of the protocol, on input public key E, and value v, the verifier obtains an encryption e of some value s under E such that $(w, y) \in \mathcal{R}$. For instance, $\mathcal{R}$ could be the relation $(w, g^w) \subset \mathbb{Z}_q \times G$. Generalizing the protocol of Asokan et al. [1], Camenisch and Damgård [8] provide a verifiable encryption scheme for a class of relations that, in particular, includes all discrete-logarithm relations that are of relevance in this paper. We denote verifiable encryption similarly as the PK's, e.g., $e := VE(\mathsf{ElGamal}, (u,v))\{\xi : y = g^\xi\}$ denotes the verifiable encryption protocol for the ElGamal scheme, whereby $\log_g y$ is encrypted in e under public key (u, v). Note that e is not a single encryption, but the verifier's entire transcript of the protocol and contains several encryptions, commitments and responses of the underlying PK.

5 An Identity Escrow Scheme with Appointed Verifiers

5.1 Key and System Setup

Our protocols are realized in the public-key model, thus the initial setup is the public-key infrastructure in which each user has a public key and has proved knowledge of the secret key to some entity, say the CA. Specifically, some group $\tilde{G} = \langle\tilde{g}\rangle = \langle\tilde{h}\rangle$ of prime order $\tilde{q}$, such that $\log_{\tilde{g}} \tilde{h}$ is unknown. Also, each user has a secret key an $x \in_R \mathbb{Z}_q$, and a corresponding public key $\tilde{S}_U = \tilde{h}^x$. The user has submitted this $\tilde{S}_U$ to the CA of this public-key infrastructure and and has executed $PK\{(\alpha) : \tilde{S}_U = \tilde{h}^\alpha\}$ with the CA. The CA sends the user a signature on $\tilde{S}_U$ and publishes $\tilde{S}_U$ and the user's name.

In addition, to get security in case the protocols are executed concurrently, we assume that all zero-knowledge proofs (PK) are carried out using the construction due to Damgård [24]. This requires to initially set up public keys for a trapdoor commitment scheme.

Other security-related system parameters are as follows: the length ℓ_n of the RSA modulus of the group manager, integer intervals $\Gamma =]-2^{\ell_\Gamma}, 2^{\ell_\Gamma}[$, $\Delta =]-2^{\ell_\Delta}, 2^{\ell_\Delta}[$, $\Lambda =]2^{\ell_\Lambda}, 2^{\ell_\Lambda+\ell_\Sigma}[$ such that $\tilde{q} < 2^{\ell_\Gamma}$, $\ell_\Delta = \epsilon(4\ell_n + 3)$ and $\ell_\Gamma = 2\ell_n$, where $\epsilon > 1$ is a security parameter, and $\ell_\Lambda > \ell_\Sigma + \ell_\Delta + 4$. Furthermore, let ℓ_v be the length of the RSA modulus of the verifier for Paillier's encryption scheme [35]. We require that $2\ell_v < \ell_\Gamma$ holds. There further are ℓ_z and ℓ_r with $\ell_z > \epsilon\ell_r + 1$ and $\ell_z + \epsilon\ell_r + 1 < \ell_v$. Define the integer intervals $\Omega =]2^{\ell_z} - 2^{\ell_r}, 2^{\ell_z} + 2^{\ell_r}$, $\Phi =]-2^{\epsilon\ell_r}, 2^{\epsilon\ell_r}[$, and $\Omega' =]2^{\ell_z} - 2^{\epsilon\ell_r+1}, 2^{\ell_z} + 2^{\epsilon\ell_r+1}[$ (ℓ_r must be large enough to make computing an ℓ_r-bit discrete logarithm modulo an ℓ_n-bit RSA modulus hard, where the modulus is the product of two safe primes.)

The public key of the group manager consists of an ℓ_n-bit RSA modulus $n = pq = (2p' + 1)(2q' + 1)$ that is the product of two safe primes, and random elements $a_4, a_3, a_2, a_1, a_0, g, h \in_R QR_n$ of maximal order. The factorization of n is the group manager's secret key. The revocation manager sets up his public and secret key for the Cramer-Shoup encryption scheme [22] over $\tilde{G}$ (i.e., the group that comes from the public-key infrastructure), i.e., $x_1, \dots, x_5 \in_R \mathbb{Z}_{\tilde{q}}$ are the secret keys and $(y_1 := \tilde{g}^{x_1}\tilde{h}^{x_2}, y_2 := \tilde{g}^{x_3}\tilde{h}^{x_4}, y_3 := \tilde{g}^{x_5})$ constitutes the public key. The revocation manager also publishes a collision-resistant hash function $\mathcal{H}$.

Each user also publishes an ℓ_n-bit RSA modulus n_U that is the product of two safe primes and two generators g_U and h_U of QR_n.

Each appointed verifier chooses a public key (n_v, g_v) of the Paillier encryption scheme, where n_v is an ℓ_v bit RSA modulus and $g_v = 1 + n_v \pmod{n_v^2}$. The verifier also publishes $\hat{G} = \langle\hat{g}\rangle = \langle\hat{h}\rangle$ of order n_v^2.

5.2 Joining with Appointed Verifier

In this protocol, aside from the public information, the user's input will be a secret key $x \in \Gamma$ and her identifier $\tilde{S}_U$ and her output will be a membership certificate tuple (s, Z, c, e, u) w.r.t. an appointed verifier V such that $s \in_R \Delta$, c is the encryption of $z = \log_{a_3} Z \bmod n$ under V's Paillier public key, $z \in \Omega$, $e \in \Lambda$ a prime, and $u^e = a_4^c Z a_2^x a_1^s a_0 \bmod n$. The group manager's input will be his secret key and all the public information in the system. His output is the user's identifier $\tilde{S}_U = \tilde{h}^x$ and also the values $S = a_1^s a_2^x$, z, c, e, u.

A secure two-party protocol that has this functionality is as follows:

1. User chooses a value $s_1 \in_R \Delta$. The integer s_1 will be the user's contribution to s. $r_x, r_s \in_R \{0,1\}^{2\ell_n}$ are also chosen. User sets $C_1 := g^{s_1}h^{r_s} \mod n$ and $C_2 := g^x h^{r_x} \mod n$, sends C_1, C_2, $\tilde{S}_U$, and the CA's signature on $\tilde{S}_U$ to the GM, and serves as the prover to verifier GM in

$$PK\{(\alpha,\beta,\gamma,\delta) : C_1^2 \equiv (g^2)^\alpha (h^2)^\beta \ \wedge\ C_2^2 \equiv (g^2)^\gamma (h^2)^\delta \ \wedge \\ \tilde{S}_U = \tilde{h}^\gamma \ \wedge\ \alpha \in \Delta \ \wedge\ \gamma \in \Gamma\} \ .$$

2. GM checks the CA's signature on $\tilde{S}_U$, chooses a random $s_2 \in_R \Delta$ and sends s_2 to U.

3. The user computes $s = (s_1 + s_2 \bmod (2^{\ell_\Delta+1} - 1)) - 2^{\ell_\Delta} + 1$, ($s$ is the sum of s_1 and s_2, adjusted appropriately so as to fall in the interval Δ) and $\tilde{s} = \left\lfloor \frac{s_1+s_2}{2^{\ell_\Delta+1}-1} \right\rfloor$ ($\tilde{s}$ is the value of the carry resulting from the computation of s above). The user then sets $S := a_1^s a_2^x$ and sends S to *GM*.
4. Now, the user must show that S was formed correctly. To that end, she chooses $r_{\tilde{s}} \in_R \{0,1\}^{\ell_n}$, sets $C_3 := g^{\tilde{s}} h^{r_{\tilde{s}}}$, sends C_3 to *GM*, and executes

$$\begin{aligned} PK\{(\alpha,\beta,\gamma,\delta,\varepsilon,\zeta,\vartheta,\xi) : C_1^2 = (g^2)^\alpha (h^2)^\beta \ \wedge \ C_3^2 = (g^2)^\varepsilon (h^2)^\zeta \ \wedge \\ S^2 = (a_1^2)^\vartheta (a_2^2)^\gamma \ \wedge \ (C_1^2 (g^2)^{(r-2^{\ell_\Delta}+1)})/(C_3^2)^{(2^{\ell_\Delta+1}+1)} = (g^2)^\vartheta (h^2)^\xi \ \wedge \\ \tilde{S}_U = \tilde{h}^\gamma \ \wedge \ \gamma \in \Gamma \ \wedge \ \vartheta \in \Delta\} \end{aligned}$$

as prover with the *GM*.
5. *GM* chooses $z \in_R \Omega$, a prime $e \in_R \Lambda$, computes $Z := a_3^z$ and $u := (a_4^c a_3^z S a_0)^{1/e} \pmod n$, encrypts z under the public key of the appointed verifier, i.e., chooses a random $r \in_R \mathbb{Z}_{n_v}$ and computes $c := g_v^z r^{n_v} \pmod{n_v^2}$. *GM* sends Z, u, e, and c to U.
6. User checks whether $u^e \equiv a_4^c Z a_2^x a_1^s a_0 \pmod n$, $e \in \Lambda$, and $c \in \mathbb{Z}_{n_v^2}$.
7. *GM* proves to the user that c indeed encrypts $\log_{a_3} Z$ and that this value lies in Ω. To this end *GM* chooses $\hat{r} \in_R \{0,1\}^{\ell_n}$, computes $\hat{Z} := g_U^z h_U^{\hat{r}}$, sends $\hat{Z}$ to U and carries out the protocol

$$\begin{aligned} PK\{(\alpha,\beta,\gamma,\delta) : \tilde{S}_U = \tilde{h}^\gamma \ \vee \ \big(c \equiv g_v^\alpha \beta^n \pmod{n_v^2} \ \wedge \\ Z^2 \equiv (a_3^2)^\alpha \pmod n \ \wedge \ \hat{Z}^2 \equiv (g_U^2)^\alpha (h_U^2)^\delta \pmod{n_U} \ \wedge \ \alpha \in \Omega\big)\} \end{aligned}$$

as the prover with the user.
8. *GM* stores $S, \tilde{S}_U, u, e, c, z$ and the user's name in its database.
9. *GM* and the user go home, listen to music, and have coffee, tea, and cake.

Remark: In step 7 the *GM* proves that it knows either the user's secret $x = \log_{\tilde{h}} \tilde{S}_U$ or that c is an encryption of $\log_{a_3} Z$ so as to leave no evidence to the user that the protocol took place.

5.3 Authenticate to an Appointed Verifier

This is a protocol between a user and an appointed verifier. The user's input is the public information, the membership certificate issued as described above, and a revocation condition *con* which specifies under which conditions the user's identity may be discovered. The verifier's input, aside from the public information, is his Paillier secret key. The verifier's output is *con*, and an encryption of the user's identifier under the revocation manager's public key with condition *con*. The verifier accepts if the user succeeds in proving knowledge of a valid membership certificate, and in proving that this membership certificate was issued to the user whose encrypted identifier is provided. The protocol is as follows:

1. The user and the verifier agree on a revocation condition *con*.

2. The user first blinds the ciphertext c, i.e., chooses random $\tilde{r}_1 \in_R \Phi$ and $\tilde{r}_2 \in_R \mathbb{Z}_{n_v}$ and computes $\tilde{c} := c g_v^{\tilde{r}_1} \tilde{r}_2^{n_v^2} \pmod{n_v}$ and $\tilde{Z} := Z a_3^{\tilde{r}_1} \pmod{n}$, then sends $\tilde{c}$ and $\tilde{Z}$ to the verifier.
3. The user computes a blinded public key for use with verifiable encryption, i.e., she chooses a random $w \in_R \mathbb{Z}_{\tilde{q}}$, computes $\tilde{u} := \tilde{h}^w$ and $\tilde{v} := \tilde{S}_U^w$ (hence $\tilde{v} = \tilde{u}^x$), and sends $\tilde{u}$, $\tilde{v}$ to the verifier.
4. The user chooses $r_1, r_2, r_3 \in_R \mathbb{Z}_{n^2}$ and $\hat{r} \in_R \mathbb{Z}_{n_v^2}$ and computes $T_1 = uh^{r_1} \bmod n$, $T_2 = g^{r_1} h^{r_2} \bmod n$, $T_3 = g^{\tilde{r}_1} h^{r_3} \bmod n$, and $\hat{T} = \hat{g}^{g_v^{\tilde{r}_1} \tilde{r}_2^{n_v}} \hat{h}^{\hat{r}}$. ($T_1$ serves as a blinded u, and T_2 is an additional commitment which will be used to prove that T_1 was formed correctly. $\hat{T}$ and T_3 are needed to show that the ciphertext c was blinded in the same way as Z.) Then the user computes the encryption E of his identifier under condition *con*, as follows: he chooses $r_4 \in_R \mathbb{Z}_q$ and sets $E := (E_1, E_2, E_3, E_4)$, where $E_1 := \tilde{g}^{r_4}$, $E_2 := \tilde{h}^{r_4}$, $E_3 := \tilde{h}^x y_3^{r_4}$, $E_4 := y_1^{r_4} y_2^{r_4 \mathcal{H}(E_1 \| E_2 \| E_3 \| con)}$. The user sends $(T_1, T_2, T_3, \hat{T}, E)$ to the verifier.
5. The user serves as prover to the verifier in

$$VE(\mathsf{ElGamal}, (\tilde{u}, \tilde{v}))\big\{(\varrho, \vartheta, \varsigma) : \ \hat{T} = \hat{g}^{g_v{}^{\vartheta} \varrho^n} \hat{h}^{\mu} \ \wedge \ T_3^2 = (g^2)^{\vartheta} (h^2)^{\varsigma} \ \wedge \ \vartheta \in \Phi\big\}$$

and in

$$\begin{aligned} PK\big\{(\alpha, \beta, \gamma, \delta, \zeta, \varepsilon, \varphi, \xi, \nu, \mu, \psi, \vartheta, \varsigma) : \ 1 = (T_2^2)^{\alpha} \big(\frac{1}{g^2}\big)^{\varepsilon} \big(\frac{1}{h^2}\big)^{\psi} \ \wedge \\ a_0^2 \tilde{Z}^2 = (T_1^2)^{\alpha} \big(\frac{1}{a_1^2}\big)^{\beta} \big(\frac{1}{a_2^2}\big)^{\nu} (a_3^2)^{\vartheta} \big(\frac{1}{a_4^2}\big)^{\varphi} \big(\frac{1}{h^2}\big)^{\varepsilon} \ \wedge \ T_2^2 = (g^2)^{\delta} (h^2)^{\zeta} \ \wedge \\ T_3^2 = (g^2)^{\vartheta} (h^2)^{\varsigma} \ \wedge \ \hat{g}^{\tilde{c}} = \hat{T}^{\varphi} \hat{h}^{\kappa} \ \wedge \ \tilde{v} = \tilde{u}^{\nu} \ \wedge \ \tilde{u} = \tilde{h}^{\gamma} \ \wedge \\ E_1 = \tilde{g}^{\xi} \ \wedge \ E_2 = \tilde{h}^{\xi} \ \wedge \ E_3 = \tilde{h}^{\nu} y_3^{\xi} \ \wedge \ E_4 = (y_1 y_2^{\mathcal{H}(E_1 \| E_2 \| E_3 \| con)})^{\xi} \wedge \\ \alpha \in \Lambda \wedge \beta \in \Delta \wedge \nu \in \Gamma \wedge \vartheta \in \Phi \wedge \varphi \in [1, n_v^2 - 1]\big\} \ . \end{aligned}$$

6. The verifier decrypts $\tilde{c}$ to get $\tilde{z}$ and checks whether $\tilde{Z} = a_3^{\tilde{z}} \pmod{n}$ and whether $\tilde{z} \in \Omega'$.

Let us consider the efficiency of the above verifiable encryption protocol. Recall that verifiable encryption works by repeating the underlying PK sufficiently many times, e.g., $k = 80$ times. Assuming that exponentiation with a $2\ell_n$-bit modulus corresponds to about 8 exponentiations with an ℓ_n-bit modulus, the total computational load of both the prover and the verifier for the verifiable encryption protocol amounts to $17k$ exponentiations with an ℓ_n-bit modulus and about 42 exponentiations with an ℓ_n-bit modulus for the PK. On the verifier's side, this load can be considerably reduced by applying so-called batch verification [4].

5.4 Convert and Authenticate

This paragraph briefly discusses how an appointed verifier can convert an appointed-verifier membership certificate into an ordinary membership certificate and how a group member can then convince anyone of her group membership.

To convert a certificate, the user and the verifier first carry out the *authenticate with appointed verifier* operation. If this operation is successful, the verifier can provide the user with the decryption of $\tilde{c}$. This will allow the user to compute the value z encrypted as c. Thus she holds values (x, s, z, c, u, e) such that $u^{2e} = (a_4^c a_3^z a_2^x a_1^s a_0)^2 \bmod n$, i.e., a valid group membership certificate. Proving possession of this certificate, i.e., authenticating as a group member to any verifier, can now be done similarly to the way it is done for an appointed verifier above. The only difference is that there is no encryption $\tilde{c}$ and no commitments T_3 and $\hat{T}$, and hence the corresponding parts in the proof-protocol are dropped: First, steps 2 and 5 are no longer needed; second, in step 4 the verifiable encryption protocol is not needed and in the PK the first term of the expression proved is replaced by $a_0^2 = (T_1^2)^\alpha (\frac{1}{a_1^2})^\beta (\frac{1}{a_2^2})^\nu (\frac{1}{a_3^2})^\vartheta (\frac{1}{a_4^2})^\varphi (\frac{1}{h^2})^\varepsilon$ while the terms $T_3^2 = (g^2)^\vartheta (h^2)^\varsigma$, $\hat{g}^{\tilde{c}} = \hat{T}^\varphi \hat{h}^\kappa$, $\tilde{v} = \tilde{u}^\nu$, and $\tilde{u} = \tilde{h}^\gamma$ are dropped. The fact that the verifiable encryption protocol is no longer needed makes the whole protocol much more efficient as it was the bulk of the computational load.

5.5 Anonymity Revocation

Upon a request $E = (E_1, E_2, E_3, E_4)$ and *con*, the revocation manager checks whether $E_4 = E_1^{x_1 + x_3 \mathcal{H}(E_1 \| E_2 \| E_3 \| con)} E_2^{x_2 + x_4 \mathcal{H}(E_1 \| E_2 \| E_3 \| con)}$ and whether the revocation condition *con* is fulfilled. If these checks succeed, he returns $\hat{S} := E_3 / E_1^{x_5}$. If E was produced in an *Authenticate to an Appointed Verifier* or an *Authenticate* protocol, $\hat{S}$ will match the identifier $\tilde{S}_U$ of the user who took part in the protocol.

5.6 Proof of Security and Appointed Verifier Property

We outline how security is proven and state the important theorems and lemmas. For details and all the proofs we refer to the full version of this paper [10].

Protecting the Honest Players. Security for the honest players is proven by providing a simulator that satisfies Definition 1. The simulator will create cryptographic instantiations for the honest parties. For every transaction between the adversary and an honest party, the simulator will execute its cryptographic part on behalf of these honest parties. If the cryptographic implementation of a protocol prescribes that a real-world honest player should behave in a way that is different from the underlying ideal-world player, then the simulator rejects. (This can happen if an adversary succeeds in proving group membership in such a way that the simulator is unable to extract a secret key to which a membership certificate was issued in a previous transaction. As a result, an ideal trusted party would tell the ideal verifier to reject the adversary's user, while the cryptographic implementation would dictate the real-world verifier to accept.)

This simulator is constructed [10] in the usual way, with the following subtle difference: in the *Authenticate* protocol, when an honest user interacts with a dishonest verifier, the simulator does not get to know which user it is and hence does not know which user to simulate towards the the verifier. There are two cases to consider here, one where the revocation manager is honest and one where he is not. For brevity we will address only the former case here: The simulator forms a ciphertext E that is an encryption of 0 the revocation manager's public key. He then creates a random public key $P = (\tilde{u}, \tilde{v})$ for the verifiable encryption and chooses $\tilde{r}_1 \in_R \Phi$, $\tilde{r}_2 \in_R \mathbb{Z}_{n_v}$, and T_1,T_2 and $\hat{T}$ at random from their corresponding domains. Then, the simulator sends $(Z', c', T_1, T_2, \hat{T}, E, P)$ to the adversary and carries out the verifiable encryption protocol:

$$VE(\mathsf{ElGamal}, (\tilde{u}, \tilde{v}))\{(\varrho, \vartheta, \varsigma) : \; \hat{T} = \hat{g}^{{g_v}^{\vartheta} \varrho^{n}} \hat{h}^{\mu} \; \wedge \; T_3^2 = (g^2)^{\vartheta} (h^2)^{\varsigma} \; \wedge \; \vartheta \in \Phi\}$$

with the adversary and finally runs the simulator for the view of the verifier in the group membership proof protocol described in Section 5.3.

The following lemma follows from the semantic security of the verifiable encryption scheme, as well as from adaptive chosen-ciphertext security of the encryption scheme under which the users' identifiers are encrypted [10].

Lemma 1. *Either the simulator produces a computationally indistinguishable view, or it rejects. The computational indistinguishability is under the decisional Diffie-Hellman assumption for the group over which the Cramer-Shoup encryption of the identifiers is done.*

The only thing left to prove security is to show that the simulator almost never rejects. We observe that the only case when the simulator rejects is when the adversary demonstrates group membership for an unauthorized user-verifier pair. We show [10] that if this simulator rejects non-negligibly often, then either there exists a polynomial-time algorithm for forging membership certificates (thus violating the strong RSA assumption or the discrete logarithm assumption), or there exists a polynomial-time algorithm for cracking the Paillier cryptosystem, or there exists a way to circumvent the knowledge extractor for one of the proofs of knowledge:

Lemma 2. *Under the strong RSA assumption, the hardness of discrete logarithms modulo a safe prime product, and the security of Paillier cryptosystem, the simulator rejects with only negligible probability.*

Putting everything together, we get:

Theorem 1. *Under standard number-theoretic assumptions, the construction presented in Section 5 is an identity escrow scheme with security guarantee for honest users, as required by Definition 1.*

Appointed Verifier Property. Given the public key $(n, a_0, a_1, a_2, a_3, a_4, g, h)$ of the group manager, and public key $(\tilde{n}, \tilde{g})$ of the appointed verifier V, for any given S, it is easy to create a tuple (Z, c, u, e) such that no one except V can distinguish it from a valid membership certificate. Create such a tuple as follows (call this procedure the *forger*): choose any $r \in_R \mathbb{Z}_{\tilde{n}^2}$, set $c := r^{\tilde{n}} \bmod \tilde{n}^2$ (c is simply the encryption of 0 under the verifier's public key), $u \in_R QR_n$, $e \in_R \Lambda$, and set $Z := u^e / a_4^c S a_0$.

Lemma 3. *Under the assumption that the Paillier cryptosystem is semantically secure, for all $x \in \Gamma$, the tuple (s, Z, c, e, u) such that $s \in_R \Delta$, and (Z, c, e, u) are created by the forger above on input $S = a_1^s a_2^x$, is indistinguishable from a valid membership certificate created by querying oracle O that, on input S, carries out step 5 of the* Join with appointed verifier *protocol.*

Proof. Let D_1 be the distribution of fake certificates as above, and D_2 be the distribution of valid certificates. Suppose that a distinguisher existed. Then we break the security of the Paillier cryptosystem as follows: we give the reduction access to the secret keys of the group manager. The reduction chooses a random $z \in \Omega$ and asks the encryption oracle to give it an encryption of either 0 or z. It is easy to see that if the oracle returns an encryption of 0, then the resulting tuple will be distributed according to D_1, while if the oracle returns an encryption of z, then the resulting tuple will be distributed according to D_2. Thus we can use the distinguisher for D_1 and D_2 to break the semantic security of the Paillier cryptosystem. □

Based on this way of forging a single membership certificate, we can now build a deceiver $\mathcal{D}$. In *fake* mode, on input a list L, $\mathcal{D}$ does not forward the messages pertaining to *Join with appointed verifier* for user A and verifier V if $(A, V) \in L$. Instead, he impersonates the group manager GM to $\mathcal{A}$. $\mathcal{D}$ proceeds as follows: it conducts steps 1 through 4 of the *Join with appointed verifier* protocol exactly the same way as GM would to get an input S. Then it creates a fake certificate (Z, c, e, u) using the forger described above. As the secret key $x = \log_{\tilde{h}} \tilde{S}_U$ of user A was given to $\mathcal{D}$ as input, $\mathcal{D}$ succeeds in carrying out the PK in step 7. It then stores this certificate.

For $(A, V) \notin L$, $\mathcal{D}$ forwards all the messages, and, in case of a successfully carried out *Join*, stores the certificate.

When $\mathcal{A}$ engages in *sid* that is an *Authenticate to appointed verifier* with some verifier V, $\mathcal{D}$ proceeds as follows (recall that verifiable encryption is by itself a three-move proof of knowledge): it first receives, from $\mathcal{A}$, all messages up to step 5 and buffers them. Then, it receives the first message of the *VE* protocol, and in particular the ciphertext $\tilde{c}$ and the value $\tilde{Z}$. By the properties of *VE*, this first message contains an ElGamal encryption under $(\tilde{v}, \tilde{u})$ of values $\tilde{r}_1$ and $\tilde{r}_2$. It checks whether $\tilde{v} = \tilde{u}^x$ for some secret key x of a player $\mathcal{A}$ controls. If this is not the case, then it knows that the verifier will reject anyway– so it forwards the message to V. If it finds the right x, then it decrypts the first message of the verifiable encryption and obtains $\tilde{r}_1$ and $\tilde{r}_2$. If the first message of the verifiable encryption is invalid, it detects that and then it knows that V will reject, so it forwards $\mathcal{A}$'s message to V. It then sets $c := \tilde{c}/(g_v^{\tilde{r}_1} \tilde{r}_1^{n_v})$. It then looks up a membership certificate that contains the ciphertext c. If it fails to find one, it knows that the verifier will reject – so it forwards the message to V.

If it finds one, and it is a valid membership certificate, then it forwards all the messages between $\mathcal{A}$ and V for this *sid*.

If it is a fake membership certificate that includes ciphertext c, it checks whether this certificate also includes the value $Z := \tilde{Z}/(a_3^{\tilde{r}_1})$. If it does not, then $\mathcal{D}$ knows that the verifier will reject anyway – so it forwards the message to V.

Otherwise, this first message of $\mathcal{A}$ is valid. Since $\mathcal{D}$ has the valid membership certificate for (A, V), $\mathcal{D}$ tells $\mathcal{F}$ to send the first valid message of an *Authenticate to appointed verifier* to V. Then $\mathcal{D}$ simulates V for $\mathcal{A}$: it creates a challenge message and sends it to $\mathcal{A}$. If $\mathcal{A}$ responds to the message so as to correctly complete the corresponding proof of knowledge and verifiable encryption, then $\mathcal{D}$ tells $\mathcal{F}$ to send V a message that corresponds to a valid response to V's challenge. Otherwise, $\mathcal{D}$ tells $\mathcal{F}$ to send to V a message that does not constitute a valid response. After that, V either responds to $\mathcal{F}$ with an accept or reject. $\mathcal{F}$ forwards that response to $\mathcal{D}$, which in turn sends it to $\mathcal{A}$.

It is easy to see that the following lemma holds [10]:

Lemma 4. *Under the assumption that the Paillier cryptosystem is semantically secure, the strong RSA assumption, and the assumption that computing discrete logarithms modulo a safe prime product is hard, the following holds: Provided that V never performs the* Convert *and* Identify *operation for A, if the probability that $\mathcal{B}$ accepts when talking to $\mathcal{D}$ in* real *mode differs non-negligibly from the probability that $\mathcal{B}$ accepts when talking to $\mathcal{D}$ in* fake *mode, then: $\mathcal{B}$ and verifier V have engaged in the* Authenticate with appointed verifier *protocol in which V accepted such that a subsequent* Identify *operation, if carried out, will point to A.*

Using Lemma 4, the following is immediate by Definition 2:

Theorem 2. *Under standard number-theoretic assumptions, the construction presented in Section 5 is an identity escrow scheme with the appointed verifier property, as required by Definition 2.*

6 Concluding Remarks

We note that in order to implement several identity escrow schemes at the same time using our methods, the set-up, apart from the public-key infrastructure, has to be repeated for each instance. In particular, the public keys of the verifiers will have to be different for each instance. It is an interesting question whether it would be possible to avoid this and yet have a practical construction that is secure against adaptive attacks. It is also interesting whether the public-key model can be eliminated from the picture.

An appointed-verifier identity escrow scheme is only the first step towards a bigger goal of realizing protocols in which it is provably hard to convince an unauthorized party of the truth of some statement. It would be interesting to apply our methods in the context of electronic voting and consider existing voting schemes and how close they come to satisfying an appropriate modification of our definition, and, if a gap appears, whether the techniques developed in this paper could resolve it.

Acknowledgments

The second author acknowledges the support of an NSF graduate fellowship and of the Lucent Technologies GRPW program.

References

1. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):591–610, 2000.
2. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *CRYPTO 2000*, vol. 1880 of *LNCS*, pp. 255–270. Springer Verlag, 2000.
3. N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *EUROCRYPT '97*, vol. 1233 of *LNCS*, pp. 480–494.
4. M. Bellare, J. A. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 236–250. Springer Verlag, 1998.
5. J. C. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *Proc. 26th STOC*, pp. 544–553. ACM, 1994.
6. S. Brands. Untraceable off-line cash in wallets with observers. In *CRYPTO '93*, vol. 773 of *LNCS*, pp. 302–318, 1993.
7. J. Camenisch. Efficient anonymous fingerprinting with group signatures. In *ASIACRYPT 2000*, vol. 1976 of *LNCS*, pp. 415–428. Springer Verlag, 2000.
8. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In *ASIACRYPT 2000*, vol. 1976 of *LNCS*, pp. 331–345, 2000.
9. J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multi-show credential system with optional anonymity revocation. In *EUROCRYPT 2001*, vol. 2045 of *LNCS*, pp. 93–118. Springer Verlag, 2001.
10. J. Camenisch and A. Lysyanskaya. An identity escrow scheme with appointed verifiers. http://eprint.iacr.org/2001, 2001.
11. J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *ASIACRYPT '98*, vol. 1514 of *LNCS*, pp. 160–174. Springer Verlag, 1998.
12. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO '97*, vol. 1296 of *LNCS*, pp. 410–424. Springer Verlag, 1997.
13. R. Canetti. *Studies in Secure Multiparty Computation and Applications.* PhD thesis, Weizmann Institute of Science, Rehovot 76100, Israel, June 1995.
14. R. Canetti. Security and composition of multi-party cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
15. D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.
16. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, Oct. 1985.
17. D. Chaum. Designated confirmer signatures. In *EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 86–91. Springer Verlag Berlin, 1994.

18. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO '88*, vol. 403 of *LNCS*, pp. 319–327. Springer Verlag, 1990.
19. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT '91*, vol. 547 of *LNCS*, pp. 257–265. Springer-Verlag, 1991.
20. L. Chen and T. P. Pedersen. New group signature schemes. In *EUROCRYPT '94*, vol. 950 of *LNCS*, pp. 171–181. Springer-Verlag, 1995.
21. R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. Manuscript. Available from `http://eprint.iacr.org`.
22. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 13–25, Berlin, 1998. Springer Verlag.
23. R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *Proc. 6th ACM CCS*, pp. 46–52. ACM press, nov 1999.
24. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*, pp. 431–444. Springer Verlag, 2000.
25. C. Dwork and A. Sahai. Concurrrent zero-knowledge: Reducing the need for timing constraints. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 105–120, 1998.
26. A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In *CRYPTO '86*, vol. 263 of *LNCS*, pp. 186–194, 1987.
27. E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO '97*, vol. 1294 of *LNCS*, pp. 16–30, 1997.
28. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 123–139, 1999.
29. D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2):84–88, 1999.
30. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT 2000*, vol. 1807 of *LNCS*, pp. 539–556, 2000.
31. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *EUROCRYPT '96*, vol. 1233 of *LNCS*, 1996.
32. J. Kilian and E. Petrank. Identity escrow. In *CRYPTO '98*, vol. 1642 of *LNCS*, pp. 169–185, Berlin, 1998. Springer Verlag.
33. A. Lysyanskaya and Z. Ramzan. Group blind digital signatures: A scalable solution to electronic cash. In *Proc. Financial Cryptography*, 1998.
34. T. Okamoto and S. Uchiyama A new public-key cryptosystem as secure as factoring. In *EUROCRYPT '98*, vol. 1403 of *LNCS*, pp. 308–318, 1998.
35. P. Paillier. Public-key cryptosystems based on composite residuosity classes. In *EUROCRYPT '99*, vol. 1592 of *LNCS*, pp. 223–239. Springer Verlag, 1999.
36. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91*, vol. 576 of *LNCS*, pp. 129–140. Springer Verlag, 1992.
37. B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *Proc. 7th ACM CCS*, pp. 245–254. ACM press, nov 2000.