Identity Escrow

Joe Kilian [*] Erez Petrank [†]

Abstract

We introduce the concept of *escrowed identity*, an application of key-escrow ideas to the problem of authentication. In escrowed identity, one party A does *not* give his identity to another party B, but rather gives him information that would allow an authorized third party E to determine A's identity. However, B receives a guarantee that E can indeed determine A's identity. We consider a number of possible features of escrowed identity schemes, and describe a variety of implementations that achieve various subsets of these features. In particular, we observe that group signature schemes can be used to escrow identities, achieving most (though not all) of the desired features.

The most interesting feature we consider is *separability*. The escrow agency is not involved in the day to day operation of the identification system, but is only called in when anonymity must be revoked. In the extreme case, there exist identity escrow schemes in which an arbitrary party (possessing a public key) can be designated an escrow agent without any knowledge or participation on their part until they are asked to revoke someone's anonymity.

1 Introduction

We consider a client that obtains regular or continual access to a service or facility. Examples include driving a toll highway with a regular commuter pass, parking regularly in a garage, entering one's club premises, or using internet services. In order to get the service, the client must convince the gate keeper that he is entitled to the service. The client can do this by identifying himself at the entrance. But must he really identify himself? Such

[*]NEC Research Institute, 4 Independence Way, Princeton, NJ 08540, USA. E-mail: `joe@research.nj.nec.com`

[†]IBM Haifa Research Lab, MATAM, Haifa 31905, Israel. Email: `erezp@haifa.vnet.ibm.com`. This work was done while the author was a post-doctoral fellow at DIMACS.

identification raises critical issues of privacy. For example, a more pervasive highway authority might as a side effect allow the tracing of people's movements to an unprecedented degree. On the other hand, what if the client remains completely anonymous, say, by an access code that is secret, but common to all clients? In certain rare circumstances, the service provider (and society at large) may have a compelling reason to know the identity of the client. For example, consider an automated access system for a parking garage. The garage cares that the person entering it is authorized to do so; the person's precise identity is normally not a valid concern. However, suppose that on some night a person was murdered in the garage. At this point, the garage owner and society at large may have a legitimate interest in knowing who was there on that night. Or, one might wish to have a computer "chat room" in which one has conditional anonymity: As long as one follows the rules laws, ones identity is secure from even the system administrator. But if one flagrantly breaks the rules (such as arranging drug deals in the "Lion King" kiddie chat room), suitable law enforcement agencies can be appealed to in order to determine one's identity. The reader may envision other examples such as a drunk driver causing a fatal accident on the highway, etc.

Traditionally, access control has been all or nothing. One obtains all the information about the other person up front, with no recourse to learn more if circumstances warrant. This rigidity generally leads one to allow less privacy, since one is likely to want as much information as one can get just in case a "bad case" arises. We give a more flexible alternative.

We consider a more flexible, two-tier approach to authentication. On the first tier, a person gives only as much information about themselves as is strictly necessary for ordinary circumstances. On the second tier, a person gives a more precise statement of their identity that may be needed in extraordinary circumstances. This second tier is only accessible with the help of a third party, which is separate from (and not under the control of) the party managing access. We describe identity escrow schemes in Section 1.1 below.

Key escrow has proven an active and contentious field of research and discussion (c.f. [24, 25, 22, 18, 21, 23, 27]). Most of the attention in this area has been restricted to the simple case of communication: party A sends an encrypted message $E_K(M)$ to party B; some centralized authority is given the capability to recover either K or the specific message M. As discussed in Section 3, group signatures has an escrow-like feature in which the anonymity of a signer may be revoked. We add yet a new domain for the application of key escrow ideas: authentication and identity schemes.

Some distinctive features of this application are that

- Escrowed identity may actually enhance privacy. By default, many identity schemes require a person to give their entire identity "up front." A protocol in which this information is only released under special circumstances may prove an acceptable, and more private substitute.

- Escrowed identity schemes work to the advantage of at least one of the parties invoking them. In traditional key escrow systems, both party only lose by following the escrow system, and have everything to gain by bypassing it (which is generally quite easy to do).

1.1 Escrowed Identity

An escrowed identity system consists of the following parties:

Identifier: The identifier is the client who identifies himself to the verifier (the gate-keeper).

Issuer: The issuer issues certificates to the identifier that allow him to identify himself in an escrowed manner.

Verifier: The verifier is typically the access provider who verifies the first-tier identity process as well as the escrow proof for the second-tier identity.

Escrow Agent(s): The escrow agent(s) use information forwarded by the verifier to make a second-tier identification of the identifier.

These parties execute the following protocols:

Initializing the system: The certificate issuer, and in some cases the escrow agent, computes whatever private information and publishes whatever public information is necessary to initialize the system.

Issuing a certificate: The certificate issuer gives a certificate to the identifier.

Checking the weak identity: The identifier convinces the verifier that he has a certificate, gives an *escrowed certificate* and convinces the verifier that the escrowed certificate is valid.

Recovering the complete identity: The verifier gives the escrow agent(s) the escrowed certificate, and the escrow agent(s) recovers the identity of the identifier.

There are a number of desirable features of escrowed identity systems. Ideally, one would like an efficient protocol that achieves all of these features. However, as we discuss below, there is a tradeoff between feature coverage and efficiency among the known escrowed identity protocols. Hence, we will describe for each implementation the features it does or does not achieve.

The following features are most pertinent to the notion of escrowed identity:

Valid first-tier identification: If a user receives a legitimate certificate from the issuer, and if he follows his protocol, then he will succeed in convincing a verifier that he is a legitimate user (i.e. has a certificate) with probability 1. Conversely, a computationally bounded user that has not been issued a certificate by the issuer will fail to convince the verifier of having a certificate with probability almost 1. (This implies that producing a certificate without the issuer's private key is computationally hard.)

Secure second-tier identity: After the verifier has seen one or more first-tier identification proofs, he cannot fake a legitimate identity in the sense described above.

Guaranteed escrow for second-tier identity: A computationally bounded user can interactively prove with high confidence that he has escrowed his second-tier identity; the escrow agency can determine this identity from the transcript of this proof. (This imply that producing a second certificate from a given legitimate on is computationally hard). One may further demand that even if many identifiers collaborate, the escrow agency may still recover the identity of one of them with high probability.

Resistance to impersonation: The escrow agency, even after recovering many identities of many users from the transcripts of weak identity proofs, cannot fake any legitimate identity in the sense described above. In particular, this implies that the escrowed identity does not reveal the certificate of the user. Similarly, one can require that the certificate issuer cannot fake the identity of someone already in the system.

Separability: The escrow agency is completely independent of the other parties unless a request to uncover the second-tier identity is made. An issuer, verifier and user can set up an identification system without ever registering with or communicating in any way with the escrow agent. The escrow agency is only "woken up" when there is a request to revoke anonymity.

Containing the escrow agent

Introducing an escrow agent into a system almost inevitably reduces its security. However, the nature of the problem and the separability property allows us to minimize the damage.

By separating the escrow agency from the initialization and normal operation of the identification system, we can have the agency be dormant most of the time. For example, if the escrow agency is implemented with secure hardware, this hardware can be stored in a secure bank vault until needed. This helps to reduce the chance that an escrow agent will be compromised.

A further check on a rogue escrow agent is that the verifier has to ask for a more precise identification. Key escrow for communication is typically coercive, and requires that someone be able to obtain the ciphertext of any two people's communications without their request or consent. Thus, the possibilities for widespread abuse are greater than with our scenario. Nevertheless, it is only prudent to allow for multiple escrow agents; the escrow agents in most of our protocols can be made to work using simple group cryptography (e.g. [15, 30]).

1.2 History and related work

An earlier version of this work appeared in [20], using cut-and-choose techniques for the zero-knowledge proofs. The current version describes much more efficient implementations based on *group signature schemes.* Group signature schemes were introduced by Chaum and Heyst [12], and subsequently developed in [13, 10, 28, 11]. Independently and concurrently with [20], Camenisch and Stadler [11] developed new schemes for efficient group signatures, one of which can quite efficiently achieve most of our goals; we describe this solution in Section 3.

At the heart of escrowed identity and group signatures is an efficient proof that an encrypted value possesses some property. Frankel, Tsiounis and Yung [17] and Young and Yung [33] give very efficient protocols of this type. Again, these proofs are are not completely applicable to our setting, but suggest that dramatically more efficient identity escrow schemes may be possible.

The notion of keeping a trusted agency dormant except for "emergencies" has been proposed in a number of contexts. Asokan, Shoup and Waidner [1] show how to use a dormant third party for a variety of applications related to the exchange of digital signatures. Young and Yung [33] show how to use a dormant escrow agent for key escrow. Brickell, Gemmel and Kravitz [5] and Stadler, Piveteau and Camenisch [32] show how to use a dormant escrow agent in electronic cash systems (more efficient schemes are presented in [29, 17]). Quite recently, Micali [26] has shown how to use a dormant agent for certified mail.

1.3 Road map

In Section 2 we describe some of the building blocks we use for our protocols. In Section 3 we discuss how to implement identity escrow using group signature schemes. In Section 4 we show how to achieve stronger separability . In Section 5 we show an implementation of an escrowed identity scheme based on the El-Gamal encryption and signature schemes.

2 Preliminaries

We describe some of the basic building blocks we use in our protocol.

2.1 Bit Commitments

We work in the argument framework of Brassard, Chaum and Crépeau [7]. In this paradigm, all parties are assumed to be computationally bounded. It is shown in [7] how to commit to bits in statistical zero-knowledge, based on the intractability of certain number-theoretic problems. Dåmgard, Pedersen and Pfitzmann [14] give a protocol for efficiently committing to and revealing strings of bits in statistical zero-knowledge, relying only on the existence of collision-intractable hash functions. This scheme is quite practical. For simplicity, we will simply speak of committing to and revealing bits when referring to the protocols of [14].

In some implementations we also commit to strings by probabilistic encryption [19] using the public key of the escrow agency. These commitments are only computationally secure. Furthermore, they allow for the escrow agents to recover the values of these commitments in addition to those revealed by the identifier in the course of the zero-knowledge proofs.

2.2 The El-Gamal signature and encryption schemes

We base one implementation of escrowed identity on the El-Gamal signature and encryption schemes [16], which we summarize, following [31], with slight modifications to suit our purposes.

In both schemes, there is a common prime p, which for our purposes is of the form $2q + 1$ where q is a prime. Let $g \in Z_p^*$ have order q. For the encryption scheme each party has a private key $X \in Z_q$ and a public key $Y = g^X$. For the signature scheme we denote the secret key by $S \in Z_q$ and the public key by $P = q^S$.

To encrypt a message $M \in Z_p$ given public key P, the sender uniformly generates $r \in Z_q$ and computes $E_Y(M, r) = (g^r, MY^r)$. The decryption function is given by $D_X(A, B) = B/A^X$.

The signer signs a message $M \in Z_{p-1}$ as follows.

1. The signer uniformly generates $r \in Z_q$, computes $a = g^r$, and casting it as an integer in $0..(p-1)$. This step is repeated until a and $p-1$ are relatively prime.

2. Using the extended Euclidean algorithm, the signer computes $b \in Z_{p-1}$ such that $Sa + rb = 1 \bmod p - 1$.

3. The signer returns (a, b).

To verify a signature (a, b) for M, the verifier checks that $P^a a^b = g^M \bmod p$.

2.2.1 Signing the "0" document is not secure

We remark on a weakness in the El-Gamal signature scheme. The document "0" can be signed efficiently by a party that does not have the secret key S. For example, by setting $a = P$ and $b = -P \bmod q$ we have $P^a a^b = P^P P^{-P} = q^0$. More generally, we can set $a = P^k \bmod p$ and setting $b = -a/k \bmod q$. For this reason, we use El-Gamal signatures for 1 instead of 0. We assume that given a number of signatures for 1 it is impossible to generate a different signature for 1. This assumption is plausible, but we do not know of any more standard assumptions that imply it.

2.3 The RSA encryption scheme

In the RSA encryption scheme the public key consists of $n = pq$ where p and q are prime and an exponent e, where e is relatively prime to n and $\phi(n)$. A message M is encrypted by computing $M^e \mod n$. The private key consists of d such that $de = 1 \mod (p-1)(q-1)$ (strictly, $de = 1 \mod \lambda(n)$ suffices), and M^e is decrypted by computing $(M^e)^d = M \mod n$.

We make an additional assumption beyond the security of RSA. We assume that for a random δ it is hard to find (a,b) such that $a^e - b^e = \delta \mod n$. Furthermore, we assume that given a set of such pairs $\{(a_i, b_i)\}$ it is hard to generate a new pair. Given d, it is easy to generate a pair (a,b) with given value of a^e by computing $a = (a^e)^d$ and $b = (a^e - \delta)^d$.

Camenisch and Stadler [11] use essentially the same assumption, and have pointed out that the system is not secure for very small e (2 or 3); the pairs (a,b) fall on a low degree curve, which can be used as a basis for an attack. However, a large e doesn't seem vulnerable to such an attack.

3 Using group signatures to escrow identity

Borrowing freely from the exposition in [11], we describe the basics of group signatures. We then proceed to describe how a particular implementation can be used to give an escrowed identity system with most of the desired features.

Group signatures

A group signature system consists of a *group manager* that oversees a group of signers. The group manager can allow other players to join the group. Any member of the group can sign a message on behalf of the group. The group manager can determine precisely who within the group signed the message, but no one else can.

This framework suggests the following set of protocols for something close to escrowed identification. Being issued a certificate corresponds to joining the signature group. The identifier can identify himself to the verifier by signing a (random) message of the verifier's choosing. To avoid replay and timing attacks, this message should include the (approximate) time and the verifier's name. To revoke anonymity the group manager determines who actually signed the message.

However, in the reduction outlined above, the group manager plays two roles: the issuer and the escrow agency. To obtain an escrowed identity system, we need to split these roles. As noted in [11], one of their implementations ([11], Section 6) allows for a considerable, though not complete separation between these roles. For completeness, we explicitly describe how this separation is made.

3.1 The Camenisch-Stadler construction

We briefly describe the parts of the Camenisch-Stadler construction that are relevant to separating the roles of the issuer and the escrow agency. We omit quantities, guarantees, protocols and other issues that are not directly relevant to this goal.

The group manager generates an RSA modulus $n = pq$, RSA exponents e_1, e_2, a cyclic group $G = \langle g \rangle$, of order n, an element $h \in G$, an El Gamal private-key, public key pair, $(\rho, y_R = h^\rho)$. It publishes $(n, e_1, e_2, G, g, h, y_R)$.

The identifier randomly generates a private x and computes $y = x^{e_1}$ and $z = g^y$. As part of the registration process, the identifier sends z to the group manager. The only operation the group manager performs relying on its private information is the computation of $\tilde{y}^{1/e_2}$ for some $\tilde{y}$ (used as part of a blind decryption to generate the certificate).

The signature incorporates a proof of knowledge of a valid certificate and the private key, x. As one part of the signature, the identifier sends $(y_R^r, h^r g^y)$, which is the El-Gamal encryption of z. A group manager can thus recover z, allowing it to determine the identity of the signer. The signing process requires knowledge of x and y.

The group manager can be split into an issuer and an escrow agency as follows. The issuer generates $n = pq$ and the corresponding RSA exponents. It then registers n with the escrow agency. The escrow agency chooses G, g, h, ρ and y_R appropriately and sends G, g, h, r_R to the issuer, keeping ρ private. It is typically not hard to verify that G is of the correct order. Also, the issuer can ensure that h is "random enough" by requiring that is be, for example, a hash of g.

The issuer can register participants without knowing ρ. The escrow agency can determine z without knowing the factorization of n. Once it has recovered z, it must go back to the issuer to determine who actually corresponds to z.

3.2 Features obtained by the Camenisch-Stadler construction

First, we note that for this and all the (serious) protocols proposed in this paper, we argue purely heuristically. All statements about security are implicitly preceded by, "Well, it sure looks to us that...". An interesting open question is to obtain efficient schemes based on well known hardness assumptions.

The above construction achieves three of our desired features, and achieves a weak form of the other two. The validity of the first-tier identification and the security of the second-tier identification derive from the corresponding security properties of group signature schemes. Furthermore, the Camenisch-Stadler construction also achieves a strong resistance to impersonation. The group center cannot forge a message from a group member, essentially because the group manager never learns x and y; this inability extends to the issuer and the escrow agency (even working in concert).

Guaranteed escrow of the second-tier identification is in a sense also guaranteed by the properties of group signature schemes. However, note that the the escrow agency and the issuer must work in concert to reveal the identity. Thus, if the issuer later refuses to help, or no longer exists, no second-tier identification may be obtained. This problem may be ameliorated by having the issuer continually inform the escrow agency of the identity corresponding to each z. For some applications, this weakness may be a strength, in that it provides another layer of protection for the anonymity of the identifier.

A weak form of separability obtained, in that the escrow agency is not involved with any transaction. However, the escrow agency has to be contacted to initialize a group. For some applications, this may be reasonable, but it is not suitable when forming groups should be a lightweight operation and the escrow agency is to be kept dormant nearly all the time.

4 A scheme achieving full separation

The greatest limitation of the group-signature based schemes is that the issuer and escrow agent cannot be completely separated - they must communicate when the system is set up and whenever anonymity must be revoked. It is of interest to see how much separation is indeed possible. Using general zero-knowledge proofs for NP assertions, great flexibility may be obtained, though with a complete loss of practicality. To allow person X into the group, the issuer can simply sign a message stating that X is in the group. To identify itself, the identifier simply (probabilistically) encrypts this message and signature in the escrow agent's public key, and gives a zero-knowledge proof that were the message decrypted it would be a valid signed message authorizing entry (further details, such as avoiding resending attacks, omitted).

We give an escrowed identification system that is vastly more efficient than the above system, but substantially less efficient than the group-signature based schemes. In its basic form, it allows for impersonation attacks by the issuer; as recent work in progress we believe we can eliminate this attack with a somewhat more complex scheme.

4.1 The protocols

Central to our protocols is an RSA public key consisting of $n = pq$ and e (relatively prime to $\phi(n)$), with a secret key d such that $de = 1 \mod (p-1)(q-1)$. In addition there is a new parameter, δ, that can be either set randomly or to a fixed number different from 0 or 1 (but must be fixed throughout the execution of the scheme).

A certificate is a pair (a, b) such that $a^e - b^e = \delta$. Given d, one can easily generate a certificate, even if a or a^e is fixed, since $b = (a^e - \delta)^d \mod n$. We assume that a and b are relatively prime to n. The structure of these certificates follows closely the methodology of [11] (and was independently put forth in [20]).

4.1.1 Initializing the system

To set up the system, the issuer chooses n, e, d and δ as above, and publishes (n, e, d). Note that unlike the previous scheme, the escrow agency has no part in setting up the system. We only assume that any potential escrow agent has a public key.

4.1.2 Issuing a certificate

The center chooses a valid certificate (a, b) such that a^e contains (say, in its low order bits) the name of the identifier, and gives (a, b) to the identifier. As another security check, a can actually contain a compact signature by the issuer; anyone can verify that only the issuer made a.

4.1.3 Checking the weak (first tier) identity

On a high level, the identifier proves that he knows a proper certificate (a, b) by a cut-and-choose protocol. The identifier and the verifier may choose their escrow agent independently during each identification session.

First, the identifier chooses independently and uniformly at random two numbers a_1, b_1 both relatively prime to n. He then sets a_2, b_2 to be the numbers satisfying $a = a_1 a_2$ and $b = b_1 b_2$. This partition is done to later hide the actual value of a and b from the verifier. The identifier also chooses uniformly and independently at random two numbers x, y such that x and y are relatively prime to n.

The identifier commits to the values of a_1, a_2, b_1, b_2, $(a_1)^e$, $(a_2)^e$, $(b_1)^e$, $(b_2)^e$, x, $x(a_1)^e$, $x(b_1)^e$, and $x(a_1 a_2)^e + y$, and $x(b_1 b_2)^e + y$. He commits to $(a_1)^e$ and $(a_2)^e$ by probabilistic encryption, using the chosen escrow agents public key. He commits to the other variables using statistical zero-knowledge commitments. The following five tests are used by the verifier to check that the committed values are correct and that the implied $a = a_1 a_2$ and $b = b_1 b_2$ satisfy $a^e - b^e = \delta$. The verifier will pick one of them at random and check that it holds.

1. The identifier opens the commitments on x, a_1, $(a_1)^e$, $a_1 x$, b_1, $(b_1)^e$ and $b_1 x$, and the verifier checks that all the values match their supposed relations.

2. The identifier opens the commitments on a_2, $(a_2)^e$, b_2, $(b_2)^e$ and $b_1 x$, and the verifier checks that all the values match their supposed relations.

3. The identifier opens the commitments on $x(a_1)^e$, $(a_2)^e$, y and $x(a_1 a_2)^e + y$, and the verifier checks that the values match their supposed relations.

4. The identifier opens the commitments on $x(b_1)^e$, $(b_2)^e$, y and $x(b_1 b_2)^e + y$, and the verifier checks that the values match their supposed relations.

5. The verifier opens the commitments on x, on $x(a_1 a_2)^e + y$, and on $x(b_1 b_2)^e + y$, and the verifier checks that x is relatively prime to n and that
$$(x(a_1 a_2)^e + y) \ - \ (x(b_1 b_2)^e + y) \ = \ \delta x.$$

Note that the tests all simultaneously hold only if $(a_1a_2)^e - (b_1b_2)^e = \delta$. Thus, a cheating identifier is caught with probability 1/5. The error probability can be decreased to ϵ by repeating this protocol $O(\log(1/\epsilon))$ times; by choosing the constant appropriately, we have that either two-thirds of the committed $(a = a_1a_2, b = b_1b_2)$ (they may have different values in each iteration) are good or the verifier accepts with probability at most ϵ. Furthermore, identifier and the verifier go through a standard proof by which the identifier can show with high confidence that most of the committed values of $a = a_1a_2$ and $b = b_1b_2$ have the same value (details omitted).

Also, if the identifier has a good certificate pair (a, b), then he can always pass all tests. Last, the view of the verifier in each of the tests can be simulated efficiently.

4.1.4 Recovering the complete (second tier) identity

The verifier gives the transcript of the proof to the escrow agent. Since the commitment on the $(a_1)^e$ and $(a_2)^e$ were done by probabilistic encryption, using the escrow agent's public key, the escrow agent can read the value of $(a_1)^e$ and $(a_2)^e$ and thus get a^e. This value plainly reveals the identity of the user; no further consultation with the issuer is required. One subtlety is that there may be multiple values, either because the prover cheated successfully in some rounds or because multiple identifiers colluded. These are thwarted by having the equality check; nearly all the recovered a^e's will be the same, and will be equal to an a from a certificate known to the identifier.

4.2 Features of the identification system

As with the group-signature based schemes, all of the arguments for the security of this scheme is heuristic; the same caveats about statements of security apply. Some necessary hardness assumptions that we make are that it is hard to find a pair (a, b) so that $a^e - b^e = \delta \mod n$ (so no one can fake an identity), that given (a, b) satisfying the above, it is hard to produce a different pair (a', b') with the same property (so the user must escrow the real a^e), and that given a^e it is not possible to find the appropriate (a, b) (so that the escrow agency cannot fake the user identity). The last condition is implied by the first one, since it is easy to produce a^e for any arbitrary a. Further discussion of this assumption is given in [11].

It appears difficult for an outsider to mimic a group member. The interactive proof does imply that the intruder has a valid certificate. Assuming that these are hard to generate (given all the side information available to an attacker), valid first-tier identification appears to hold. Similarly, the zero-knowledge proof implies that with high probability the escrow agent will be able to recover a^e for a valid certificate, implying that the second-tier identity has been escrowed. However, without the escrow agent, the proof of knowledge of a certificate is zero knowledge, so the second-tier identity is secure.

The strongest feature of the protocol is its separability. The escrow agent is completely uninvolved unless asked to revoke anonymity. Indeed, the identifier and verifier have complete freedom of who they pick as their agent for any individual transaction. Anyone with a public key known to the identifier and verifier may be designated an escrow agent, with no prior interaction required.

The weakest feature of the protocol as it now stands is its resistance to impersonation. The issuer can forge any player's identity as soon as a certificate is issued. However, it can be verified that knowing $(a_1)^e$ and $(a_2)^e$ and seeing the rest of the proof reveals nothing about b. Hence, even after determining the identifier's identity, the escrow agency and the verifier cannot team up to impersonate the identifier. Hence, this system works best when the issuer is under high security and preferably is destroyed (erases its private data) when no further certificates are to be issued. We note that since it's sole operation is an RSA decryption, it can be implemented via group cryptography, increasing the security of the system. We also note as a result of this weakness, the escrow agent's goal is to determine the identifier, but not to prove this identity.

4.3 Recent enhancements

We report on work in progress that will be described in detail in an upcoming longer version of this paper. We believe we can modify the above protocol, with some loss of efficiency, to make it resistant to impersonation. We briefly describe the basic tricks involved.

First, one can achieve resistance to impersonation in a similar manner as in [11] by having a be g^r for some element $g \in Z_n^*$, where the random r is chosen by the identifier. Care must be taken so that this discrete log problem is hard. As part of the identification process, the identifier proves knowledge of r, which is never revealed to anyone.

Unfortunately, this way of choosing a doesn't allow the identifier's name to be encoded. So instead, three certificates (a_1, b_1), (a_2, b_2) and (a_3, b_3)

are generated, such that the identifier knows the discrete log of a_1, and a_3 contains his identity.

A potential impersonation attack by the issuer would simply substitute a new value of a_1 whose discrete log is known by the issuer. However, given the existence of space efficient signature schemes, a_3 and a_1 can be "linked" by the identifier, so that only that value of a_1 can be used to establish someone's identity, and only the identifier can make such a link. Essentially, a_3 contains the identifier's signature for a_1. However, it isn't possible to directly prove in zero knowledge that these committed pairs are linked (they aren't normally revealed). One attack is for two identifiers to mix their pairs, so that no linked pairs are recovered. To thwart this attack, a_1, a_2 and a_3 are additionally constrained so that $a_3 = a_1 + a_2$, and this relation is proven during the identification protocol, using standard cut and choose techniques. The issuer can arrange things so that for any 3 valid certificates $a_3 = a_1 + a_2$ implies that a_3 and a_1 are linked. Many, many details omitted.

5 An El-Gamal based identification system

Given the rather nonstandard and quite similar assumptions used by the previous two schemes, one would like to make sure that plausible schemes can be based on alternative cryptographic functions. We construct an identity escrow system using the El-Gamal signature scheme as the underlying cryptographic primitive. Unfortunately, the protocol is even more inefficient than the last, and does not enjoy its strengths, but serves as evidence that identity escrow does not rely on what is essentially a single nonstandard assumption.

We first give a high level discussion of this scheme.

5.0.1 Initializing the system

The issuer begins by choosing keys for the encryption scheme and for the signature scheme. For both schemes he chooses a big prime p satisfying $p = 2q + 1$ for a prime q, and a random quadratic residuosity g in Z_p^*. The issuer then chooses a secret key S for the signature scheme and computes the related public key $P = g^S \mod p$. The escrow agent chooses a secret key X for its encryption scheme and computes $Y = g^X \mod p$. The issuer publishes g, p, P, Y. In the sequel all operations are done modulo p unless otherwise stated.

5.0.2 Issuing a certificate

The valid certificates will be the set of all signatures on the number 1. Namely, legitimate identities will be all pairs (a, b) satisfying $P^a a^b = g \mod p$. The issuer selects a random signature of "1" which is a pair (a, b), and sends (a, b) to the identifier. Specifically, the issuer chooses a random number $r \in [0..q-1]$ and computes $a = g^r$. (Note that a is a random quadratic residue modulo p.) The issuer tries again if $a = q$,[1] otherwise, the issuer computes the number b satisfying $aS + rb = 1 \mod q$. (a standard calculation in the field Z_q.) The issuer sends (a, b) to the identifier and saves a to allow it to help the escrow agent revoke anonymity.

5.0.3 Checking the weak (first tier) identity

We go into the details of this process in Section 5.1 below. But in a nutshell, in order to identify himself, the identifier provides an El-Gamal encryption of a, and then proves in perfect zero knowledge that he knows a pair (a, b) such that $P^a a^b = g \mod p$ and such that a is encrypted in the cipher-text he provided.

5.0.4 Recovering the complete (second tier) identity

If the identity of a user has to be revealed, the verifier sends the escrow agent the encryption of a. The escrow agent decrypts it and, with the help of the issuer, determines which identifier had that value of a.

5.1 Verifying the identity in zero knowledge

Let us get into the details of the identity verification process. Recall that Party A should not get any knowledge from the interaction with U, but only be convinced that U is a proper user. To this end, U commits on a few numbers, and by A's request, U opens a few of them. A learns nothing from seeing the opened commitments, but if U tries to cheat, A catches him with a constant probability. Thus, repeating the process $O(\log(1/\epsilon))$ times, A is convinced that indeed U has a proper identity with probability $1 - \epsilon$.

User U begins by encrypting a, i.e., selecting uniformly at random $R \in Z_q$ and computing the encryption $(\alpha, \beta) = (g^R, Y^R \cdot a)$, which he sends to A. Next, U partitions a, b and R into shares in the following manner. For b and for R the user U chooses a random sum modulo q. Namely, he chooses

[1]The number a equals q with (negligible) probability $2/(p-1)$.

uniformly at random $b_1, R_1 \in Z_q$, and then sets $b_2 = b - b_1 \mod q$ and $R_2 = R - R_1 \mod q$. The value of a is partitioned in a more involved manner. User U splits it into a product $a_1 \cdot a_2$ which equals a both modulo p and modulo q. He does this in the following manner. U chooses uniformly at random a number $a_1 \in [1..pq-1]$ such that a_1 is relatively prime to pq. Next, U chooses the unique $a_2 \in [1..pq-1]$ satisfying $a_2 = a/a_1 \mod p$ and $a_2 = a/a_1 \mod q$. This can be done by the Chinese Remainder Theorem. Note that for any fixed a (which is assumed to be relatively prime to pq), a_2 (as well as a_1) is randomly distributed amongst the numbers in $[1..pq-1]$ which are relatively prime to pq.

We first describe the tests on a high level; a more detailed explanation follows. We first specify some of the commitments that U makes. These commitments are the ones needed to state the high level tests, but more commitments will be required by the implementations of these tests.

5.1.1 Commitments

U commits to each of the following values: a_1, a_2, b_1, b_2, R_1, R_2, Y^{R_1}, Y^{R_2}, $(a_1)^{b_1}$, $(a_1)^{b_2}$, $(a_2)^{b_1}$, $(a_2)^{b_2}$, g^{R_1}, g^{R_2}, Y^{a_1}, Y^{a_2}.

5.1.2 Tests

We describe on a high level a set of checks. A picks one check at random, and U proves that the test holds by opening some of his commitments. There are 34 low-level tests which are described in high level by the following 6 tests:

1. A multiplication test that $Y^{R_1} \cdot Y^{R_2} \cdot a_1 \cdot a_2 = \beta$. Note that each of the multiplicands is a random number that can be simulated, and g is public. (This consists of 6 basic tests which are described in Subsection 5.1.3 below.)

2. A multiplication test that $g^{R_1} \cdot g^{R_2} = \alpha$. Note that each of the multiplicands is a random number that can be simulated, and g and α are public. (This consists of 5 basic tests which are described in Subsection 5.1.3 below.)

3. A multiplication test that $(P^{a_1})^{a_2} \cdot (a_1)^{b_1} \cdot (a_2)^{b_1} \cdot (a_1)^{b_2} \cdot (a_2)^{b_2} = g$. (This consists of 8 tests which are described in Subsection 5.1.3 below.)

4. U proves to A that $a_1 \cdot a_2 \mod pq$ is a number in the range $1..p-1$. See Section 5.1.4 for the details of implementing this test (which consists of 9 basic tests). A discussion of why this is a crucial test appears in the appendix of [20].

5. For $i = 1, 2$ and for $j = 1, 2$ the user U opens the commitments on a_i on b_j and on $(a_i)^{b_j}$ and A checks that indeed the value of the exponentiation is correct. (These are 4 basic tests.)

6. For $i = 1, 2$ the user U opens the commitments on R_i, a_i and on g^{R_i} and on Y^{a_i} and A checks that both exponentiations are correct. (These are two basic tests.)

These techniques are quite standard, and one may check that seeing one of these tests is perfectly simulatable. Also, if all tests hold then the multiplications hold as well. And finally, if the multiplications hold, and the user follows the protocol as above, then he never fails to convince A.

5.1.3 Implementing the multiplication tests

Let us describe the standard manner in which the multiplication tests are implemented. In these tests, at most one of the operands is revealed; we use the test in situations where this leakage does not pose a problem. The first test we are interested in is a multiplication test that $Y^{R_1} \cdot Y^{R_2} \cdot a_1 \cdot a_2 = \beta$. The value of β is given to A. To this end U chooses uniformly at random and independently 4 numbers t_1, t_2, t_3, t_4 in Z_p^*. U commits on the values of $t_1 Y^{R_1}$, $t_2 Y^{R_2}$, $t_3 a_1$, $t_4 a_2$, and $t_1 t_2 t_3 t_4$ all modulo p. The following 6 tests check the multiplication.

1. U opens the commitments on t_1, Y^{R_1}, and $t_1 Y^{R_1}$ and A checks that the values match.

2. U opens the commitments on t_2, Y^{R_2}, and $t_2 Y^{R_2}$ and A checks that the values match.

3. U opens the commitments on t_3, a_1, and $t_3 a_1$ and A checks that the values match.

4. U opens the commitments on t_4, a_2, and $t_4 a_2$ and A checks that the values match.

5. U opens the commitments on t_1, t_2, t_3, t_4, and $t_1 t_2 t_3 t_4$ and A checks that the values match.

6. U opens the commitments on $t_1Y^{R_1}$, $t_2Y^{R_2}$, t_3a_1, t_4a_2, $t_1t_2t_3t_4$, and U checks that the multiplication $t_1Y^{R_1}\cdot t_2Y^{R_2}\cdot t_3a_1\cdot t_4a_2$ equals $t_1t_2t_3t_4\beta$.

In a similar manner one can construct 5 basic tests and the corresponding commitments to check that $g^{R_1}g^{R_2}=\alpha$.

The second multiplication test should check that $(P^{a_1})^{a_2}\cdot(a_1)^{b_1}\cdot(a_2)^{b_1}\cdot(a_1)^{b_2}\cdot(a_2)^{b_2}=g$. For this test, U chooses independently and uniformly at random 5 numbers t_5,t_6,t_7,t_8,t_9 in Z_p^*. U commits on each of these 5 values and also on $P^{a_1}t_5$, $(P^{a_1})^{a_2}\cdot {t_5}^{a_2}$, $(a_1)^{b_1}t_6$, $(a_2)^{b_1}t_7$, $(a_1)^{b_2}t_8$, and $(a_2)^{b_2}t_9$, and on the value of $(t_5)^{a_2}t_6t_7t_8t_9$. The following 8 tests check the validity of the commitments and the correctness of the multiplication asserted.

1. U opens the commitments on t_5, on a_1 and on $P^{a_1}t_5$, and A checks that the values match.

2. U opens the commitments on $P^{a_1}t_5$, on a_2, and on $(P^{a_1})^{a_2}\cdot {t_5}^{a_2}$ and A checks that the values match.

3. U opens the commitments on t_6, on $(a_1)^{b_1}$ and on $(a_1)^{b_1}t_6$, and A checks that the values match.

4. U opens the commitments on t_7, on $(a_2)^{b_1}$, and on $(a_2)^{b_1}t_7$, and A checks that the values match.

5. U opens the commitments on t_8, on $(a_1)^{b_2}$ and on $(a_1)^{b_2}t_8$ and A checks that the values match.

6. U opens the commitments on t_9, on $(a_2)^{b_2}$, and on $(a_2)^{b_2}t_9$ and A checks that the values match.

7. U opens the commitments on values of all $t_5,t_6,\ldots,t_9$, on the value of a_2 and on the value of the product $(t_5)^{a_2}t_6t_7t_8t_9$, and A checks that the values match.

8. U opens the commitments on $(P^{a_1})^{a_2}\cdot {t_5}^{a_2}$, $(a_1)^{b_1}t_6$, $(a_2)^{b_1}t_7$, $(a_1)^{b_2}t_8$, and $(a_2)^{b_2}t_9$, and on the value of $(t_5)^{a_2}t_6t_7t_8t_9$. A checks that the product
$$(P^{a_1})^{a_2}\cdot {t_5}^{a_2}\cdot(a_1)^{b_1}t_6\cdot(a_2)^{b_1}t_7\cdot(a_1)^{b_2}t_8\cdot(a_2)^{b_2}t_9$$
equals the product

5.1.4 Testing a range property modulo $n = pq$

Let q, p be two primes such that $q < p$. A useful tool in our system is a zero knowledge test which verifies that a given pair of numbers $a_1, a_2 \in [0, 1, \ldots, pq-1]$ satisfies that $a_1 a_2 \mod pq$ is a number in the range $[0..p-1]$. A solution to this problem, for a general range, is given by Bellare and Goldwasser [2]; for greatest efficiency they use an improvement due to Cramer based on the techniques of [9]. In their scenario, the prover commits on the value a (which has to be in the right range) by committing on each of the bits in its binary representation. However, these and other such protocols depend intimately on how the value is committed to; the commitment method we use (a is committed to as a product of committed values, a_1 and a_2) precludes the direct use of this solution. In [20], we give a simple cut-and-choose type proof for this commitment format; which is omitted here due to space limitations.

5.2 Features of the identification system

The interactive proof establishes that the identifier has a valid certificate (a, b) and that he has escrowed the value of a. We know of no way of generating valid certificates, or of generating a new certificate from a number of other valid certificates. Thus, heuristically, this argues that the first-tier identification is valid and that the second-tier identification has been escrowed. However, we note that as with the group-signature based scheme, the escrow agent needs the issuer's help to revoke anonymity. Without the escrow agent, the zero-knowledge proof only reveals the El Gamal probabilistic encryption of a valid certificate, so the second-tier identity seems secure.

The protocol is weakly separable in the same way as with the group-signature scheme: the escrow agent must be involved during the initialization and needs the issuer's help to revoke anonymity.

The escrow agent only sees a, but doesn't receive b. However, the issuer can impersonate any identifier.

Acknowledgements

We thank Markus Stadler for useful discussions.

References

[1] Asokan, Shoup and Waidner. Optimistic Fair Exchange of Digital Signatures. IBM Research Report RZ2973, November 17, 1997.

[2] M. Bellare and S. Goldwasser. Verifiable partial key escrow. *Proceedings of the Fourth Annual Conference on Computer and Communications Security*, ACM, 1997. Preliminary version appeared as Technical Report CS95-447, Dept. of CS and Engineering, UCSD, October 1995.

[3] M. Bellare and S. Goldwasser. Encapsulated key escrow. MIT Laboratory for Computer Science Technical Report 688, April 1996.

[4] M. Ben-Or, S. Goldwasser and A. Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computations. In *Proc. of the 20th Annu. Symposium on the Theory of Computing*, pages 1–10, 1988.

[5] E. Brickell, P. Gemmel and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. In Proc. 6th Symposium on Discrete Algorithms, 1995, pp. 457-466

[6] D. Boneh and M. Franklin. Efficient generation of shared RSA keys. Advances in Cryptology – CRYPTO '97 Proceedings, pp. 425-439. Lecture notes in Computer Science #1294, Springer Verlag, Berlin, 1997.

[7] G. Brassard, D. Chaum and C. Crépeau. *Minimum Disclosure Proofs of Knowledge.* In *JCSS*, pages 156–189. 1988.

[8] D. Chaum, C. Crepau, and I. Dåmgard. Multiparty unconditionally secure protocols. In *Proc. of the 20th Annu. ACM Symp. on the Theory of Computing*, pages 11–19, 1988.

[9] R. Cramer, I. Damgård and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. Advances in Cryptology – CRYPTO '94 Proceedings, pp. 174–187. Lecture Notes in Computer Science #839, Berlin: Springer-Verlag, 1994.

[10] Camenisch. Efficient and generalized group signatures. Advances in Cryptology — EUROCRYPT '97, volume 1233 of Lecture Notes in Computer Science, pages 465–479. Springer Verlag, 1997.

[11] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. Advances in Cryptology – CRYPTO '97 Proceedings, pp. 410–424. Lecture notes in Computer Science #1294, Springer Verlag, Berlin, 1997.

[12] D. Chaum and E. van Heyst. Group signatures. Advances in Cryptology — EUROCRYPT '91, volume 547 of Lecture Notes in Computer Science, pages 257– 265. Springer-Verlag, 1991.

[13] L. Chen and T. P. Pedersen. New group signature schemes. Advances in Cryptology — EUROCRYPT '94, volume 950 of Lecture Notes in Computer Science, pages 171–181. Springer-Verlag, 1995.

[14] I. Dåmgard, T. Pedersen and B. Pfitzmann. On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures. Advances in Cryptology – CRYPTO '93 Proceedings, pp. 250-265. Lecture Notes in Computer Science #773, Berlin: Springer-Verlag, 1994.

[15] Yvo Desmedt and Yair Frankel. Theshold cryptosystems. Advances in Cryptology – CRYPTO '89 Proceedings, pp. 307–315. Berlin: Springer-Verlag, 1990.

[16] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. Advances in Cryptology – CRYPTO '89 Proceedings, pp. 10–18. Berlin: Springer-Verlag, 1985.

[17] Y. Frankel, Y. Tsiounis and M. Yung. "Indirect Discourse Proofs": Achieving Efficient Fair Off-Line E-Cash. Advances in Cryptology–ASIACRYPT '96 proceedings, pp. 286–300. Lecture Notes in Computer Science #1163. Springer-Verlag, 19851996.

[18] Y. Frankel and M. Yung. Escrow Encryption Systems Visited: Attacks, Analysis and Designs. Advances in Cryptology – CRYPTO '95 Proceedings, Berlin: Springer-Verlag, 1995.

[19] S. Goldwasser and S. Micali. Probabilistic Encryption. In *JCSS* Vol 28(2), pages 270-299, 1984.

[20] J. Kilian and E. Petrank. Identity Escrow. Theory of Cryptography Library, `ftp://theory.lcs.mit.edu/pub/tcryptol/97-11.ps`, August 1997.

[21] J. Kilian and F. T. Leighton. Fair Cryptosystems, Revisited. Advances in Cryptology – CRYPTO '95 Proceedings, Berlin: Springer-Verlag, 1995.

[22] F. T. Leighton. Failsafe key escrow systems. Technical Memo 483, MIT Lab. for Computer Science, August 1994.

[23] A. Lenstra, P. Winkler and Y. Yacobi. A Key Escrow System with Warrant Bounds. Advances in Cryptology – CRYPTO '95 Proceedings, Berlin: Springer-Verlag, 1995.

[24] S. Micali Fair public-key cryptosystems. Advances in Cryptology – CRYPTO '92 Proceedings, Berlin: Springer-Verlag, 1993.

[25] S. Micali. Fair public-key cryptosystems. Technical Report 579, MIT Lab. for Computer Science, September 1993.

[26] S. Micali. Certified E-Mail With Invisible Post Offices. Talk at Workshop on Secure Computation, Weizmann Institute, June, 1998.

[27] S. Micali and R. Sydney. A Simple Method for Generating and Sharing Pseudo-Random Functions, with Applications to Clipper-like Key Escrow Systems. Advances in Cryptology – CRYPTO '95 Proceedings, Berlin: Springer-Verlag, 1995.

[28] H. Petersen. How to convert any digital signature scheme into a group signature scheme. Security Protocols Workshop, Paris, 1997.

[29] J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. In proceedings, ESORICS: European Symposium on Research in Computer Security", Springer-Verlag, 1996.

[30] A. De Santis, Y. Desmedt, Y. Frankel and M. Yung. How to Share a Function Securely (Extended Summary). Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, pp. 522–533, Montréal, Québec, May 23–25, 1994.

[31] Schneier, B. (1993). *Applied Cryptography.* John Wiley.

[32] M. Stadler, J.-M. Piveteau and J. Camenisch Fair blind signatures. In Proc. Eurocrypt 95, 1995, LNCS 921, pp. 209 - 219

[33] Adam Young and Moti Yung. Auto-Recoverable Auto-Certifiable Cryptosystems. Eurocrypt 98, LNCS 1403 (Ed. K. Nyberg), pp. 17–32.