DIMACS Technical Report 97-28
July 1997

Identity Escrow

by

Joe Kilian[1]
NEC Research Institute
E-mail: joe@research.nj.nec.com

Erez Petrank [2]
DIMACS Center, P.O.Box 1179,
Piscataway, NJ 08855.
Email: erez@dimacs.rutgers.edu.

[1]Permanent Member
[2]DIMACS post-doctoral fellow

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs, Bellcore, and Bell Labs.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

We introduce the notion of *escrowed identity*, an application of key-escrow ideas to the problem of identification. 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 give protocols for escrowed identity based on the El-Gamal (signature and encryption) schemes and on the RSA function. A useful feature of our protocol is that after setting up A to use the system, E is only involved when it is actually needed to determine A's identity.

Keywords: Cryptography, Key escrow, Proofs of identity.

1 Introduction

Key escrow has proven an active and contentious field of research and discussion (c.f. [18, 19, 16, 13, 15, 17, 20]). Essentially all 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. We consider a new domain for the application of key escrow ideas: Escrow in identity schemes.

Some distinctive features of this application are that

- Escrowed identity may actually enhance privacy. By default, many identity schemes often 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 Motivation

The motivation for traditional identification schemes in access control, communication and commerce need not be discussed. A rapidly increasing number of interactions take place on the purely information level, and are likely to become ubiquitous. As just one example, on some toll bridges one can now electronically identify oneself to a toll booth instead of waiting in line to pay a toll. Although this application is cost effective (or will be) and more convenient, it raises critical issues of privacy; a more pervasive toll collection system might as a side effect allow the tracing of people's movements to an unprecedented degree. (See Subsection 1.3 for comparison with the solutions proposed for electronic cash).

We address the tension between party A's desire for privacy and anonymity and B's (and society's) desire to know who they are interacting with. This balance point of this tension can shift under certain often rare circumstances. For example, consider an automated parking garage. What is paramount to the garage is 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 messages: 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.

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.

1.2 Escrowed Identity

We consider a more flexible, two-tier approach to identificatio. On the lower tier, a person gives only as much information about themselves as is strictly necessary for ordinary circumstances. In the garage exampe above, this means proving authorization to use the garage. 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, i.e., the escrow agency, which is preferably separate from and not under the control of the party managing access (such as the garage owners).

An escrowed identity system consists of the following parties:

Identifier: The identifier is the party who identifies themselves to the verifier. (The user of the garage in the above example.)

Issuer: The issuer issues certificates that allow the identifier to identify themselves 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. (This is the garage keeper in the above example.)

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:

Issuing a certificate: In this protocol, the certificate issuer gives a certificate to the identifier.

Checking the weak identity: In this protocol, 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: In this protocol, the verifier gives the escrow agent(s) the escrowed certificate, and the escrow agent(s) recover the original certificate.

Our demands from a secure identity escrow system are given formally in Section 2, but let us say informally what we expect from such a scheme.

The identification procedure works: 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.

Faking a legitimate identity is hard: 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.

The escrowed identity is hidden: The checking identity protocol is zero knowledge to the verifier. Namely, the verifier gets only the fact that the user has a certificate and nothing else.

Ducking the escrow is hard: The probability that a computationally bounded user passes the check for weak identity but his identity cannot be recovered from the transcript by the escrow agents is almost 0.

Enhanced security: Even the escrow agency, after recovering many identities of many users from the transcripts of weak identity proofs, cannot fake any legitimate identity in the sense described above. This implies that the escrowed identity does not reveal the certificate of the user.

As well as introducing the notion of escrowed identity we give two concrete implementations. One is based on the RSA encryption scheme and the other on the El-Gamal signature and encryption schemes. We do not make use of the inefficient general constructions for zero-knowledge proofs, but instead use in a strong way the algebraic properties of RSA and El-Gamal.

Remarks: In the El-Gamal based implementation described in this paper the issuer and the escrow agent are the same, but in the RSA based implementation we allow them to be distinct. We believe it may be possible to extend our El-Gamal protocol to allow for this distinctness, though perhaps with more heuristic security assumptions.

We also note that it is somewhat less crucial for the Escrow Agents to be more than one person. There is a further check on a rogue escrow agent, which 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 our protocols can be made to work using simple group cryptography (e.g. [9, 21]).

It might also be useful for the identity issuer to be implemented using group cryptography. For the RSA based scheme, the issuer needs to create a standard RSA public key, agree on a random string and perform RSA decryptions. Recently, all the necessary RSA operations (including setting up the public key) have been implemented in the group cryptography framework [4].

1.3 Related work on electronic cash

Some works on electronic commerce considered a similar problem. One particular such paper is by Frankel, Tsiounis and Yung [12]. In this work, a protocol is given for electronic cash transactions for which a trusted party can later reveal the identity of the payer. This is the first work that allows the transaction to be made without the trusted party playing any role during the transaction itself, and thus has the flavour of identity escrow: The identity of the user is not revealed during the transaction, but can be determined later by the trusted party.

Although this task seems to be of similar flavour, we note there is one imprtant difference between this type of transaction and identity escrow. The user of electronic cash is allowed to use his cash only once. If he uses his coins more than once, all guarantees for his privacy become invalid. Usually, two transcripts of such transactions allow efficient computation of the users identity. In the protocol of [12] this complete collapse of security does not happen, but still something bad happens: The verifier will be able to tell if a user identifies more than once. Namely, after a user identifies himself once, the verifier can tell whenever the user comes to visit again. This is something we would not like to have in an identity escrow scheme, and we believe that designing a good identity escrow scheme on top of efficient protocols for electronic transactions is an intersting open problem.

1.4 Road-map

In Section 2 we give the formal definition of an identity escrow system. In Section 3 we describe some of the building blocks we use for our implementations. In Section 4 we show an RSA-based implementation of an escrowed identity scheme. In Section 5 and 5.1 we show an implementation of an escrowed identity scheme based on the El-Gamal encryption and signature schemes. Last, in Section 6 we provide our zero knowledge test to check that a number is small which we need for our El-Gamal based scheme and which might be of independent use.

2 Identity Escrow: Definition

Let us start with a formal definition of an identity escrow scheme.

The parties

As described in the introduction, an escrowed identity system consists of the following parties:

Identifier: The identifier is the party who identifies himself to the verifier.

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

Verifier: The verifier is typically the access provider who verifies the first-tier (regular) identity process as well as the escrow proof for the second-tier identity (i.e., a possibility of revealing the identity of the user in extreme cases according to court order).

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

The protocols

These parties execute the following possible protocols:

Issuing a certificate: In this protocol, the certificate issuer is given his private key, an identity of an identifier that should get a certificate, all the public information (such as public keys in the scheme), and a security parameter. The identifier is given his keys, his identity, the public information, and the security parameter in the input. The issuer and the identifier engage in a protocol, by the end of which the identifier gets a legitimate certificate to be used in the "checking a certificate" protocol.

Checking a certificate: In this protocol, the identifier convinces the verifier that he has a certificate, gives an *escrowed certificate* and convinces the verifier that the escrowed certificate is valid. The input to the identifier is his certificate got from the "issuing a certificate" protocol, his keys, his identity, the public information, and the security parameter. The input to the verifier is the security parameter and the public information.

Recovering the complete identity: In this protocol, the verifier gives the escrow agent(s) the escrowed certificate, and the escrow agent(s) recover the original certificate. The input to the verifier is the transcript of a "checking the certificate" protocol, which he sends to the escrow agents. The escrow agents are given the transcript, their secret keys, the public information, and the security parameter, and they come up with the identity of the verifier in that transcript. They reveal this identity to the authorities and not necessarily to the verifiers.

The requirements

The above three protocols form an identity escrow scheme with error parameter ϵ if the following conditions hold:

The identification procedure works: If the issuer follows his protocol and issues a certificate to User U, and if User U uses this certificate to escrow identity to a verifier V, and if all parties honestly follow their protocols then the verifier is allways convinced that the user has a proper certificate.

Faking a legitimate identity is hard: For any probabilistic polynomial time machine A, if A is given the inputs of the identifier but not including the issued certificate (i.e., Machine A gets the keys of the identifier, the public information, and the security parameter), then A passes a certificate check with the verifier accepting his proof, with probability at most ϵ. The probability is taken over the coin-tosses of the verifier and Machine A in the certificate-check protocol, and over the random choice of keys for the issuer, and identifier. (Note that this implies that producing a certificate without knowing the issuers private key is computationally hard.)

The escrowed identity is hidden: The proof of identity given by the user in the certificate-check protocol is zero knowledge. Namely, if the user has a valid certificate and is following

his role in the protocol, then for any (possibly cheating) probabilistic polynomial time verifier V^*, there exists a probabilistic polynomial time simulator S_{V^*}, such that the view of V^* in the certificate-check protocol (when V^* is input the public information, the security parameter, and any auxiliary input a of length polynomial in the security parameter k) is computationally indistinguishable from the output of S_{V^*} on the same input.

Ducking the escrow is hard: Let A be any probabilistic polynomial time machine. Suppose A gets the inputs of a legitimate identifier in the "checking a certificate" protocol. Namely, A gets a legitimate certificate got from the "issuing a certificate" protocol, his keys, his identity, the public information, and the security parameter. Suppose also that A engages in the "checking a certificate" protocol with a verifier V. Assume also that V follows the protocol of the verifier honestly whereas A may be cheating while playing the role of the identifier. Later, the transcript of the conversation is used in the "recovering the complete identity" protocol between the verifier and the escrow agents, and both the verfier and the escrow agents follow their protocols honestly. Then, the probability that the vefrifier accept in the "checking a certificate" protocol, but the escrow agents fail to come up with the identity of the identifier as input to A, is at most ϵ. Here, the distribution is taken over the choice of keys for the issuer, and the escrow agents, and over the coin tosses of the issuer (while issuing the certificate that A gets) the coin tosses of Machine A and the coin tosses of the verifier.

Enhanced security: Even the escrow agency, after recovering many identities of many users from the transcripts of weak identity proofs, cannot fake any legitimate identity with probability greater than ϵ in the sense described above. This implies that the escrowed identity does not reveal the certificate of the user.

3 Preliminaries

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

3.1 Bit Commitments

We work in the argument framework of Brassard, Chaum and Crépeau [5]. In this paradigm, all parties are assumed to be computationally bounded. It is shown in [5] how to commit to bits in statistical zero-knowledge, based on the intractability of certain number-theoretic problems. Dåmgard, Pedersen and Pfitzmann [8] 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 and is used heavily in our protocol. For simplicity, we will simply speak of committing to and revealing bits when referring to the protocols of [8].

For our RSA based implementation we also commit to strings by probabilistic encryption [14] using the public key of the escrow agency. These commitments are only computationally secure, but allow for the escrow agents to recover the values of these commitments. This

allows for the escrow agents to recover the value of these commitment in addition to those revealed in the course of the zero-knowledge proof.

3.2 The El-Gamal signature and encryption schemes

We base our implementation of escrowed identity on the El-Gamal signature and encryption schemes [10], which we summarize, following [22], 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 $Y = g^X \bmod p$. For the signature scheme we denote the secret key by $S \in Z_q$ and the public key by $P = q^S \bmod p$ (the different variables are purely notational).

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

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

1. The signer uniformly generates $r \in Z_q$, computes $a = g^r \bmod p$, 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 $Sa + rb = M \bmod p-1$.

3. The signer returns (a, b).

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

3.2.1 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. We remark that other weak forgeries (i.e., signing random messages) are known (see [23] Page 206-207).

3.3 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 $\phi(n)$. For our purposes, e should be chosen randomly; we do not use small-exponent RSA. A message M is encrypted by computing $M^e \bmod n$. The private key consists of d such that $de = 1 \bmod (p-1)(q-1)$ (strictly, $de = 1 \bmod \lambda(n)$ suffices), and M^e is decrypted by computing $(M^e)^d = M \bmod n$.

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 $a^e - b^e = \delta \bmod n$. 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 $b = (a^e - \delta)^d$.

4 The RSA-based identification scheme

In this section we implement an identification scheme whose underlying cryptographic scheme is RSA. This scheme is more efficient than the El-Gamal scheme described in Section 5 below. In our RSA based system, the public key consists of $n = pq$ and e (the secret key is d such that $de = 1 \bmod (p-1)(q-1)$) and a new parameter δ. The parameter δ can be either set randomly or to a fixed number different than 0 or 1, but it must be fixed throughout the execution of the scheme. A certificate is a pair (a, b) such that $a^e - b^e = \delta$. Good pairs are trivial to generate by anyone who knows d.

The issuer publishes n, e, and δ. His secret is d. To issue an identity to a new user, the issuer chooses a pair (a, b) such that $a^e - b^e = \delta$. Note that knowing d, the issuer can do that for any choice of a^e. He chooses a^e to be the identity of the new user. The user gets (a, b) as his secret information for future identifications.

In order to prove that he has proper identity, the user proves in zero knowledge that he knows a pair (a, b) satisfying $a^e - b^e = \delta \bmod n$. The proof will be zero knowledge to the verifier, but will disclose the identity of the user, i.e., the value of a^e, to the escrow agency. All calculations are done modulo n.

4.1 The identification process

Here is how a user proves that he knows a proper certificate (a, b). The identifier is going to commit to a few values, and then the verifier chooses a test in which the identifier opens some of the committed values and the verifier checks that the values match their supposed values. To this end, 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 on the values of a_1, a_2, b_1, b_2, $(a_1)^e$, $(a_2)^e$, $(b_1)^e$, $(b_2)^e$, x, y, $x(a_1)^e$, $x(b_1)^e$, $x(a_1 a_2)^e + y$, and $x(b_1 b_2)^e + y$. He commits to $(a_1)^e$ and $(a_2)^e$ by probabilistic encryption, using the escrow agents public key. (Note that the public key of the escrow agent is not the pair (n, e) which is the public key of the issuer.) 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, $(a_1)^e$, $x(a_1)^e$, b_1, $(b_1)^e$ and $x(b_1)^e$, 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 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 $x(a_1a_2)^e + y$, 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 $x(b_1b_2)^e + y$, and the verifier checks that the values match their supposed relations.

5. The identifier opens the commitments on x, on $x(a_1a_2)^e + y$, and on $x(b_1b_2)^e + y$, and the verifier checks that x is relatively prime to n and that

$$x(a_1a_2)^e + y \; - \; (x(b_1b_2)^e + y) \; = \; \delta x.$$

Note that the tests all hold iff $(a_1a_2)^e - (b_1b_2)^e = \delta$. Thus, a cheating identifier is caught with probability 1/5. The error probability can be decreased to ϵ by repeating this protocol $5\log(1/\epsilon)$ times. 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 by an efficient machine.

Since the commitment on the $(a_1)^e$ and $(a_2)^e$ were done by probabilistic encryption, using the escrow agents public key, then the escrow agents can read the value of $(a_1)^e$ and $(a_2)^e$ and thus get a^e which is the identity of the user. Although the protocol is repeated many times, one proper execution is enough to have the identity escrowed. Note that the issuer need not play any role after issuing certificates and the escrow agent need not play any role until being called in to determine someone's identity. Finally, 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.

The hardness assumptions that we make are that it is hard to find a pair (a, b) so that $a^e - b^e = \delta \bmod n$ (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.

5 The El-Gamal identification system

We construct an identity escrow system using the El-Gamal signature scheme as the underlying cryptographic primitive. The identities are distributed by an authorized issuer. 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 residue $g \neq 1$ in Z_p^* (g is a random element of order q). The issuer then chooses a secret key S for the signature scheme and computes the related public key

$P = g^S \bmod p$. The issuer also chooses a secret key X for the encryption scheme and computes $Y = g^X \bmod p$. The issuer publishes g, p, P, Y. In the sequel all operations are done modulo p unless specifically otherwise stated.

Legitimate identities will be the set of all signatures on the number 1. Namely, legitimate identities will be all pairs (a, b) satisfying $P^a a^b = g \bmod p$.

Choosing an identity for a user: To choose an identity for a new user U, the issuer selects a random signature of "1" which is a pair (a, b), and sends (a, b) to U. The issuer checks that a doesn't equal q. If it does, the issuer chooses another signature on 1 so that a is not q. [1] For future identification, the issuer saves the number a together with the name of the user U. This will be U's identifier.

To be more specific, 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 $a = q$, otherwise, the issuer computes the number b satisfying $aS + rb = 1 \bmod q$. (This is standard calculation in the field Z_q.) The issuer sends (a, b) to the user and saves a as the identifier of U.

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

Identifying a user in case of court decision: If the identity of a user has to be revealed, the issuer gets the encryption of a, he decrypts it, and discloses the identity of the user.

Privacy: When a user U identifies himself to a party A, Party A gets zero knowledge from the identification process. In particular, Party A cannot fake U's identity in future interactions. This follows from the fact that in order to identify himself, a user must prove in zero knowledge that he knows (a, b) such that $P^a a^b = g \bmod p$.

We now go on and give the details of the verification process.

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 sum partition at random modulo q. Namely, he chooses uniformly at random $b_1, R_1 \in Z_q$, and then sets

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

$b_2 = b - b_1 \bmod q$ and $R_2 = R - R_1 \bmod q$. 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 $a_2 = a/a_1 \bmod p$ and $a_2 = a/a_1 \bmod q$. 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.

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}.

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 5.2 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 5.2 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 5.2 below.)

4. U proves to A that $a_1 \cdot a_2 \bmod pq$ is a number in the range $1..p-1$. See Section 6 for the details of implementing this test (which consists of 9 basic tests). See also Appendix A to see why this is a crucial test.

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.

5.2 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 $t_1 Y^{R_1}$, $t_2 Y^{R_2}$, $t_3 a_1$, $t_4 a_2$, $t_1 t_2 t_3 t_4$, and U checks that the multiplication $t_1 Y^{R_1} \cdot t_2 Y^{R_2} \cdot t_3 a_1 \cdot t_4 a_2$ equals $t_1 t_2 t_3 t_4 \beta$.

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 $(t_5)^{a_2} t_6 t_7 t_8 t_9$. 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 $(t_5)^{a_2} t_6 t_7 t_8 t_9$, 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 $(t_5)^{a_2} t_6 t_7 t_8 t_9$. 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 $\beta \cdot (t_5)^{a_2} t_6 t_7 t_8 t_9$.

6 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 $a_1 a_2 \bmod pq$ is a number in the range $[0..p-1]$. A solution to this problem, for a general range, is given by Bellare and Goldwasser [1]; for greatest efficiency they use an improvement due to Cramer based on the techniques of [7]. 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. This is not applicable to our case, in which we commit on two random a_1, a_2 whose product is a (the value of interest). We use this specific commitment in our other tests as well. Thus, we provide an alternate scheme which is adequate for our purposes.

Our scheme holds for arbitrary ranges, but for simplicity we describe it for the specific range needed in our identification protocol. We assume a, a_1, a_2 are all relatively prime to pq. We start with the following simple observation.

Observation 6.1 *Let x be a number such that $2^{x+1} < p < 2^{x+2}$. The number $a_1 a_2 \bmod pq$ is in the range $[0..p-1]$ iff it is a sum of a subset of the following multi-set of numbers: $\{0, 2^0, 2^1, \ldots, 2^x, p - 2^{x+1}\}$ modulo pq. (Note that each number may be used in the sum only as many times as it appears in the multi-set.)*

For the zero knowledge property, it would be harmful if each pair (a_1, a_2) had a different number of terms in the summation. Therefore we restate the above observation as:

Observation 6.2 *The number $a_1 a_2 \bmod pq$ is in the range $[0..p-1]$ iff s can be expressed as a sum of $x+3$ numbers of the multiset containing zero $x+2$ times and then the numbers: $\{0, 2^0, 2^1, \ldots, 2^x, p - 2^{x+1}\}$. Summation is done modulo pq. (Note that each of these number may be used in the sum at most once. Thus zero may be used at most $x+3$ times.)*

Denote the multiset of the above observation by W. Next, we would like to present a perfect zero knowledge test for the condition of Observation 6.2. The secret information is the multiplication value $a_1 a_2 \bmod pq$. We do not worry about leakage of one of the numbers in the pair. The prover is going to show that there is a subset of $x+3$ elements in W which sum up to $a_1 a_2$ modulo pq. Note that in order to achieve zero knowledge the verifier should not be allowed to see any of the elements that are used in the sum.

The verifier starts by partitioning each $w \in W$ into w^1 and w^2 such that w^1 is a random element in $[0, pq-1]$ and w^2 satisfies $w^2 = w - w^1 \bmod pq$. The prover then randomly permutes the order of the $2x+5$ elements and commits on the shares of each of these elements. These are $4x+10$ commitments. Denote the committed shares by w_i^j for $j = 1, 2$ and $i = 1, \ldots, 2x+5$. Next the prover chooses the indices $I = \{i_1, \ldots, i_{x+3}\}$ such that the numbers committed on in these places add up to $a_1 a_2 \bmod pq$. These indices exist by Observation 6.2. The prover also commits on these indices. The goal of the prover is to show that $\sum_{i \in I} w_i = a_1 a_2 \bmod pq$. To this end, the prover chooses a random number $r \in [0..pq-1]$ and he will actually show that $\sum_{i \in I} w_i + r = a_1 a_2 + r \bmod pq$. The prover commits also on r, on the result of summing $(\sum_{i \in I} w_i^1) + r$, and on the result of the summation $(\sum_{i \in I} w_i^2) - (a_1 a_2 + r)$. Next, we would like to describe a commitment on $a_1 a_2 + r$ that is checkable without yielding the value of $a_1 a_2$.

We use the Chinese Remainder Theorem. Let t_q be the number in $[0..pq-1]$ which satisfies

$$\begin{aligned} t_q &= 1/a_2 \bmod q \\ t_q &= 0 \bmod p \end{aligned}$$

and let t_p be the number in $[0..pq-1]$ which satisfies

$$\begin{aligned} t_p &= 0 \bmod q \\ t_p &= 1/a_1 \bmod p. \end{aligned}$$

We can always find such t_p and t_q since a_1 and a_2 are relatively prime to pq. We first note that $t_p a_2 + t_q a_1$ equals 1 both modulo p and modulo q. Therefore it is also 1 modulo pq. Thus, we get

$$(a_1 + t_q \cdot r) \cdot (a_2 + t_p \cdot r) = a_1 a_2 + r \bmod pq.$$

The prover commits on $a_1 a_2 + r \bmod pq$, but also to some intermediate results in this calculations so that it can be checked. So the prover also commits on $t_p r, t_q r$, and on the values of $(a_1 + t_q \cdot r)$ and $(a_2 + t_p \cdot r)$ In what follows, we describe 9 tests that check the validity of the committed values and the correctness of the assertion. The following 5 tests verify that indeed the commitment on $a_1 a_2 + r$ is valid.

1. The prover opens the commitment on a_1, on r and on $t_p r$ and the verifier checks that $t_p r$ is properly set.

2. The prover opens the commitment on a_2, on r and on $t_q r$ and the verifier checks that $t_q r$ is properly set.

3. The prover opens the commitment on a_1, on $t_q r$ and on $a_1 + t_q r$ and the verifier checks that the values match.

4. The prover opens the commitment on a_2, on $t_p r$ and on $a_2 + t_p r$ and the verifier checks that the values match.

5. The prover opens the commitment on $a_1 + t_q r$, on $a_2 + t_p r$ and on $a_1 a_2 + r$ and the verifier checks that the later is the product of the first two (modulo pq).

The next 4 tests verify that the summation is proper and the the assertion holds.

1. The prover opens all the commitments on all w_i^j for $j = 1, 2$ and $i = 1, \ldots, 2x + 5$ and the verifier checks that they indeed present a permutation of the multi-set W.

2. The prover opens the commitments on I, on w_i^1 for all $i \in I$, on r, and on the summation $(\sum_{i \in I} w_i^1) + r$, and checks that $|I| = x + 3$ and that the commitment on the summation matches the real sum.

3. The prover opens the commitments on I, on w_i^2 for all $i \in I$, on $a_1 a_2 + r$, and on the summation $(\sum_{i \in I} w_i^2) - (a_1 a_2 + r)$, and checks that $|I| = x + 3$ and that the commitment on the summation matches the real sum.

4. The prover opens the commitments on the summation $(\sum_{i \in I} w_i^1) + r$ and the commitment on the summation $(\sum_{i \in I} w_i^2) - (a_1 a_2 + r)$, and the verifier checks that sum of these two numbers is 0 mod pq.

The verifier chooses at random one of these tests and asks the prover to open the relevant commitments so that he can verify the correctness of the test.

Clearly, if $a_1 a_2$ mod pq is in the range $[0..p-1]$ then the prover can commit on values as instructed by the above protocol and pass any of these tests with probability 1. On the other hand, if all tests hold, then $a_1 a_2$ mod pq must be in the right range. (This assumes that the verifier cannot break the commitment scheme.) Thus, if $a_1 a_2$ mod pq is not in the right range, then at least one of the tests must fail. The verifier will find this out with probability at least $1/9$. The soundness error probability can be decreased by repetitions, but since we only present this as a tool, we leave the amplification to be done by the calling protocol. Last, in each of the 9 possible tests the verifier only gets to see random numbers (in the corresponding ranges) and his view can be perfectly simulated.

References

[1] **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.

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

[3] 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.

[4] "Efficient generation of shared RSA keys", Submitted to Crypto 97.

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

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] G. Davida, Y. Frankel, Y. Tsiounis and M. Yung. Anonymity control in e-cash. In *1st Financial Cryptography conference*, Anguilla, BWI, 1997. Also available at `http://www.ccs.neu.edu/home/yiannis/pubs.html`.

[12] Y. Frankel, Y. Tsiounis and M. Yung. Indirect discourse proofs: achieving fair off-line e-cash. In *Advances in Cryptology, Proc. of Asiacrypt '96 (Lecture Notes in Computer Science 1163)*, Springer-Verlag, 1996, pp. 286–300. PUBLISHER = "Springer-Verlag", International patent pending. Available at `http://www.ccs.neu.edu/home/yiannis/pubs.html`

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

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

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

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

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

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

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

[20] 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.

[21] 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.

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

[23] Stinson, D. R. (1995). *Cryptography Theory and Practice* CRC Press.

A Breaking the El-Gamal-based identification scheme when $a_1 a_2 \bmod pq$ is not smaller than p

In this appendix we note that if we do not check that $a_1 a_2 \bmod pq$ is smaller than p then the user can break the scheme. Namely, the user can prove a proper identity without actually escrowing his key. Recall that the user U has (a, b) such that $P^a a^b = g \bmod p$.

The user chooses at random $1 \leq x \leq q-1$ and computes:

$$\begin{aligned} c_1 &= a + bx \bmod q \\ c_2 &= \frac{a}{P^x} \bmod p \end{aligned}$$

Next, U uses the Chinese Remainder Theorem to compute $c \in [0..pq-1]$ such that $c = c_1 \bmod q$ and $c = c_2 \bmod p$. For this c it holds that

$$P^c c^b = g \bmod p$$

Thus, the user may choose to identify using a_1, a_2 whose product is c, and be able to pass all tests, except of-course for the test that c has to be smaller than p modulo pq.

If, however, $c < p-1$ modulo pq, then we can be certain that having $c \bmod p$ (decrypted by the issuer) gives us a value c satisfying that (c, b) is a signature for the message "1". Faking such a c is equivalent to getting another signature for the message "1".