Verifiable Encryption and Applications to Group Signatures and Signature Sharing

Jan Camenisch Ivan Damgård

March 23, 1999

Abstract

We generalize and improve the security and efficiency of the verifiable encryption scheme of Asokan et al., such that it can rely on more general assumptions, and can be proven secure without assuming random oracles. We show a new application of verifiable encryption to group signatures with separability, these schemes do not need special purpose keys but can work with a wide range of signature, identification, and encryption schemes already in use. Finally, we extend our basic primitive to verifiable threshold and group encryption. By encrypting digital signatures this way, one gets new solutions to the verifiable signature sharing problem.

1 Introduction

A *verifiable encryption scheme* is in its basic form a two-party protocol between a prover P and a verifier V. Their common input are a public encryption key E, a public value x, and a binary relation $\mathcal{R}$. As a result of the protocol, V either rejects or obtains the encryption of some value w under E such that $(x, w) \in \mathcal{R}$ holds. For instance, $\mathcal{R}$ could defined such that $(x, w) \in \mathcal{R}$ if and only if w is a signature on message x w.r.t. to some fixed public key. In other words, P claims to have given V the encryption of a valid signature on x.

The protocol should ensure that V accepts an encryption of an invalid w with only negligible probability. Moreover, V should learns nothing except the encryption of w and the fact that w is valid w.r.t. x. In particular, if the encryption scheme is semantically secure, the protocol should be zero-knowledge.

The encryption key E can belong to P, but typically belongs to a third party – and even in this case the third party should not need to take part in the protocol, in other words P does not need to know the secret key corresponding to E.

Verifiable encryption schemes are employed in many cryptographic protocols (although the term verifiable encryption is not always used). Examples are digital paymentsystems with revokable anonymity (e.g., [3, 18]), verifiable signature sharing (e.g., [19]), (publicly) verifiable secret sharing (e.g., [29]), or fair exchange of signature [1]. However, only the schemes presented in [1, 23, 31] are not ad-hoc constructions using a specific encryption scheme that suits the particular application; in fact, our protocols can be seen as a generalization of these.

The concept of verifiable encryption was introduced in [30] in the context of publicly verifable secret sharing schemes, and in a more general form in [1], for the purpose of fair exchange of signatures. The idea here is that if A and B wish to exchange their signatures on some message, they will first exchange verifiable encryptions of them, using as E the public key of some trusted third party. If this was successful, it will be safe for A to just reveal his signature to B. Even if B never answers, A can get B's signature by having the trusted party decrypt it. (Some care needs to be taken here to avoid that one party falsely accuses the other of cheating, see [1] for details). Micali [24] also proposes the use of provable encryption of data for third parties to solve several variants of the fair exchange problem.

The verifiable encryption scheme from [1] can be based on any secure public key encryption scheme. But it has only been proven secure in the random oracle model, and only works for relations $\mathcal{R}$ containing pairs of form $(x, f^{-1}(x))$, where f is a one-way group homomorphism. In this paper, we show how to modify and generalize that scheme to achieve the following:

- Our scheme can be proved secure without relying on random oracles
- It retains the property that it can be based on any public-key encryption scheme.
- The relation $\mathcal{R}$ can be any relation possessing a 3-move proof of knowledge which is an Arthur-Merlin game, i.e., as the second message, the verifier sends a random challenge. This proof should be honest verifier zero-knowledge, and a cheating prover should be unable to answer more than one challenge correctly. We call this a Σ-protocol in the following. This includes the earlier case of group homomorphisms as a special case. We show that this generalization enables new applications of verifiable encryption.
- The efficiency is improved: the verifiable encryption obtained by V actually consists of several encryptions in the underlying encryption scheme, this is true both of our scheme and of [1]. In [1], the number of encryptions to be stored by V is linear in the security parameter k, while our scheme requires only $O(\log k)$ encryptions to be kept.

Our results are targeted against a situation where a public-key infrastructure already exists, i.e., users already have (certified) public-key pairs for encryption and signatures. However, we assume that these keys are not necessarily designed with other and more advanced primitives in mind, such as group signatures, identity escrow, fair contract signing, and blind signatures with revocable anonymity. We believe this is a very realistic scenario.

We show that our verifiable encryption scheme can be applied to build all of these primitives, where the security can be proved based only on security of the existing infrastructure: no special assumptions are needed on the existing encryption scheme, and the signature scheme only needs to satisfy that one can prove knowledge of a signature on a given message by a Σ-protocol. All standard signature schemes (RSA, El-Gamal, DSS) satisfy this. In particular, our group signature scheme is the first of its kind with these properties. The efficiency is reasonable, we typically need resources corresponding to a linear number of encryptions/signatures in the existing schemes. Of course, solutions with provable security could always be obtained using general

$$
\begin{array}{lcr}
P(x, w) & & V(x) \\
\hline
(t, r) := \sigma_t(x, w) = (f(r), r) \text{ with } r \in_R \{0,1\}^* & & \\
 & \xrightarrow{\quad t \quad} & \\
 & \xleftarrow{\quad c \quad} & c \in_R \{0,1\} \\
s := \sigma_s(x, w, r, c) = \begin{cases} r + w & c = 1 \\ r & c = 0 \end{cases} & & \\
 & \xrightarrow{\quad s \quad} & \\
 & & o := \begin{cases} (t, c, s) & 1 = \sigma_v(x, t, c, s) \\ \perp & \text{otherwise} \end{cases} \\
\hline
 & & (o)
\end{array}
$$

Figure 1: An example of a Σ-protocol: a proof of knowledge of a pre-image under a (one-way) group homomorphism $f(\cdot)$, i.e., $\mathcal{R} = \{(x, f^{-1}(x))\}$. The predicate $\sigma_v(x, t, c, s)$ equals 1 if and only if $f(s) = t \cdot x^c$ holds.

zero-knowledge techniques, but only at a prohibitive loss of efficiency. Also, much more efficient schemes can sometimes be obtained by ad-hoc constructions, but this requires relying on particular properties of the encryption and/or signature scheme involved, and sometimes means that no proofs of security can be given.

We extend our basic primitive to *verifiable group encryption* involving $n > 1$ third parties (called *proxies*) where only certain subsets of them can jointly decrypt the secret. That is we are given n public encryption keys $E_1, \ldots, E_n$. The prover and the verifier further agree any monotone access-structure over $\{1, \ldots, n\}$. Then, if V accepts, he is convinced that he has obtained an encryption of a valid secret, that a subset of the proxies can decrypt if and only if that subset is contained in the access-structure[1]. For example, one can decide that for some $t < n$, any subset of at least t players can decrypt, whereas less than t players cannot. Verifiable group encryption can be used to implement verifiable signature sharing, yielding more general solutions for this problem than what was previously known.

2 Preliminaries

2.1 Σ-Protocols

A Σ-protocol [9, 11] for a boolean relation $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ is a three move honest-verifier zero-knowledge proof of knowledge for $\mathcal{R}$. That is, a string x is common input to prover P and verifier V, and P demonstrates knowledge of a w such that $(x, w) \in \mathcal{R}$. We call w a *witness* for x, and the set of x's that have witnesses is called $L_{\mathcal{R}}$. Of course, carrying out a Σ-protocol for some relation R makes sense only if it is a hard relation for (at least) one of the two involved parties. A Σ-protocol can be

[1]This notion of group encryption should not be confused with the notion of threshold encryption [13]. In the latter case, a number of parties publishes a *single* public key and the access structure is determined by these parties during the setup of the system

defined by three (probabilistic) procedures σ_t, σ_s, and σ_v as follows. Then, on input x and w, P uses σ_t to compute a so-called *commitment* t that the prover sends the verifier as the first message. Also, some side-information r for the prover is produced. Then the verifier sends a random bit string c, called a *challenge*. The prover uses x, w, r, and c as input to σ_s to compute the *response* s. Finally, σ_v is a predicate taking x, t, c, and s as input that V uses to check whether s is a valid response, i.e., V accepts if $\sigma_v(x, t, c, s) = 1$ holds. A triple (t, c, s) such that $\sigma_v(x, t, c, s) = 1$ is called an *accepting* triple for x.

We require that if P and V follow the protocol, V always accepts, whereas a cheating prover can answer at most one challenge correctly per commitment. More precisely, there is some polynomial-time procedure ρ that, given two accepting triples (t, c_1, s_1) and (t, c_2, s_2) with $c_1 \neq c_2$, computes w such that $(x, w) \in \mathcal{R}$.

We also require that a Σ-protocol is honest verifier perfect zero-knowledge in the particular sense that there is a simulator which, given input x and challenge c, computes a t and an s such that (t, c, s) is accepting w.r.t. x, and has a distribution equal to that of real converations with the honest verifier where c occurs as challenge[2]. Figure 1 shows an example of a Σ-protocol.

Cramer et al. [11] show that different Σ-protocols can be composed to obtain Σ-protocols for statements such as *"I know a witness to* $x_1 \in L_{\mathcal{R}_1}$ *or a witness to* $x_2 \in L_{\mathcal{R}_2}$*"* while retaining efficiency. Note that the zero-knowledge property in particular implies that V does not learn whether P knows a witness to x_1 or to x_2. The idea is to run the two Σ-protocols in parallel. The prover gets to decide which challenges he answers in the two instances, however, they must sum to a value chosen by the verifier. More general, we have the following lemma.

Lemma 1 (Composition of Σ-protocols [11]). *Given Σ-protocols for relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$, then there exist*

- *a Σ-protocol for the relation* $\mathcal{R}_{\binom{n}{1}} = \left\{ \left((x_1, \ldots, x_n), w \right) \mid \exists i \, : \, (x_i, w) \in \mathcal{R}_i \right\}$ *and*
- *a Σ-protocol for the relation*

  $$\mathcal{R}_\Gamma = \left\{ \left((x_1, \ldots, x_n), (w_1, \ldots, w_n) \right) \mid \exists S \in \Gamma \, : \forall i \in S : \, (x_i, w_i) \in \mathcal{R}_i \right\},$$

  where Γ is a monotone access-structure over $\{1, \ldots, n\}$.

The latter Σ-protocol allows P to demonstrate knowledge of witnesses to a qualified subset of the instances $x_1, \ldots, x_n$ without revealing which subset is involved. The construction of the protocol employs a secret sharing scheme for the access-structure Γ^* which is the structure dual to Γ. Such a scheme always exists since Γ is a monotone access-structure. Furthermore, if Γ is a threshold access-structure then so is Γ^* in which case there exist efficient secret sharing schemes for Γ^*, e.g., the one by Shamir [28]. We refer to [11] for further details.

The probably best known special case of relations $\mathcal{R}$ with Σ-protocols are public-key identification schemes such as the ones by Feige, Fiat, and Shamir [16], by Guillou and Quisquater [22], or by Schnorr [27]. A protocol that is of interest in the context of this paper is a proof of knowledge of a pre-image under a group homomorphism (see Figure 1). It turns out that this protocol is very useful in practice

[2] We require perfect zero-knowledge only for simplicity. Computational zero-knowledge could be handled too, however, known examples of Σ-protocols are typically perfect zero-knowledge

because demonstrating (in zero-knowledge) that one knows a signature on given message reduces to demonstrating that one knows a pre-image under a group homomorphism. This is true for any of the standard signature schemes in use today (RSA, DSA, etc.) (see [1, 19]).

2.2 Probabilistic Encryption Schemes

A triple (G, E, D) of probabilistic polynomial-time algorithms is a polynomial secure public key encryption system (see for instance [21, 25]) if we have the following:

1. For every output $(E, D) \in G(1^k)$ and all messages $m \in \{0, 1\}^k$ we have $D(E(m)) = m$.

2. For all probabilistic algorithms T and M, all polynomials $p(\cdot)$, and all sufficiently large k we have

$$\begin{aligned} Pr[T(1^k, E, m_0, m_1, \alpha) = m : (E, D) := G(1^k); (m_0, m_1) := M(E, 1^k);\\ m \in_R \{m_0, m_1\}; \alpha := E(m)] < \frac{1}{2} + \frac{1}{p(k)} \ . \end{aligned}$$

That is, the adversary gets to see the public key, selects messages m_0 and m_1 in any way he wants, and gets an encryption of either m_0 or m_1. The system is secure if he cannot guess which is the case with probability essentially better than $1/2$.

For convenience, E denotes the public key as well as the actual encryption algorithm, and D the secret key as well as the decryption algorithm. Furthermore, as we often need to make the random choices in the encryption algorithm explicit, we will write sometimes $E(r, m)$ rather than $E(m)$, where r contains all coin-flips to be made for encryption; hence E will in these cases denote a "deterministic" algorithm.

3 Verifiable Encryption

3.1 Definition of Verifiable Encryption

We give a definition of a secure verifiable encryption scheme for a relation $\mathcal{R}$ following [1].

Definition 1 (Secure Verifiable Encryption). *Let $\mathcal{R}$ be a binary relation and let $L_{\mathcal{R}} = \{x | \exists w : (x, w) \in \mathcal{R}\}$. A* secure verifiable encryption *scheme for a relation $\mathcal{R}$ consists of a two party protocol (P, V) and a recovery algorithm R. We let $V_P(E, x, k)$ denote the output of V when interacting with P on input E, x, k, where k is a security parameter. We require that the following three properties hold:*

Completeness: *If P and V are honest then $V_P(E, x, k) \neq \perp$ for all (E, D) and for all $x \in L_{\mathcal{R}}$.*

Validity: *For all polynomial time $\tilde{P}$, for all $(E, D) \in \mathcal{G}(1^k)$, for every positive polynomials $p(\cdot)$, and all sufficiently large k we have*

$$Pr\big[(x, R(D, \alpha)) \notin \mathcal{R} \text{ and } \alpha \neq \perp \ : \ \alpha := V_{\tilde{P}}(E, x, k)\big] < \frac{1}{p(k)} \ .$$

Computational Zero-Knowledge: *For every $\tilde{V}$ there exists a expected polynomial-time simulator $S_{\tilde{V}}$ with black-box access to $\tilde{V}$ such that for all distinguishers A, all polynomials p, all $x \in L_{\mathcal{R}}$, and all sufficiently large k, we have*

$$Pr\big[A(E, x, \alpha_i) = i \;:\; (E, D) := G(1^k); \alpha_0 := S_{\tilde{V}}(E, x, k); \alpha_1 := \tilde{V}_P(E, x, k); i \in_R \{0, 1\}\big] < \frac{1}{2} + \frac{1}{p(k)}\,.$$

The completeness property should need no discussion. Validity ensures that it almost never happens that an honest verifier accepts simultaneously with recovery failing to compute a witness for $x \in L_{\mathcal{R}}$. A consequence of this is that a verifiable encryption scheme is necessarily also a proof of knowledge of such a witness. Finally, the zero-knowledge condition assures the prover that no verifier can learn anything beyond the fact that $x \in L_{\mathcal{R}}$ and that a witness for x can be recovered from the output.

Our definition of zero-knowledge differs conceptually from the one given in [1]. In particular, our definition requires that a verifiable encryption scheme is zero-knowledge for every instances in $L_{\mathcal{R}}$ whereas the definition given in [1] only requires that the protocol is zero-knowledge for instances that the adversary generates, or is able to generate.

The verifiable encryption scheme for group-homomorphisms described in [1] can be seen as a special case of our definition with respect to the relations $\mathcal{R}$ that are considered. In particular, the relation defined by $\{(x, w) | x = f(w)\}$, where f is a group homomorphism, is a subclass of the relations having a Σ-protocol.

We note that computationally zero-knowledge is the best we can achieve due to the requirement that a witness is (and should be) recoverable from the conversation. Also note that the encryption scheme must be semantically secure (cf. Section 2.2) in order for the zero-knowledge property to be satisfied.

Our zero-knowledge definition allows the simulator to rewind the verifier. In some applications it may be desirable to require that simulation can be done without rewinding, since this implies that the protocol is *concurrent* zero-knowledge [14], i.e., even an arbitrary interleaving of different instances of the protocol is simulatable. We note that one variant of our protocol in fact has this stronger property.

3.2 A Verifiable Encryption Scheme

We first present a very simple but not very efficient scheme, and then move on with improvements.

Let (G, E, D) be a semantically secure crypto-system. Let $(E, D) := G(1^k)$ be the public and secret key of a third party. Also we are given a relation with a Σ-protocol defined by procedures σ_t, σ_s, and σ_v. Assume for simplicity that the verifier can choose only between 0 and 1 as challenges. The idea is now simply that given x, the prover will start a conversation in the Σ-protocol. Using his knowledge of a witness w he can compute answers to both $c = 0$ and $c = 1$, and he supplies encryptions under E of both of these. The verifier can now ask P to open one of these encryptions to check if it contains a valid answer. If this is true for both encryptions, decrypting the other one allows to recover w (due to the properties of the Σ-protocol), whereas if at least one of them contains garbage, the prover will be caught with probability $1/2$. Concretely, we have:

$P(x, w, E)$		$V(x, E)$
$r_0, r_1 \in_R \{0,1\}^\ell$, $(t, r_t) := \sigma_t(x, w)$		
$s_0 := \sigma_s(x, w, r_t, 1)$, $s_1 := \sigma_s(x, w, r_t, 0)$		
$e_0 := E(r_0, s_0)$, $e_1 := E(r_1, s_1)$	$\xrightarrow{t, e_0, e_1}$	
	$\xleftarrow{c}$	$c \in_R \{0,1\}$
$s := s_c$, $r := r_c$	$\xrightarrow{s, r}$	
		$o := \begin{cases} (c, e_{\bar{c}}, s, t) & 1 = \sigma_v(x, t, c, s),\ e_c = E(r, s) \\ \perp & \text{otherwise} \end{cases}$
		(o)

Figure 2: One round of the basic verifiable encryption scheme. Recovery takes place by decrypting $e_{\bar{c}}$ and using soundness of the Σ-protocol to compute a valid w.

Theorem 2. *Let $\mathcal{R}$ be a relation that has a Σ-protocol. The protocol depicted in Figure 2 is a secure verifiable encryption scheme for $\mathcal{R}$ when sequentially repeated k times.*

Proof. Completeness: This can easily be verified.

Validity: The algorithm R will fail, only if in every iteration at least one of e_0, e_1 contains invalid protocol responses. This means that V will accept with probability at most $1/2$ in each iteration, and so it accepts all iterations with probability at most 2^{-k}.

Computational Zero-Knowledge: We can build a simulator for one iteration using standard resetting techniques: we first choose a random bit c', our guess at what V's challenge will be. Then we simulate a conversation (t, c', s) in the Σ-protocol. We encrypt s to get $e_{c'}$, and encrypt something random to get $e_{1-c'}$. Then we send t, e_0, e_1 to $\tilde{V}$, and get c back. If $c' = c$, we open e_c as the honest prover would do. Otherwise, we rewind and try again. By honest verifier zero-knowledge of the Σ-protocol and semantic security of the encryption, the probability that $c' \neq c$ is at most $1/2 + 1/p(k)$ for some polynomial p and the output is computationally indistinguishable from a real conversation. It also follows that the expected run-time of the simulator is $O(kT(k))$, where $T(k)$ is the time needed for one attempt to simulate one iteration. ☐

The main drawback of our basic scheme is that the verifier must store an encryption and a conversation in the Σ-protocol for *each* repetition and that it needs to be repeated $\Theta(k)$ times sequentially. In Figure 3 an improved scheme is depicted, that allows to store much less encryptions and triples. The idea is that in the basic step, the prover will supply a valid triple where the challenge is 0, but where the prover's response is encrypted. The verifier can then ask the prover either to open the encryption or to supply a valid answer to challenge 1. The point is that the verifier only needs to remember the unopened encryptions and related triples.

Furthermore, the protocol is made constant round by relying on a commitment scheme, i.e., a function *Commit* that takes as input the string α to commit to and an additional input string β. Then a player can commit by sending $T := \mathit{Commit}(\alpha, \beta)$, where β is randomly chosen (we write just $\mathit{Commit}(\alpha)$ in the following). We require that the distributions of commitments to different α's are computationally indistinguishable.

On the other hand it should be computationally hard to open a commitment in two different ways, i.e., to find $\alpha \neq \alpha'$ and β, β' such that $\mathit{Commit}(\alpha, \beta) = \mathit{Commit}(\alpha', \beta')$. There are numerous efficient constructions known of such schemes (see for instance [10]), but in fact our assumptions in this scenario are already sufficient to ensure their existence: $\mathcal{R}$ must be a hard relation, in order for our scenario to make sense (there would be no point in P encrypting w while trying to keep it secret from V, if V can easily find w from the public x) and it is known that a Σ-protocol for a hard relation implies the existence of a secure commitment scheme [12].

Commitment schemes generally require a once-and-for-all set-up phase where the function for computing commitments is chosen and verified. For simplicity, we do not include this explicitly in the protocol description, nevertheless the set-up phase does need to be considered in the proof of security.

We will require our commitment scheme to be *trapdoor*, that is, there is a piece of trapdoor information, knowledge of which allows opening a commitment in an arbitrary way. The set-up phase will then have the form that the verifier generates the *Commit*-function together with the trapdoor information, sends the function to the prover, and proves knowledge of the trapdoor. The commitments implied by Σ-protocols allow all this to be done efficiently.

Theorem 3. *Let $\mathcal{R}$ be a relation possessing a Σ-protocol. The protocol depicted in Figure 3, when using a secure commitment scheme, is a secure verifiable encryption scheme for $\mathcal{R}$ for any u such that $\log k < u < k/2$.*

Proof. Completeness: This can easily be verified.

Validity: Consider an arbitrary prover P^*. Assume P^* convinces V about an incorrect encryption with probability larger than a polynomial fraction $1/p(k)$ for infinitely many k (and some x). We then show that P^* can be used to break the commitment scheme with non-negligible probability, contradicting the assumption.

Fix a k and x such that P^* cheats V with probability larger than $1/p(k)$. Fix a random set of coin-flips R for P^*. Say R is good if, when using R as random input, P^* cheats V with probability at least $1/2p(k)$. Then the probability that R is good is also at least $1/2p(k)$. So assume in the following that we have a good R. Since all inputs to P^* are now fixed, any conversation starts with a fixed commitment T.

The number of different subsets I that can be chosen by V is $\binom{k}{u}$. Using Sterlings bounds on $k!$,$u!$, and $(k-u)!$, we have

$$\binom{k}{u} = \frac{k!}{u!(k-u)!} > \frac{1}{q(k)}(\beta - 1)^{k/\beta}\left(\frac{\beta}{\beta - 1}\right)^{k},$$

where $q(k)$ is some polynomial and we have set $u = k/\beta$. For $\log k < u < k/2$ this is clearly super-polynomial. It follows that we can by rewinding P^* and sending it random values of C efficiently find conversations $(T, C, v), (T, C', v')$, where $C \neq C'$, and where P^* has cheated V. If v and v' involve the same opening of T, then the same set of triples from the underlying Σ-protocol are involved in both cases. Then since $C \neq C'$ there is an index i such that v contains t_i, e_i, s_i, where $(t_i, 1, s_i)$ is an

$$
\begin{array}{lcr}
P(x, w, E) & & V(x, E) \\
\hline
\text{for } i = 1, \dots, k: & & \\
\quad (t_i, \tilde{r}_i) := \sigma_t(x, w),\ \tilde{s}_i := \sigma_s(x, w, t_i, 0, \tilde{r}_i) & & \\
\quad r_i \in_R \{0,1\}^{|x|},\ e_i := E(r_i, \tilde{s}_i) & & \\
T := \mathit{Commit}((t_1, e_1), \dots, (t_k, e_k)) & \xrightarrow{\quad T \quad} & \\
 & & C \in_R 2^{\{1,\dots,k\}}\ ;\ |C| = u \\
 & \xleftarrow{\quad C \quad} & \\
 & \xrightarrow{\text{open } T} & \\
s_i := \begin{cases} \sigma_s(x, w, \tilde{r}_i, 1) & i \in C \\ \tilde{s}_i & i \notin C \end{cases} & \xrightarrow{s_1, \dots, s_k, \{r_i | i \notin C\}} & \\
\multicolumn{3}{r}{o := \begin{cases} \{e_i, s_i, t_i | i \in C\} & \begin{cases} 1 = \sigma_v(x, t_i, 1, s_i) & i \in C \\ 1 = \sigma_v(x, t_i, 0, s_i),\ e_i = E(r_i, s_i) & i \notin C \end{cases} \\ \perp & \text{otherwise} \end{cases}} \\
\hline
 & & (o)
\end{array}
$$

Figure 3: An improved verifiable encryption scheme using a commitment scheme. Reconstruction is straightforward by decrypting the e_i's in the output. The integers k and u are security parameters.

accepting triple, and where ν' contains s_i', the decryption of e_i, where $(t_i, 0, s_i')$ is also accepting. But by soundness of the Σ-protocol this implies that the recovery algorithm does manage to find a correct w, contradicting the fact that P^* cheats V. Thus it must be the case that opening of T contained in ν is different from the one occurring in ν', and we have broken the commitment scheme.

Computational Zero-Knowledge: Note that if we know the subset C that $\tilde{V}$ will choose, it is easy to simulate: for all $i \notin C$, that is, where we have to open the encryption e_i, we prepare an accepting triple $(t_i, 0, s_i)$ and encrypt s_i to get e_i. For the other i's, we prepare an accepting triple $(t_i, 1, s_i)$ and encrypt something random in e_i. This will clearly allow us to satisfy the verifier's requests, and the only difference between this and a real conversation is in the contents of the unopened encryptions.

Hence the strategy of a simulator is to use the proof of knowledge of the trapdoor that V^* gives in the set-up phase of the commitment scheme. The simulator attempts to extract the trapdoor by rewinding V^* in this phase, and in parallel (for technical reasons) runs an exhaustive search for the trapdoor. This ensures that the simulator will in expected polynomial time extract the trapdoor. Then any commitment can be opened to reveal any value desired, and the rest of the simulation is trivial, by the above. □

Remark. There is another variant of our protocol which requires 4 messages, but has the advantage of being secure against an unbounded prover. Here, V commits to his choice of C in advance, P sends $(s_1, r_1), ..., (s_k, r_k)$, V opens the commitment, and P responds to the I revealed. This can be proven zero-knowledge using techniques from [20].

Remark. Given a collision intractable hash function h, one can sometimes make the protocol more efficient by committing to the (shorter) value $h((s_1, r_1), ..., (s_k, r_k))$ in the first step.

Remark. Note that our simulator in the proof of the zero-knowledge property uses rewinding, but only in the set-up phase for the commitment scheme. Since this phase can be done once and for all in a preprocessing, it is feasible to ensure that this one phase is not interleaved with other protocols. This will imply that the entire protocol is concurrent zero-knowledge (see also [15]).

Remark. In practice one is often interested in a particular error-probability ϵ. There will then be many values k, u with $1/\binom{k}{u} \leq \epsilon$, and one should then of course choose whatever particular k and u that fit the application best.

It is easy to make a non-interactive variant of this using the Fiat-Shamir heuristic: the prover computes $(t_1, e_1), \ldots, (t_k, e_k)$, determines the challenge C from $h((t_1, e_1), \ldots, (t_k, e_k))$, where h is some suitable (hash) function, and finally appends valid responses $(s_1, r_1), \ldots, (s_k, r_k)$. All this can be verified as before by V. It is straightforward to show that this is secure in the random oracle model, replacing calls to h by calls to the oracle.

4 Verifiable Group Encryption

In our basic primitive, the prover and the verifier have to trust the third party to behave as expected. This is, the prover must trust him/her only to recover the encrypted witness when appropriate while the verifier relies on the third party to decrypt the witness when required. To achieve higher security against fraudulent third parties, we extend our basic primitive to verifiable *group* encryption, which on its own is a useful concept in many cases. Here, the witness gets encrypted for n third parties, called *proxies*, such that only designated subsets of them can jointly recover the witness. Although it superficially looks more complicated, it turns out to be trivial to implement using our basic verifiable encryption scheme, as we shall see.

Informally, verifiable group encryption takes place in a similar model as ordinary verifiable encryption, that is P and V interact on common input x, where P knows w such that $(x, w) \in \mathcal{R}$. As before, it is instructive to think of w as being a signature on x w.r.t. some fixed public key. Now, however, n public encryption keys $E_1, \ldots, E_n$ are involved, and a monotone access structure Γ over $\{1, \ldots, n\}$ is agreed by P and V. Then an honest V obtains from P encryptions $E_1(w_1), \ldots, E_n(w_n)$ such that a valid w can be reconstructed from a subset A of the w_i's, if $A \in \Gamma$, whereas a set $A \notin \Gamma$ gives no information. Finally, if honest proxies forming a set $A \in \Gamma$ decide to reconstruct w, they can do so successfully, even if dishonest proxies also participate.

This notion of "group encryption" should not be confused with the notion of threshold encryption [13]. In the latter case, a number of parties publishes *single* public key and the access structure is determined by these parties during the setup of the system.

$$
\begin{array}{lcr}
P(u, m, y, s', E) & & V(u, m, y, E) \\
\hline
r \in_R \mathbb{Z}_q,\ t := y^r & & \\
s_0 := r,\ s_1 := r + s \pmod q & & \\
s_{00}, s_{10} \in_R \mathbb{Z}_q & & \\
s_{01} := s_0 - s_{00} \pmod q,\ s_{11} := s_1 - s_{10} \pmod q & & \\
r_{00}, r_{01}, r_{10}, r_{11} \in_R \{0,1\}^{\ell} & & \\
e_{00} := E_0(r_{00}, s_{00}),\ e_{10} := E_0(r_{10}, s_{10}) & & \\
e_{01} := E_1(r_{01}, s_{01}),\ e_{11} := E_1(r_{11}, s_{11}) & & \\
& \xrightarrow{t, e_{00}, e_{10}, e_{01}, e_{11}} & \\
& \xleftarrow{c} & c \in_R \{0,1\} \\
\tilde{s}_0 := s_{c0},\ \tilde{r}_0 := r_{c0} & & \\
\tilde{s}_1 := s_{c1},\ \tilde{r}_1 := r_{c1} & \xrightarrow{\tilde{s}_0, \tilde{s}_1, \tilde{r}_0, \tilde{r}_1} & \\
& & s := \tilde{s}_0 + \tilde{s}_1 \pmod q \\
& & o := \begin{cases} (c, e_{0\bar{c}}, e_{1\bar{c}}, s, t) & \begin{cases} t = y^c (u g^{h(u\|m)})^s \\ e_{0c} = E_0(\tilde{r}_0, \tilde{s}_0), e_{1c} = E_1(\tilde{r}_1, \tilde{s}_1) \end{cases} \\ \perp & \text{otherwise} \end{cases} \\
\hline
& & (o)
\end{array}
$$

Figure 4: An example of verifiable group encryption for $\Gamma = \{\{0,1\}\}$. For simplicity only one round is described (cf. previous section). Schnorr signature on m is defined to be (c', s') such that $c' = h(u\|m)$ holds, where $u = g^{c'} y^{s'}$ and y is the public key of signer. The signature can be verifiable encrypted by providing u and m and using the relation $\{(u/g^{h(u\|m)}, w) | w = \log_y(u/g^{h(u\|m)}) \pmod q\}$. Recovery takes place by computing $(-1)^c(D_0(e_{0\bar{c}}) + D_1(e_{1\bar{c}}) - s) \pmod q$.

4.1 Realization

A verifiable group encryption scheme can be realized using a secret sharing scheme for the chosen access structure Γ. We just need to observe that our verifiable encryption scheme from before works based on *any* public key encryption scheme. In particular, we will execute it using the following encryption scheme: given input s, use the given secret sharing scheme to get shares $s_1, \ldots, s_n$, and then let the output ciphertext be $E_1(s_1), \ldots, E_n(s_n)$. Clearly, you can compute s from a correctly formed ciphertext, if you can decrypt a subset of the s_i's corresponding to a set in Γ.

From the construction it is clear that, due to the properties of secret sharing schemes, the proxies can reconstruct the witness when given $E_1(s_1), \ldots, E_n(s_n)$ and Γ by decryption and pooling the shares. This is possible even with the participation of malicious proxies as long as the honest proxies form a set $A \in \Gamma$, however, they might not be able to do this efficiently in the presence of malicious proxies who provide incorrect values for the s_i's. If the encryption scheme used directly allows proxy i to prove that s_i is indeed the value encrypted in $E_i(s_i)$, the problem is trivial to solve. If this is not the case, we can modify the encryption scheme and encrypt the random choices used for encryption of the shares as well for the respective proxies, i.e., our encryption scheme would output $\big(E_1(r_1, s_1), E_1(\tilde{r}_1, r_1)\big), \ldots, \big(E_n(r_n, s_n), E_n(\tilde{r}_n, r_n)\big)$. Now, proxy i can prove correct decryption by providing r_i and s_i.

Remark. In case the scheme is made non-interactive using the Fiat-Shamir heuristic [17], the access structure Γ should also be included in the hash-function.

4.2 Verifiable Group Encryption Versus Verifiable Signature Sharing

The concept of *verifiable signature sharing* (VΣS) [6, 19] involves a *signature receiver* who distributes shares of a signature on a public message to a set of proxies such that all proxies can verify that this has been done correctly, and a qualified set of proxies can always reconstruct the signature, even in presence of malicious proxies. Trivially, a verifiable group encryption scheme can be used to implement VΣS: we simply execute verifiable group encryption of a signature on the given message, such that the signature receiver plays the role of the prover, and the proxies together play the role of the verifier. If the interactive variant is used, this is done by having proxies generate challenges for the prover by collective coin-flipping. With the non-interactive variant, no special precautions are needed, and the scheme becomes even publicly verifiable. The possibility to use any encryption scheme for the individual proxies solves an open problem raised in [6]. Moreover, a (publicly) verifiable secret sharing scheme, e.g., see [30], can be obtained along the same lines.

5 Application to Group Signatures and Identity Escrow

A *group signature scheme* [2, 4, 5, 7, 8, 26] allows a member of a group of users to sign a message on the group's behalf. The scheme protects the privacy of signers in that the verifier should not be able to find the identity of the signer. However, to handle special cases where the scheme is misused by some user, there is a *revocation manager* who can indeed find the identity of the signer from the signature. In some schemes, there is also a *group manager* who takes care of the key set-up and enrollment of users in the group. No collusion of users (even including the group manager) should be able to forge signatures such that it is not possible for the revocation manager to reveal the identity of the signer. Furthermore, no collusion of users (even including both managers) should be able to forge signatures such that upon revocation they seem to originate from another user. Moreover, we want to minimize the involvement of parties in the protocols. This means that managers should not be involved in creating a signature, and ideally also that the revocation manager is not involved in establishing the group, and is in fact completely inactive until revocation of anonymity is needed.

The interactive equivalent of group signatures are group identification scheme with revocable anonymity (also called identity escrow [23]). Here, the goal is for a group member to anonymously identify himself as a member - rather than being able to sign a message. Except for this, the security properties of group signatures carry over directly.

For both kind of schemes, it is of course desirable that they can be implemented based on keys that users and managers already have established, even if those keys were not indented to be used in group identification. This property is called *separability*[3].

In the following, we show how to use a verifiable encryption scheme to design a separable group identification scheme with revocable anonymity. This scheme can be proved secure, assuming only security of the encryption, signature, and identification schemes involved. We then modify this to a (non-interactive) group signature scheme using the Fiat-Shamir heuristic [17]. This scheme is secure assuming in addition that the heuristic is valid for the protocol and hash function involved. It can be proved secure with no additional assumptions in the random oracle model. For a formal model of group signatures and identity escrow we refer to [5, 23].

5.1 Realizations

We describe the basic idea for the case where the users' public keys are of some signature schemes and then extend the scheme such that the users' public keys can also be of some interactive identification schemes.

We assume we are given public keys $P_1, \ldots, P_n$ of secure signature schemes for n players, and that for each signature scheme employed, there is a Σ-protocol for the relation $\{(x, w)|\ w \textit{ is a valid signature on message } x\}$. As mentioned earlier, this is true of any signature scheme for which reduction functions to a group-homomorphism exists [1]. To prove our scheme's security formally, we need that the signature schemes are secure against chosen message attacks. Finally, we assume that a revocation manager has been selected, who has published a public key E to a semantically secure encryption scheme. Now, by Lemma 1 and Theorem 3 we get a group identification scheme with revocable anonymity, as follows:

The group's public key consists just of the group manager's signature (certificate) on the tuple $(P_1, \ldots, P_n)$. To identify as a group member, the prover computes the signature on message x (randomly chosen by the verifier) with respect to his/her public key. Then the prover and the verifier carry out the verifiable encryption protocol for the relation

$$\{(x, w)|w \textit{ is a valid signature on } x \textit{ with respect to one of the public keys } P_1, \ldots, P_n\} ,$$

where encryption public key used is E, the one of the revocation manager. We can do this because, by Lemma 1, an efficient Σ-protocol for this relation can be derived from the Σ-protocols we assumed exist for each single signature scheme.

This derived Σ-protocol proves that w is valid w.r.t. one of the public keys, but yields no information about which one is involved. This, and the zero-knowledge property of the verifiable encryption implies anonymity for the prover. Furthermore, if some coalition of users could impersonate another user, then they could also forge this user's signatures. Finally, the anonymity can be revoked just by decryption, and will yield correct results by validity of the verifiable encryption. With respect to the revocation manager's ability to prove correct decryption of the witness the remarks of the previous section applies. That is, either the underlying encryption scheme allows this or, if not, the verifiable encryption scheme is modified as described above.

[3]The term originates from [23]. However, we use it in a stronger sense, that is, in [23] it only denotes the fact that the revocation manager is able to choose his/her public key independently.

Clearly, applying the Fiat-Shamir heuristic [17] yields a group *signature* scheme from this construction: the message x which was chosen by the verifier before, will now be the message to be signed. And we hash x and the prover's first message in the verifiable encryption protocol to get a challenge. This can be proved secure in the random oracle model.

In summary, we have argued:

Theorem 4. *Given any secure signature scheme with an associated Σ-protocol, and any secure public key encryption scheme. Then there exists a secure separable group identification scheme with revocable anonymity, and a separable group signature scheme secure in the random oracle model. The complexities of the schemes are linear in the group's size, the security parameter, and in the complexity of the signature and encryption schemes.*

What if we are not given a number of signature public keys, but in stead public keys to an interactive identification protocol, which we assume to be Σ-protocol (such as Fiat-Feige-Shamir or Schnorr)? Can we still build a group identification scheme? We could try using directly the identification protocol together with the 1-out-of-n method in the verifiable encryption scheme. But this would allow the revocation manager to compute the secret key of the prover and hence the scheme could be used only once.

In the following we provide a method to transform the Σ-protocol of any identification scheme such that we can nevertheless build a group identification scheme from it. The idea is to only prove the knowledge of the response in an instance of the identification protocol rather than providing it. Let $\mathcal{R}_i$ be the relation of some identification scheme with the Σ-protocol $(\sigma_t, \sigma_s, \sigma_v)$ and let x be the instance of $\mathcal{R}_i$ consistent with the public key P_i. Define the relation $\tilde{\mathcal{R}}_i = \{(s, (x, t, c)) | 1 = \sigma_v(x, t, c, s), x \in L_{\mathcal{R}_i}\}$. In other words, t and c were the first two messages in the identification protocol and the third message is a witness to $\tilde{x} = (x, t, c)$ for $\tilde{\mathcal{R}}_i$. Thus, if the prover and the verifier construct $\tilde{x}$ together as the first two step of the identification protocol, and then the prover proves the knowledge of a witness to $\tilde{x}$ for $\tilde{\mathcal{R}}_i$, then the verifier will be convinced, that the prover also knows a witness to x for $\mathcal{R}_i$. For most known identification protocol there exists an efficient Σ-protocols for this latter relation. Figure 5 shows as an example such a transformation of the Schnorr identification protocol.

This transformation allows us now to construct a group identification scheme with revocable anonymity as follows. The prover and the verifier carry out the first two steps of the identification protocols for every public key P_i to get instances for the new relations $\tilde{\mathcal{R}}_i$ and then carry out the Σ-protocol for proving knowledge of a witness to (at least) one of these new instances (using Lemma 1). The rest of the construction is the same as in case we start from signature schemes. It is clear that we can also mix identification schemes and signature schemes.

5.2 Extensions and Related Work

Extensions to generalized group identification schemes with revocable anonymity or generalized group signature schemes [2] are straight forward. In this case the prover is actually a set of group members who demonstrate that they form a qualified set w.r.t some access structure, but without revealing which set is present. Thus they

$$
\begin{array}{lcr}
P(g, z = g^y, y) & & V(g, z) \\
\hline
r \in_R \mathbb{Z}_q^*,\ t = g^r & \xrightarrow{\quad t \quad} & \\
 & \xleftarrow{\quad c \quad} & c \in_R \{0,1\}^k \\
\tilde{z} := t/z^c & & \tilde{z} := t/z^c \\
\cdots\cdots\cdots & \cdots\cdots\cdots & \cdots\cdots\cdots \\
\tilde{r} \in_R \mathbb{Z}_q^*,\ \tilde{t} = g^{\tilde{r}} & \xrightarrow{\quad \tilde{t} \quad} & \\
 & \xleftarrow{\quad \tilde{c} \quad} & \tilde{c} \in \{0,1\}^k \\
\tilde{s} := \tilde{r} + \tilde{c}(r + yc) \pmod q & \xrightarrow{\quad \tilde{s} \quad} & o := \begin{cases} (\tilde{t}, \tilde{c}, \tilde{s}) & \tilde{t} = \tilde{z}^{\tilde{c}} g^{\tilde{s}} \\ \perp & \text{otherwise} \end{cases} \\
\hline
 & & (o)
\end{array}
$$

Figure 5: Transforming Schnorr's identification protocol. The prover's public key is $(z, G = \langle g \rangle)$ where $|G| = q$ and $y = \log_g z$ is the prover's secret key. In the protocol steps above the dotted line, the prover and the verifier jointly construct the instance $(\tilde{z}, G = \langle g \rangle)$ of the new relation. The protocol steps below the dotted line constitute the well-known protocol for proving the knowledge of $\tilde{z}$, i.e., the witness in the new relation.

have to do joint computations and some precautions must be taken. We refer to [2] for details.

To protect group members from fraudulent behavior of the revocation manager, the ability of revocation can be distributed to several parties by using our verifiable group encryption instead of ordinary verifiable encryption (cf. Section 4).

Comparing our group signature schemes with previous proposals (none of which provide separability) our schemes are clearly less efficient – that's the price we have to pay for complete separability.

The schemes proposed in [4, 5, 23] have the property that the group's public key does not depend on the size of the group. Our verifiable encryption scheme can be used as a subroutine in these scheme allowing the revocation manager to independently choose *any* encryption scheme. It remains an open problem to find efficient schemes providing complete separability where the group's public key does not depend on the size of the group.

6 Miscellaneous Applications

6.1 How to Hide Your Trustees

The goal here is that there are n trustees and the prover encrypts the witness for *one* of them but not for the others. However, the verifier must not learn which particular trustee is able to decrypt. This can be an advantage if the application requires that the witness be reconstructible by access to only one trustee, but you still want some protection against verifiers trying to coerces the trustee(s) into revealing the secret.

This can be solved by simply observing that our verifiable encryption scheme is itself a Σ-protocol, and hence applying Lemma 1 directly gives a protocol for this problem.

This generalizes easily to any access structure, i.e., the verifier gets convinced that some qualified subset of trustees can each *individually* decrypt the witness, but he does not learn which subset. Note that this is not the same as verifiable group encryption, where any qualified subset can decrypt *collectively*.

6.2 Blind Signatures With Revokable Anonymity

Another application that uses verifiable encryption are blind signature schemes and e-cash systems that provide revokable anonymity (e.g., [3, 18, 31]). Here we have four parties, a customer, a bank, a shop, and a trustee. The two basic protocols in these schemes are a withdrawal protocol between the bank and the customer and a payment protocol between the shop and the customer. During both protocols, the customer is required to provide some string and to prove that this string is an encryption under the trustee's public key of some information that will allow to link the two protocols. For these tasks, our verifiable encryption schemes can be used as a subroutine in these protocols hence allowing the trustee to independently choose *any* encryption scheme.

Acknowledgments

The authors are grateful to Christian Cachin for his comments on an earlier version of this paper.

References

[1] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In K. Nyberg, editor, *Advances in Cryptology — EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer Verlag, 1998.

[2] J. Camenisch. Efficient and generalized group signatures. In W. Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 465–479. Springer Verlag, 1997.

[3] J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. In E. Bertino, H. Kurth, G. Martella, and E. Montolivo, editors, *Computer Security — ESORICS 96*, volume 1146 of *Lecture Notes in Computer Science*, pages 33–43. Springer Verlag, 1996.

[4] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In K. Otha and D. Pei, editors, *Advances in Cryptology — ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 160–174. Springer Verlag, 1998.

[5] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 410–424. Springer Verlag, 1997.

[6] D. Catalano and R. Gennaro. New efficient and secure protocols for verifiable signature sharing and other applications. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *Lecture Notes in Computer Science*, pages 105–120, Berlin, 1998. Springer Verlag.

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

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

[9] R. Cramer. *Modular Design of Secure yet Practical Cryptographic Protocol*. PhD thesis, University of Amsterdam, 1997.

[10] R. Cramer and I. Damgård. Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free? In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *Lecture Notes in Computer Science*, pages 424–441, Berlin, 1998. Springer Verlag.

[11] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Y. G. Desmedt, editor, *Advances in Cryptology — CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer Verlag, 1994.

[12] I. B. Damgård. On the existance of bit commitment schemes and zero-knowledge proofs. In G. Brassard, editor, *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 17–27, 1990.

[13] Y. Desmedt and Y. Frankel. Threshold cryptography. In *Advances in Cryptology — CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer-Verlag, 1990.

[14] C. Dwork, M. Naor, and A. Sahai. Concurrent zero knowledge. In *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*, 1998.

[15] C. Dwork and A. Sahai. Concurrrent zero-knowledge: Reducing the need for timing constraints. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *Lecture Notes in Computer Science*, pages 105–120, Berlin, 1998. Springer Verlag.

[16] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1:77–94, 1988.

[17] A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology — CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer Verlag, 1987.

[18] Y. Frankel, Y. Tsiounis, and M. Yung. "Indirect discourse proofs:" Achieving efficient fair off-line e-cash. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology — ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 286–300. Springer Verlag, 1996.

[19] M. Franklin and M. Reiter. Verifiable signature sharing. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 50–63. Springer Verlag, 1995.

[20] O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[21] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, Apr. 1984.

[22] L. C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In C. G. Günther, editor, *Advances in Cryptology — EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer Verlag, 1988.

[23] J. Kilian and E. Petrank. Identity escrow. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume 1642 of *Lecture Notes in Computer Science*, pages 169–185, Berlin, 1998. Springer Verlag.

[24] S. Micali. Efficient certificate revocation and certified e-mail with transparent post offices. Presentation at the 1997 RSA Security Conference.

[25] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, 17(2):412–426, April 1988.

[26] H. Petersen. How to convert any digital signature scheme into a group signature scheme. In M. Lomas and S. Vaudenay, editors, *Security Protocols Workshop*, Paris, 1997.

[27] C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):239–252, 1991.

[28] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, Nov. 1979.

[29] M. Stadler. *Cryptographic Protocols for Revocable Privacy*. PhD thesis, ETH Zürich, 1996. Diss. ETH No. 11651.

[30] M. Stadler. Publicly verifiable secret sharing. In U. Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 191–199. Springer Verlag, 1996.

[31] M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind signatures. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology — EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 209–219. Springer Verlag, 1995.