Certificates of Recoverability with Scalable Recovery Agent Security

Eric R. Verheul

PricewaterhouseCoopers, GRMS Crypto Group, P.O. Box 85096, 3508 AB Utrecht
Eric.Verheul@[nl.pwcglobal.com, pobox.com]

Abstract. We propose new schemes for Certificates of Recoverability (CRs). These consist of a user's public key and attributes, its private key encrypted in such a way that it is recoverable by one or more Key Recovery Agents (KRAs), plus a publicly verifiable proof of this (the CR). In the original schemes, the level of cryptographic security employed by the KRA and the users is necessarily the same. In our schemes the level of cryptographic security employed by the KRA can be set higher, in a scalable fashion, than that being employed by the users. Among the other improvements of our schemes are its applicability to create CRs for cryptosystems based on the Discrete Log problem in small subgroups, most notably the Digital Signature Standard and Elliptic Curve Crypto systems. Also, the size of the constructed proofs of knowledge can be taken smaller than in the original schemes. We also present several new constructions and results on the hardness of small parts, in the setting of Diffie-Hellman keys in extension fields.

1 Introduction

In a Public Key Infrastructure there are several roles, of which we name:

Participants Who rely on public key encryption and need to be confident that the associated private key is owned by the user mentioned in the public key.

Certification Authorities This confidence is obtained through the use of *public key certificates*, which bind public keys and its attributes to a user. The binding is achieved by having a trusted *Certification Authority* verifying the user's identity and other attributes and then digitally sign each certificate. This verification is usually performed by an optional entity, called *Registration Agent* "closer to the user" than the CA.

Key Recovery Agents Sometimes a user or the organization the user works for, requires an emergency back-up of its private key and deposits it with a trusted Key Recovery Agent (KRA).

It could be part of a Certification Policy of a user's organization or CA that no certificates are issued unless the associated private key is properly escrowed. In [27] the concept of *Certificate of Recoverability* (CR) is introduced which supports such a policy. Here a user offers three components to a CA (or RA):

C1 Its public key, identity and other attributes.

C2 An encrypted copy of its private key, decryptable only by one or more Key Recovery Agents.

C3 A publicly verifiable proof that the first two components are correctly formed, i.e. that the private key of the public key in component **C1** coincides with the encrypted private key in component **C2**. This component is called the Certificate of Recoverability.

After verifying the proof in **C3**, the CA issues a regular certificate on the public key. In addition, the CA archives the three components or sends them to the KRA(s). In a conventional escrow scheme, i.e. without the component **C3**, the CA (or RA) would have setup a KRA-acceptance phase. This consists of sending components **C1** and **C2** to all the KRAs to verify the validity of **C2**. The central advantage of a CR scheme compared with a conventional escrow system, is that the CA can perform this KRA-acceptance phase without having to communicate with the KRA(s). This gives rise to the many advantages of a CR scheme, like speed, cost effectiveness and security.

The publicly verifiable proof in component **C3** should preferably have the property of being perfectly zero-knowledge, or being a parallel version of it, see [26]. Also, observe that the user's private key is contained in both his public key and in component **C2**. So, preferably there should be a guarantee for *combined* security, i.e. that this combination of encryption schemes (used by users and KRAs) does not result in an exploit, yielding (parts of) the user's private key. Compare [11], where combined (in)security is discussed for the RSA cryptosystem.

In [27] a construction for the CR concept is given, which is related to a construction in [24]. Here a prime number p and a cyclic group $G = \langle g \rangle$ of order p are generated (for instance by the CA) in which the the discrete logarithm problem is intractable. The idea is that the public keys of users takes the form g^x, where $0 \leq x < p$, is the associated private key.
In addition, the ElGamal encryption scheme [8] in a multiplicative subgroup Γ of order ω in the (basic) finite field $GF(p)$ is used to encrypt a copy of the private key x in **C2**. The last component **C3** consists of a transcript of a perfect zero-knowledge proof of knowledge, proving that the private key x is encrypted in both **C1** and **C2**. Moreover, a guarantee of combined security for this construction is given in [24].

In both [24] and [27] it is suggested to combine the construction of the system parameters of users and KRAs. One starts with choosing the order ω as a large prime number, such that $p = 2\omega + 1$ is prime as well as $r = 2p + 1$. Then take G as the subgroup of order p in the multiplicative group of $GF(r)$ and Γ is the subgroup of order ω in the multiplicative group of $GF(p)$.
For sufficient security, r, p, ω should be of size ≥ 1000 bits. Also, to obtain a soundness level of 2^{-v} in the above scheme, the size of a transcript is roughly v times the number of bits in p, e.g. 100,000 bits for $v = 100$ and $|p| = 1000$.

This type construction of CRs has two major drawbacks:

1. The level of security given by the KRA's public key is the same as the security given by the public keys of the users. The KRA's private key gives access to *all* users' private keys, as it is a *master key*. Hence the KRA's public key is subject to more severe threats than the users' public keys. For instance, attackers wanting access to communication of different users, could form an alliance to break the KRA's public key. It is "best practice" that master keys, such as KRA's public keys, should be considerably more difficult to break than user's keys, i.e. public keys of the users. This is also why the signing key of a (root) Certifying Authority is typically longer than that of its users.

   We remark that using stronger KRA's public keys is only sensible, provided breaking *one* user's public key does not imply breaking *all* users' public keys. This is the case when using DL-based schemes. In the setting of a finite field, breaking one discrete logarithm based public key considerably helps in breaking other ones, but that still takes sub-exponential time (cf. [10], [26, p.172]). In the setting of Elliptic Curve Cryptosystems, breaking one public key doesn't help (much) in breaking other public keys: an exhaustive search of proportional to the square root of the group's order still has to be conducted.
2. The use of DL-based schemes in groups of relative short order (of say of size 160 bit), can be very beneficiary in terms of speed and data-storage. Such groups are employed in the Digital Signature Standard and in Elliptic Curve Crypto systems. However, the schemes from [27], forces users to use groups of orders of at least 1024 bits to have sufficient security; making it impossible to employ these beneficiary groups.

To remedy these two drawbacks, one could use the generic verifiable encryption scheme that appears in [2]. However, the resulting interactive proof in **C3** would only have the property of computational zero-knowledge. Also, no guarantee for *combined* security, as defined above, is given in [2].

Our System We propose two, related schemes for CRs which does not have the above mentioned drawbacks. In our schemes, users can use any type of cyclic group G of prime order p in which the discrete logarithm is intractable. This means that the size of p should be larger than 160 bits; but no further restrictions are made. The user's private key is encrypted for the KRA, using the ElGamal scheme in a field extension $GF(p^t)$. Here one can choose to employ the classical ElGamal scheme, i.e. using the whole multiplicative group of $GF(p^t)$, or the subgroup variant as proposed by Schnorr [23], using a multiplicative subgroup of $GF(p^t)$ of suitable, large prime order.

In both variants the proofs of knowledge, **C3**, can be formed in a perfect zero-knowledge fashion, the size of which can be taken considerably smaller than that from [27] and [24]. Moreover, by employing optimal normal bases our schemes are more efficient than comparable schemes in [27]. Finally, the guarantee for combined security is proven to be given by the hardness of the Diffie-Hellman Decision Problem.

Apart from key-escrow applications, our techniques can be applied in the design of electronic cash systems providing revocable anonymity [25] in which the identity of the system's users can be recovered when the system is abused for criminal activities. The recovery of the identity is done with the help of trustees, whose role can be compared with that of the above mentioned KRAs.

We remark that we do not incorporate the CR (i.e. **C3**) itself in the issued certificate. There is no need to do so, and in our schemes this would also introduce the possibility of the establishment of shadow public keys, see [12].

Outline of the paper Apart from presenting new schemes for Certificates of Recoverability, we present several new results and constructions in multiplicative subgroups of extension fields. In Section 2 we will first discuss new results on the hardness of small parts of the key exchanged in the Diffie-Hellman protocol in extension fields. Section 3 deals with proving knowledge on equality of logarithms in the setting of extension fields. In Section 4, we present our basic scheme, using the results of Sections 2 and 3 to prove security and functionality. In this section we also describe generalizations, like incorporating secret sharing, and efficiency improvements of the scheme.

2 DL-based cryptosystems over extension fields

2.1 Introduction

Let H be a multiplicative, cyclic group in which the discrete logarithm problem is intractable and h is a generator of H of order o. The h, H and o are system parameters given to all participants. The Diffie-Hellman (DH) key agreement [7] was the first practical solution for allowing two parties to agree over an insecure channel on a common secret key. The security of it lies in the *Diffie-Hellman problem* of computing values of the function $\mathrm{DH}(h^x, h^y) = h^{xy}$. A drawback of the basic DH scheme is that the parties involved, Alice and Bob say, can only share one secret key. This is remedied in several schemes related to the DH scheme, such as the ElGamal encryption scheme [8]. Here Bob picks a random $0 \leq x < o$ and publishes his public key $y = h^x$. If Alice wants to send a message $m \in G$ to Bob, then Alice picks a random $0 \leq k < o$ and sends $(A, B) = (h^k, m \cdot y^k)$ to Bob. Upon receipt, Bob decrypts the message by forming B/A^x which is equal to the message m.
Two other problems are related to DH problem. The first one is the *Diffie-Hellman Decision* problem: given $a, b, c \in \langle h \rangle$ determine whether $c = DH(a, b)$. The DH problem is at least as difficult as the DH Decision problem. The second related problem is the *Discrete Logarithm* (DL) problem in $\langle h \rangle$ is given $a \in \langle h \rangle$, find $0 \leq x < o$ such that $a = h^x$. The DL problem is at least as difficult as the DH problem. For the security of the Diffie-Hellman scheme, all three problems should be intractable.

We call h' a *prime subgenerator* of h if $h' \in \langle h \rangle$ and the order of h' is a prime number. By virtue of the Pohlig-Hellman algorithm [17], the difficulty of the

DL problem w.r.t. h, given the factorization of the order of h, is as difficult as solving the DL problem for all prime subgenerators of h.

Clearly, one can construct the Diffie-Hellman and related schemes in any multiplicative subgroup $\Gamma = \langle\gamma\rangle$ of $GF(p^t)$ of order ω. For solving the discrete logarithm problem for a prime subgenerator γ' of γ of order ω', one can use an index calculus (IC) based algorithm that has a heuristic expected asymptotic running time of $L(p^s, 1/3, 1.923 + o(1))$, see [1] and [13], where s is the smallest divisor of t such that $\langle\gamma'\rangle$ is contained in a subfield of $GF(p^t)$ isomorphic to $GF(p^s)$. If $p = 2$ then the constant 1.923 can be replaced by 1.587, see [6]. Alternatively one can use Birthday Paradox (BP) based algorithms (e.g. Pollard's rho algorithm [18]) that have expected running times exponential in the size of the ω'. More precisely, breaking the Discrete Logarithm problem can be solved in expected $O(\sqrt{(}\omega'))$ elementary operations in $GF(p^t)$.

This leads us to the conclusion from [13] that *w.r.t. attacks known today* the intractability of the discrete logarithm problem in Γ depends on the existence of a prime generator γ' of γ whose minimal surrounding subfield and prime order are of sufficient size. The particular form of the field itself, is not relevant. In other words, if $GF(p^t)$ is the minimal surrounding field of a subgroup of prime order, then - *w.r.t. attack known today* - the discrete logarithm in this subgroup is approximately as difficult as the discrete logarithm in a subgroup of prime order of a basic field $GF(P)$ if the size of P is approximately equal to as t times the size of p, and the order of both subgroups are about the same size.

Hence, a suitable generator γ in a field extension $GF(p^t)$ should have a prime subgenerator that is not contained in one of the proper subfields and should have a suitably large order. In [13], A.K Lenstra proposes a simple, practical method for the construction of a field extension and suitable generator generating a relatively small subgroup in it. The idea is that one fixes the size of the prime number p and a number t such that p^t is "large" enough. Then one looks for a large prime factor ϖ in the value of the cyclotomic polynomial $\phi_t(p)$. The latter can be done using trial divisions with the primes up to, say 10^5; any other reasonable bound or method will do. Finally, one constructs a generator γ of order ϖ, by looking for an element different from 1, such that $g^{(p^t-1)/\varpi} = 1$. If one factorizes $p^t - 1$, one can also find a generator of the whole multiplicative group of $GF(p^t)$.

Using the above construction, the size of ϖ is about $\varphi(t) \cdot |p|$ bits (where $\varphi(.)$ is Euler's totient function) which grows as least as fast as $p^{t/\log(log(t))}$. The complexity of the BP based algorithms grows much faster, than the complexity of the IC based algorithms. So if the size of the surrounding field is large enough to resist the sub-exponential, IC based algorithms, then the order ϖ will usually be large enough "automatically" to resist the BP based algorithms as well.

2.2 Partial Security of DL-based cryptosystems over extension fields

Let $\gamma \in GF(p^t)$ be a generator of a group Γ of order ω. In our applications, γ is not contained in a genuine subfield of $GF(p^t)$ and ω is either equal to a large prime number ϖ or to $p^t - 1$. We start with some well-known notions; if J is an element of $GF(p)[X]$, i.e. a polynomial with coefficients in the basis field $GF(p)$, then for any natural number i, the i-th coefficient of J is denoted by $[J]_i$. If $F = \sum_{i=0}^{t} a_i X^i$ is an irreducible polynomial of degree t in $GF(p)[X]$, then we can describe the extension field $GF(p^t)$ as $GF(p)[X]/(F)$, i.e. each element f in $GF(p^t)$ can be uniquely written modulo F, as a polynomial of degree $< t$. In this setting, for any natural number i less than the degree of F, we let the i-th coefficient $[f]_i$ denote $[f \bmod F]_i$.
There are many such representations and some have special properties. For instance, if $t + 1$ is also prime and that p is a primitive element modulo $t + 1$, then one can use a special irreducible polynomial, the zeros of which form an optimal normal basis. Which such a basis, one can do exponentiation in $GF(p^t)$ even more efficiently than in a basic field of comparable size. See [13], where it is also shown how such pairs p, t, and the special irreducible polynomial can be easily generated.
Some constructions, like the coefficients, are dependent of the concrete representation of $GF(p^t)$ one has chosen. Throughout the remainder of this paper we assume that one has chosen an irreducible polynomial F, yielding a concrete representation of $GF(p^t)$.
Below we prove that computing coefficients (i.e. "small parts") of the Diffie-Hellman key, is as difficult as computing the whole key. These results are similar to the ones in [4] where the security of the Most Significant Bits (MSB) of Diffie-Hellman keys in a basic finite field $GF(p)$ is studied.

Theorem 2.1 *We use the above terminology. Given an oracle that computes a coefficient of a Diffie-Hellman key γ^{xy} on basis of γ^x, γ^y, then there exists a polynomial time algorithm that solves the Diffie-Hellman problem in $\Gamma = \langle\gamma\rangle$.*

Proof: The function that returns a coefficient of a fixed position less than t is linear and is hence a summing function (see below) by Proposition 2.3 below. The result now immediately follows from Theorem 2.4 below. □
It follows in particular that *any* bit of the Diffie-Hellman key in $GF(2^t)$ is as hard to compute as the whole. This offers a more conventional alternative for the new variant of the Diffie-Hellman protocol mentioned in [4] having exactly this property. In our situation, the size of p is not small (≥ 160 bit or more) and we immediately obtain the following corollary, which is crucial for the security of our scheme. This result immediately follows directly from Theorem 2.1.

Corollary 2.2 *Form, in the above terminology, the public key $\psi = \gamma^x$ with private key $0 < x < \omega$. Then solving $S \in GF(p)$ from the ElGamal encryption $(\gamma^k, S \cdot [\psi^k]_i)$ for some fixed $0 \leq i \leq t - 1$ and $1 \leq k \leq \omega$ random, is equivalent to solving the Diffie-Hellman problem in $\Gamma = \langle\gamma\rangle$.*

Theorem 2.1 is a consequence of a much broader result. Let n be a non-negative number and consider the integers $e_1, ..., e_n$ (the "exponents") and the elements $\lambda_1, ..., \lambda_n \in GF(p^t) \setminus \{0\}$ (the "multipliers") and consider the following *summing* function $Z(.) : \langle\gamma\rangle \to GF(p^t)$ defined by:

$$Z(\kappa) = \sum_{i=1}^{n} \lambda_i \cdot \kappa^{e_i}.$$

The number n is called the *degree* of the summing function and the number $d = \gcd(e_1, e_2, ..., e_n, \mathrm{ord}(\gamma))$ is called the *order* of the summing function. Note that if all multipliers are elements of the basic field $GF(p)$, then its form is representation-independent.
The following result implies that the collection of summing functions is rather large.

Proposition 2.3 *If $f(.)$ is a linear mapping from $GF(p^t)$ onto $GF(p)$ (i.e. a functional), then the restriction of $f(.)$ to $\langle\gamma\rangle$ is a summing function of order 1. Moreover, the multipliers occurring in the definition of summing function can be easily determined. In particular it follows that the restriction to $\langle\gamma\rangle$ of the* trace *function $Tr(.)$ of $GF(p^t)$ onto the base subfield $GF(p)$ defined by (cf. [16]):*

$$Tr(\alpha) = \alpha + \alpha^p + ... + \alpha^{p^{t-1}}.$$

is a summing function of order 1.

Proof: It is evident that the trace function is a summing function of order 1. From [16, Theorem 2.24] it follows that (the restriction to $\langle\gamma\rangle$ of) any functional is of the form $\alpha \to Tr_K(\beta \cdot \alpha)$ for some fixed $\beta \in GF(p^t)$ and is hence a summing function of order 1.
We note that given a functional $f(.)$, of which we assume it can be efficiently evaluate in our concrete representation, one can easily compute the associated β (and thus the multipliers) from $f(1), f(\gamma), ..., f(\gamma^{t-1})$ using the following equality:

$$\begin{pmatrix} 1 & 1 & ... & 1 \\ \gamma & \gamma^p & ... & \gamma^{p^{t-1}} \\ \vdots & \vdots & ... & \vdots \\ \gamma^{t-1} & \gamma^{(t-1)p} & ... & \gamma^{(t-1)p^{t-1}} \end{pmatrix} \cdot \begin{pmatrix} \beta \\ \beta^p \\ \vdots \\ \beta^{p^{t-1}} \end{pmatrix} = \begin{pmatrix} f(1) \\ f(\gamma) \\ \vdots \\ f(\gamma^{t-1}) \end{pmatrix} \qquad (1)$$

Recall that γ is an element in $GF(p^t))$ that can not be embedded in a proper subfield of $GF(p^t)$, so $\gamma, \gamma^p, ..., \gamma^{p^{t-1}}$ are all distinct and the above matrix is a non-singular matrix of the Vandermonde type. □

Theorem 2.4 *In the above terminology, let $Z(.)$ be a summing function of order d. Also let $\mathcal{O}$ be an oracle that on basis of any γ^a and γ^b computes $Z(\gamma^{ab})$. Then there exists a polynomial time algorithm that computes γ^{abd} on basis of γ^a and γ^b. That is, for $d = 1$ there exists a polynomial time algorithm that solves the whole Diffie-Hellman problem in $\langle\gamma\rangle$.*

Proof: Let $V = \gamma^x$ and $W = \gamma^y$ be any elements of $\langle\gamma\rangle$. With the help of oracle $\mathcal{O}$ one can not only determine $Z(\gamma^{xy})$, but also $Z(\gamma^{x(y+i)})$ (by giving the oracle the input V and $\gamma^i \cdot W$) for $i = 0, ..., n-1$, where n denotes the order of $Z(.)$. This results in the following equality:

$$\begin{pmatrix} 1 & 1 & ... & 1 \\ V^{e_1} & V^{e_2} & ... & V^{e_n} \\ \vdots & \vdots & ... & \vdots \\ V^{(n-1)\cdot e_1} & V^{(n-1)\cdot e_2} & ... & V^{(n-1)\cdot e_n} \end{pmatrix} \cdot \begin{pmatrix} \lambda_1 \cdot \gamma^{xye_1} \\ \lambda_2 \cdot \gamma^{xye_2} \\ \vdots \\ \lambda_n \cdot \gamma^{xye_n} \end{pmatrix} = \begin{pmatrix} Z(\gamma^{xy}) \\ Z(\gamma^{x(y+1)}) \\ \vdots \\ Z(\gamma^{x(y+n-1)}) \end{pmatrix} \quad (2)$$

The above matrix is of the Vandermonde type. Let us first consider the case that all $V^{e_1}, V^{e_2}, ..., V^{e_n}$ are all different. Then this matrix is regular, and so from equality (2) one can determine the elements $\gamma^{xy\cdot e_1}, \gamma^{xy\cdot e_2}, ...\gamma^{xy\cdot e_n}$. From this one can determine the element γ^{xyd} by taking a suitable combination of the $\gamma^{xy\cdot e_1}, \gamma^{xy\cdot e_2}, ...\gamma^{xy\cdot e_n}$.
To prove the general case, observe that if V^{e_i} and V^{e_j} are equal then so are γ^{xye_i} and γ^{xye_j}. So by restricting to a maximal collection of different V^{e_i}, one can determine γ^{xyd} by the above argument. □

We again use the above terminology. An element $\omega = \gamma^x \in \langle\gamma\rangle$ can also be described in terms of its minimal polynomial of degree t. This representation is not unique, as all conjugates of ω, i.e. $\omega, \omega^p, \omega^{p^2}, ..., \omega^{p^{t-1}}$, have the same minimal polynomial. The i-th order coefficient of this polynomial equals

$$(-1)^i S_i(\omega, \omega^p, \omega^{p^2}, ..., \omega^{p^{t-1}}), \quad (3)$$

where S_i denotes the elementary symmetric polynomial in t variables of degree i. One can easily prove that expression (3) yields a summing function. By virtue of [13, Lemma 2.4] it follows that for $1 \leq i \leq t-1$, the order of this summing function can not contain prime factors $\geq i$. That is, the order of this summing function is very small. Hence it follows from Theorem 2.4 that for $1 \leq i \leq t-1$, determining an i-th order coefficient of the minimal polynomial of the Diffie-Hellman key, is as difficult as determining the whole key.

Note that one can easily determine the minimal polynomial of γ^{xy} on basis of the minimal polynomial of either γ^x and y or the minimal polynomial of γ^y and x. On this idea variants of the Diffie-Hellman scheme can be based in which the involved parties send each other the minimal polynomials γ^x, γ^y rather than the elements themselves. The exchanged secret is the minimal polynomial of the element γ^{xy}, or some of the non-zero order coefficients of this. More generally one can take the value of a summing function of low order, evaluated in γ^{xy}.
It easily follows from Theorem 2.4 that such variants are as at *least* as secure as the original Diffie-Hellman scheme in our concrete representation of $\langle\gamma\rangle$. As minimal polynomials are representation-independent it directly follows that breaking such a variant means breaking the original Diffie-Hellman scheme in any representation of $\langle\gamma\rangle$. The converse, which is less interesting for the security of our variant, is also true but its proof is not elementary. The proof follows by determining zeros of the exchanged minimal polynomials in a fixed representation,

which can be done in deterministic polynomial time by the technique of H.W. Lenstra Jr. in [15].
The above variant is used in [5] to construct a variant of the Diffie-Hellman scheme in which the number of bits exchanged is only a third of what is normally used, while the offered security against attacks known is the same.

Thus far, we have only discussed the unlikeliness that non-noisy oracles exist, that always give correct answers. Now we will briefly discuss the possible existence of noisy oracles $\mathcal{O}$ that on basis of uniformly random γ^x and γ^y in $\langle\gamma\rangle$ compute $Z(\gamma^{xy})$, with non-negligible probability ϵ. In the general context *non-negligible* means that $1/\epsilon$ is less than a polynomial in $\log_2(|GF(p^t)|) = t \cdot \log_2(p)$. A first indication that such oracles do not exist follows from the easily verified observation that with such an oracle one can solve the Diffie-Hellman Decision problem in $\langle\gamma\rangle$, that is considered hard. See [19].
In our context (the scheme in section 4) it's more natural to say that a probability ϵ is *non-negligible* if $1/\epsilon$ is less than a polynomial in $\log_2(p)$. Indeed, a (small) parameter t is designed and then fixed; only the parameter p then varies. We have a heuristic proof for the following variant of Theorem 2.4 (of which similar variants of Proposition 2.1 and Corollary 2.2 can be easily deduced).

Theorem 2.5 *Let $Z(.)$ be a summing function on Γ of degree 1. Given a (noisy) oracle $\mathcal{O}$ that on basis of uniformly random γ^x and γ^y computes $Z(\gamma^{xy})$, with non-negligible probability ϵ then there exists a probabilistic polynomial time algorithm that solves the Diffie-Hellman problem in $\langle\gamma\rangle$.*

Heuristic Proof: Let $V = g^x$ and $W = g^y$ be any elements of $\langle\gamma\rangle$. With the help of oracle $\mathcal{O}$ one can determine $Z(\gamma^{(x+i)(y+j)})$ with probability ϵ for randomly chosen $0 < i, j < \omega$ (by giving the oracle the input $g^i \cdot V$ and $g^j \cdot W$).

We now perform the following operations. Consider k randomly pairs $(i_1, j_1), ...(i_k, j_k)$ where is k somewhat larger than $(d + log_2(d))^2/\epsilon$, and give the oracle the input $g^i \cdot V$ and $g^j \cdot W$ resulting in outputs $o_1, ..., o_k$. Now consider the following equality:

$$\begin{pmatrix} (V^{i_1}W^{j_1}\gamma^{i_1 j_1})^{e_1} & ... & (V^{i_1}W^{j_1}\gamma^{i_1 j_1})^{e_n} \\ (V^{i_2}W^{j_2}\gamma^{i_2 j_2})^{e_1} & ... & (V^{i_2}W^{j_2}\gamma^{i_2 j_2})^{e_n} \\ \vdots s & ... & \vdots \\ (V^{i_k}W^{j_k}\gamma^{i_k j_k})^{e_1} & ... & (V^{i_k}W^{j_k}\gamma^{i_k j_k})^{e_n} \end{pmatrix} \cdot \begin{pmatrix} \delta_1 \\ \delta_2 \\ \vdots \\ \delta_k \end{pmatrix} = \begin{pmatrix} o_1 \\ o_2 \\ \vdots \\ o_k \end{pmatrix} \tag{4}$$

We denote the matrix (respectively vector) on the lefthand side (4) by R (respectively (δ)), and the vector on the right hand side of (4) by (o). Next we look at all d row numbers and look at the associated $d \times d$ regular submatrix R' and the vector (o') of size d. We then try to uniquely solve (δ) from the equality $R' \cdot (\delta) = (o')$ and store this vector.
As soon as we obtain two coinciding results (δ') and (δ'') the algorithm finishes

and outputs a candidate for γ^{xy} by assuming that

$$(\delta') = \begin{pmatrix} \lambda_1 \cdot \gamma^{xye_1} \\ \lambda_2 \cdot \gamma^{xye_2} \\ \vdots \\ \lambda_n \cdot \gamma^{xye_n} \end{pmatrix}$$

and by taking a suitable combination of the δ_i (as in the proof of Theorem 2.4). We now indicate that with non-negligible probability this algorithm indeed returns γ^{xy}. If a vector (o') is not correct, then we may assume that any resulting (δ') behaves like a random element in $GF(p^t)$. So, the probability that two resulting (δ') and (δ'') coming from incorrect (o') and (o'') is negligible.
By the choice of k, we can expect with large probability that at least $(t+log_2(t))^2$ coordinates of the vector on the right hand side of equation (4) are correct. Now we assume that the matrix on the lefthand side (4) behaves like a random matrix with entries of $GF(p^t)$. One can easily show that the probability that any matrix of size $(d + log_2(d)) \times d$ is of rank d is larger than $1/d$. So with non-negligible probability one can expect two different submatrices R', R'' and whose associated (o') and (o'') are correct. □

3 Double Deckers in Field Extensions

From now on we let $G = \langle g \rangle$ be a cyclic group of order prime order p in which the discrete problem is intractable. We introduce a new notion, that combines G, $GF(p)[X]$, and $GF(p^t)$.

For a polynomial J in $GF(p)[X]$ we define the *power* polynomial, g^J by

$$\sum_{i=0}^{\deg(J)} g^{[J]_i} X^i.$$

J is called the *exponent* polynomial of g^J. If $F = \sum_{i=0}^{t} a_i X^i$ is an (irreducible) polynomial of degree t in $GF(p)[X]$, then we obtain a natural equivalence on the power polynomials: two power polynomials are equivalent iff the difference of their exponent polynomials is a multiple of F; the resulting equivalence group is called the group of *power polynomials modulo* F. It is clear that we can represent power polynomials modulo F as t tuples. The collection of power polynomials, resp. power polynomials modulo a polynomial F of degree t will be denoted by $\mathcal{E}$, resp. $\mathcal{E}/(F)$.

Let $P_1 = \sum_{i=0}^{m} U_i X^i$ and $P_2 = \sum_{i=0}^{n} V_i X^i$ be two power polynomials. Then the *product* of $P_1 \cdot P_2$ is defined as the power polynomial $\sum_{i=0}^{\max(m,n)} U_i \cdot V_i X^i$ The *inverse* P_1^{-1} of P_1 is defined as $\sum_{i=0}^{m} U_i^{-1} X^i$. If $P_1 = g^A$ and $P_2 = g^B$, then it follows that the $P_1 \cdot P_2 = g^{A+B}$ and $P_1^{-1} = g^{-A}$. Furthermore, if $C = \sum_{j=0}^{r} c_j X^j$ is a polynomial in $GF(p)[X]$, then P_1^C is defined as $\sum_{k=0}^{k=m+r} \prod_{i+j=k} U_i^{c_j} X^k$. If follows that if $P_1 = g^A$, that then P_1^C is equal to $g^{A \cdot C}$.

Proposition 3.1 *Let $P_1 = g^A = \sum_{i=0}^{m} U_i X^i$ be a power polynomial of degree m, and let F be an (irreducible) polynomial of degree t in $GF(p)[X]$. Then (without knowing A) one can determine $g^{A \bmod F}$.*

Proof: For a proof, let $F = \sum_{i=0}^{t} f_i X^i$, with $f_t \neq 0$. Observe that

$$g^{A - \frac{a_m}{f_t} \cdot F \cdot X^{t-m}} = g^A \cdot (U_m)^{-f_t^{-1} F \cdot X^{t-m}}.$$

which is a power polynomial of degree $< m$, equivalent with P_1 modulo F and constructible without knowledge of A. The proposition now follows from an inductive argument. □

As we remarked before, we represent any p-ary polynomial modulo an irreducible p-ary polynomial F of degree t (i.e. an element in $GF(p^t)$) as a p-ary polynomial of degree $< t$. We will also represent any p-ary power polynomial modulo F as one of degree $< t$. By the virtue of Proposition 3.1 we can determine this representation *without* explicitly knowing the exponent polynomial.
As before, we let γ be an element of a multiplicative group of $GF(p^t)$, that is not contained in a proper subfield of $GF(p^t)$ and let $\psi \in \Gamma = \langle \gamma \rangle$. Also, let the order of γ be denoted by ω.
Now, suppose that person P (for prover) gives the elements $\mu \in GF(p^t)$ and $M \in \mathcal{E}$ to person V (for verifier) and states:

Assertion DD: There exists a number k less than ω such that

$$\mu = \gamma^k \quad M = g^{(\psi^k)} \tag{5}$$

The following protocol, a variant of the double decker protocol in [24], lets P prove statement DD without revealing anything about k or ψ^k.

Protocol 3.2

Pr-1 *The Prover generates a random number l less than the order ω of γ, calculates $\nu = \gamma^l$, $N = g^{(\psi^l)}$ and hands ν and N over to V.*
Pr-2 *V generates a random $h \in \{0, 1\}$, and presents h as a challenge to P.*
Pr-3 *P calculates $z = l - h \cdot k \pmod{\omega}$ and hands z over to V.*
Pr-4 *V verifies that both*
Pr-4a. *$\nu = \gamma^z \cdot \mu^h$, and*
Pr-4b. *$N = g^{(\psi^z)}$ if $h = 0$ and $N = M^{(\psi^z)}$ if $h = 1$.*

The protocol satisfies the following properties which are easily verified.

Completeness If statement DD is true, then V will accept it.
Soundness If L-E is not true, then with a probability less than $1/2$ it will be accepted by V.
Security If L-E is true, then V can not learn any secret information on k or ψ^k by following the protocol.

Using the Fiat-Shamir heuristic [9] one can convert Protocol 3.2 in a non-interactive scheme with knowledge error 2^{-v} for a security parameter v. To this end, let $H(.)$ be a secure hash function with output length equal to v bits. For $i = 1, ..., v$ the Prover chooses $0 < l_i < \omega$ randomly en calculates (as in Pr-1 above) $\nu_i = \gamma^{l_i}$ and $N_i = g^{(\psi^{l_i})}$. Then he computes the v-tuple

$$Z = (z_1, ..., z_v) = (l_1 - h_1 \cdot k \pmod{\omega}, ..., l_v - h_v \cdot k \pmod{\omega})$$

where h_i denotes the i-th bit of $H = H(\mu, M, \nu_1, N_1, ..., \nu_v, N_v)$. The non-interactive proof consists of Z and H. To verify, for $i = 1, ..., v$, one re-constructs ν_i, N_i by:

$$\nu_i = \gamma_i^z \cdot \mu^{h_i}$$
$$N_i = \begin{cases} (\psi^z) & \text{if } h_i = 0 \\ M^{(\psi^z)} & \text{if } h_i = 1 \end{cases}$$

and then verifies that $H(\mu, M, \nu_1, N_1, ..., \nu_v, N_v)$ equals H. The size of the non-interactive proof is about $\omega \cdot v$ bits.
As is also explained in [3], a (dishonest) Prover could try to guess the v bits, h_i, and then form the accompanying responses z_i. If the Prover's ability to try this is bound by K times, then the *probability of failure* is of the non-interactive proof equals $K \cdot 2^{-v}$. The choice of v should take this into account. In [3] it is indicated that for a probability of failure less then 2^{-w}, one should take $v = 2w$, that is $K = 2^v$. Of course this is rather arbitrary.

4 Our Scheme

As before, let $G = \langle g \rangle$ be a multiplicative, cyclic group of prime order p in which the discrete logarithm problem is intractable. We will not further specify G, but in a typical example G is the multiplicative group of a finite field or the group of points on an elliptic curve over a finite field. The g, G and p are system parameters that are not yet fixed; only the size $|p|$ of p is fixed. As in the scheme of [27], we'll generate g, G, p until we find one for which we can construct a suitable KRA encryption scheme (see below). The user's public key will be of the form $y = g^x$ where $0 \leq x < p$ is the user's private key.

Our security objective is to design an asymmetric cryptosystem based on the Discrete Logarithm problem in a finite field that has "asymmetric security" of D bits, e.g. $D \geq 1024$. To this end, choose a number t such that $t \cdot |p| \geq D$. Next, construct random (user) system parameters g, G, p such that p is of size t. Next look for a large enough prime number ϖ in the value of the cyclotomic polynomial $\phi_t(p)$ (see Section 2), to support the security objective. If this fails, generate new random (user) system parameters g, G, p. Next, generate γ in $GF(p^t)$ of order ω either equal to $p^t - 1$ or ϖ. For the first generation, a factorization of $p^t - 1$ is required. So γ either generates the whole multiplicative group of $GF(p^t)$, or a subgroup of order ω. In either case, the Discrete Logarithm is infeasible w.r.t. attacks known today (see Section 2).

Both options correspond with the two types of ElGamal encryptions (classical and subgroup based) the KRA can use, each leading to a different scheme for CRs. In either case, the KRA's public key will be of the form $\psi = \gamma^{x_K}$ where $0 \leq x_K < \omega$ is the KRA's private key.
Both schemes actually support CRs for a variable number s (with $1 \leq s \leq t$, of different public/private keys $y_{s-1}, ..., y_0 \in G$ owned by the same user (e.g. supporting different applications or Certification Policies) simultaneously. So, by taking $s = 1$ in the below schemes we get a Certificate of Recoverability as defined by Young & Yung, for higher s we get a multiple version of this.

Scheme 4.1 *The user generates his s private keys, $x_0, ..., x_{s-1}$ and forms the following three components:*

C1 *The user's public keys $y_{s-1} = g^{x_{s-1}}, y_{s-2} = g^{x_{s-2}}, ..., y_1 = g^{x_1}, y_0 = g^{x_0}$, his identity and other attributes.*

C2 *The $s + 1$ tuple:*

$$(A, B_{s-1}, B_{s-2}, ..., B_0) = (\gamma^k, x_{s-1}^{-1} \cdot [\psi^k]_{s-1}, x_{t-2}^{-1} \cdot [\psi^k]_{s-2}, ..., x_0^{-1} \cdot [\psi^k]_0),$$

where $0 < k < \omega$ is randomly chosen by the user.

C3 *A transcript proving that the $x_{s-1}, x_{s-2}, ..., x_1, x_0$ appearing in component* **C1***, are also appearing in component* **C2***, i.e. are recoverable from it by the KRA.*

Observe that, provided that the components **C1** and **C2** are correctly formed, then the KRA can first recover ψ^k from A, then x_i^{-1} and thus x_i from B_i $(i = s-1, ..., 0)$. The transcript in C3, consists of two parts. The first part is the $(t - s)$-tuple

$$(d_{t-1}, d_{t-2}, ..., d_s) = (g^{[\psi^k]_{t-1}}, g^{[\psi^k]_{t-2}}, ..., (g^{[\psi^k]_s}). \tag{6}$$

The second part is a transcript of a non-interactive proof that there exists a $0 < k < \omega$ such the following two equalities in power polynomials hold:

$$\begin{aligned} g^{(\psi^k)} &= d_{t-1}X^{t-1} + d_{t-2}X^{t-1} + ... + d_sX^s + \\ &\quad + y_{s-1}^{B_{s-1}}X^{s-1} + y_{s-2}^{B_{s-2}}X^{s-2} + ... + y_0^{B_0} \\ \gamma^k &= A \end{aligned} \tag{7}$$

If the above two equalities hold for some k, then this shows that **C1** and **C2** are correctly formed, i.e. contain the same x_i for $i = s - 1, ..., 0$. It is clear from Section 3 how to form the transcript in the schemes as mentioned in Scheme 4.1. With respect to the choice of the security parameter v, 2^{-60} seems like a reasonable probability of failure. Moreover, certification is usually an on-line process, in which a users authenticates himself using an access code given by an RA. Hence one can easily incorporates an online challenge of the Certification Authority in the input of the hash resulting in the challenges $h_1, ..., h_v$, and puts a maximum time on the certification session, then 2^40 seems like a reasonable

bound for K. This means that $v = 100$ seems like a reasonable choice for the security parameter.

As an illustration, suppose that G is a group consisting of a collection of points on an elliptic curve of prime order p of 160 bit length, then one can compare this with the security of the ElGamal encryption system in the multiplicative group of a basic field of size 1600 bits, cf. [14]. So to get the same level for security for the KRA - as in the construction of CRs in [27] - one could employ our subgroup variant in $GF(p^t)$ with $t = 10$. This would result in a CR of about 64,000 bits, much smaller than the length of a comparable CR in the schemes in [27], which would be of size 160,000 bits. Observe that $t = 10$ also facilitates the use of optimal normal bases (as mentioned in Section 2.2), making our scheme also more efficient than comparable schemes in [27]. To get more security for the KRAs, say "1900 bits", one could use a similar technique in $GF(p^t)$ with $t = 12$. This would then still result in a CR of 64,000 bits, still smaller than the length of the above mentioned transcript.

4.1 Security Analysis of our Scheme

With respect to security, let us first look at Scheme 4.1 with $s = t$. Then the first part of component **C3** (mentioned in formula (6)) is empty, and it follows that **C3** is a transcript of a proof of knowledge from which no secret information can be extracted. Now, an attacker could try to find one of the user's private keys, x_i say, using one the following three strategies. First, he could try to find x_i from component **C1**, which would mean breaking the discrete logarithm in G.
Secondly, he could try to find x_i from component **C2**, which would mean breaking the Diffie-Hellman problem in Γ by Corollary 2.2. This problem is designed to be more difficult (in a scalable fashion) than to break the public key of the user.
Finally, he could try to find x_i from *combining* the information from the components **C1** and **C2**. As the discrete logarithm problems employed in **C1** and **C2** are only related by the number p, it seems unlikely that combining *both* components **C1** and **C2** will be very beneficiary. We'll now briefly discuss a formal result in this direction.

Theorem 4.2 *Let the ElGamal encryptions used for the KRA be classical, i.e. the generator γ generates the whole multiplicative group of $GF(p^t)$ and let i be one of $0, ..., t-1$. Also, let the factorization of $p^t - 1$ be known. Then using an efficient algorithm $\mathcal{P}$ computing x_i from* **C1** *and* **C2***, one can either construct*

1. *an efficient algorithm $\mathcal{P}_{\mathcal{DL}}$ solving the discrete logarithm problem in G, or*
2. *an efficient algorithm $\mathcal{P}_{\mathcal{DDH}}$ solving the Decision-Diffie-Hellman problem w.r.t. the base γ.*

Sketch of Proof: Let γ' be any prime subgenerator of γ. Using the technique appearing in the proof of [24, Proposition 1], one can verify that using an efficient algorithm $\mathcal{P}$ computing x_i from **C1** and **C2**, one can either construct

1. an efficient algorithm $\mathcal{P}_{\mathcal{DL}}$ solving the discrete logarithm problem in G, or
2. an efficient algorithm $\mathcal{P}_{\mathcal{DDH}(\gamma')}$ solving the Decision-Diffie-Hellman problem w.r.t. the base γ'.

So, using $\mathcal{P}$, one either obtains an efficient algorithm $\mathcal{P}_{\mathcal{DL}}$ solving the discrete logarithm problem in G, or one obtains efficient algorithms solving Decision-Diffie-Hellman problem w.r.t. all prime subgenerators in the multiplicative group of $GF(p^t)$. From the latter algorithms, one obtains an efficient algorithm $\mathcal{P}_{\mathcal{DDH}}$ solving the Decision-Diffie-Hellman problem w.r.t. the base γ. □

Let us assume that the DH decision problem in Γ is as hard as the DL problem in Γ, which is constructed to be more difficult than the DL problem in G. Then we can conclude from the previous result that the best strategy an attacker can follow to determine x from **C1** and **C2** is to break the DL problem in G. Hence, at least from a *theoretical* point of view, using the classical ElGamal scheme in our scheme is more secure than using the subgroup variant. We don't expect that using the subgroup variant of the ElGamal in our scheme is less secure in practice than using the classical ElGamal scheme.

We are left with a general security proof of the Scheme 4.1, let $s = s_0$ with $1 \leq s_0 < t$. Now, the idea is that we will construct an arbitrary "output" of our scheme with $s = s_0$ from an arbitrary "output" with $s = t$. So, if one has an advantage in breaking the scheme with $s = s_0$, one also has this advantage for breaking the scheme with $s = t$. As we have shown the security for the scheme with $s = t$ this would conclude the security proof of the scheme with $s = s_0$ also.

For this construction, let **C1**, **C2** and **C3** be an arbitrary output for the scheme with $s = t$. Forming **C1'** and **C2'** for the scheme with $s = s_0$ simply means removing some information from **C1** and **C2**. Also, from the transcript in **C3'** one obtains a representation of the power polynomial $g^{(\psi^k)}$ (see equality (7)). So we can form the first part $(d_{t-1}, d_{t-2}, ..., d_s)$ of **C3'** by simply taking the highest $t - s$ coefficients of the power polynomial $g^{(\psi^k)}$. Finally, the second part of **C3'** must be a random transcript that proves that equality (7) holds. However, the second part of **C3** provides us with such a random transcript.

4.2 Improvements and generalizations of our Scheme

In our basic scheme, the user has total freedom in choosing his private keys which might be useful in some situations (e.g. when using already existing public keys). However, if there are no restrictions on the private keys other than that they should be uniformly random, then the component $\mathbf{C_2}$ can be reduced, like is done in [27]. This consists of first forming γ^k where $0 < k < \omega$ is randomly chosen by the user and choosing the private x_i equal to $[\psi^k]_i$ for $i = 0, ..., s-1$. On basis on this one can form components $\mathbf{C_1}$ and $\mathbf{C_2}$, the latter which only needs to contain γ^k as $x_i^{-1} \cdot [\psi^k]_i = 1$. The component $\mathbf{C_3}$ is formed as before. The output of this scheme is $s \cdot \omega$ bits shorter than the basis scheme.
An additional advantage of this improvement would be that the construction of shadow public keys, see [12], would be more difficult than in the basic scheme.

Indeed, it is not hard to leak 20 bits of information in each of the y_i appearing in C_1 in the basic scheme as they can be independently chosen. So, if $t = 12$ then this would that 240 bits subliminal bits can be leaked in the component C_1. Of course, this problem arises with any private key-escrow scheme where the user can construct his public key himself and which allows for several, say 12, (certified) public keys to be used. Nonetheless, using the above mentioned improvement reduces this risk, as the public keys are no longer generated independently.

In our scheme, we have used a public key ψ of which the private part x_K was in the possession of only one Key Recovery Agent. It is well-known that Scheme 4.1 can very conveniently support the use of a KRA's public key in which the private key is secretly shared among n share-holders. For instance, suppose all KRA share-holders have chosen a private sub-key $0 < x_i < \omega$, and a public sub-key $\psi_i = \gamma^{x_i}$. Then their product ψ will be the shared KRA public key; it easily follows that private keys of users can reconstructed without the KRA share-holders having to come together. In the subgroup variant of our scheme where ω is a prime number, participating KRA's can (without the need for a trusted dealer), for any $1 \leq k \leq n$, construct a public key ψ in such a way that if it is used in our basic scheme, then the user's private key can be reconstructed only if k out of n KRA sub-holders cooperate. See [20] and [21]. Note that if one wants to use this technique to implement a k out of n key recovery scheme, one requires $\binom{k}{n}$ which grows exponentially in k.

Adapting the technique of [22] to our scheme is then a better option. This means that the private part, i.e. $\log_\gamma(\psi)$ of the public key ψ is "virtual", i.e. is not known by anybody. It might be more practically constructible to secretly share the private part of ψ by all participating KRAs using the technique indicated above. The user encrypts his private keys using our scheme resulting in the components $\mathbf{C_1}$, $\mathbf{C_3}$, $\mathbf{C_3}$. The user supplements this by using the non-interactive Public Verifiable Secret Sharing Scheme in [22], secretly sharing ψ^k among the KRA's, where k coincides with the one in equality (7). The choice of the number of KRAs required to retrieve the user's private key, is up to the user himself. Of course, accepting this choice is up to the Certificate Authority and should be in accordance with its policy. One can easily prove that supplementing the Public Verifiable Secret Sharing Scheme to our scheme does not weaken security.

5 Conclusion

We have proposed two new schemes for Certificates of Recoverability, making it possible for a PKI user to escrow its private keys in a publicly verifiable way, by means of encrypting it with a Key Recovery Agent's public key and depositing this with any other party. In our schemes, the cryptographic security employed by the Key Recovery Agents can be set higher, in a scalable fashion, than that being employed by the users. Among the other improvements of our scheme are its applicability to create CRs for cryptosystems based on the Discrete Log problem in small subgroups such as Elliptic Curve Cryptosystems, Also, the size of the

constructed proofs of knowledge can be taken smaller. We have additionally shown two ways to support secret sharing in our scheme. Finally, we have also presented several new constructions and results on the hardness of small parts, in the setting of Diffie-Hellman keys in extension fields.

References

1. M. Adleman, J. DeMarrais *A subexponentional algorithm over all finite fields*, CRYPTO '93 Proc., Springer-Verlag, 147-158.
2. N. Asokan, V. Shoup, M. Waidner, *Optimistic Fair Exchange of Digital Signatures*, Eurocrypt'98 Proc., Springer-Verlag, 591-606.
3. M. Bellare, P. Rogaway, *Random Oracles are Practical: A paradigm for Designing Efficient Protocols*, 1st ACM Conference on Computer and Communications Security, ACM Press, 1993, 62-73.
4. D. Boneh, R. Venkatesan, *Hardness of Computing the Most Significant Bits of Secret Keys in Diffie-Hellman and Related Schemes*, CRYPTO'96 Proc. Springer-Verlag, 129-142.
5. A.E. Brouwer, R. Pellikaan, E.R. Verheul, *Doing More with Fewer Bits*, Asiacrypt'99 Proc., Springer-Verlag.
6. D. Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, IEEE Trans. on IT, 30, 1984, 587-594.
7. W. Diffie, M.E. Hellman, *New directions in cryptography*, IEEE Trans. on IT 22, 1976, 644-654.
8. T. ElGamal, *A Public Key Cryptosystem and a Signature scheme Based on Discrete Logarithms*, IEEE Trans. on IT 31(4), 1985, 469-472.
9. A. Fiat, A. Shamir, *How to prove yourselve: Practical solutions to identification and signature problems*, CRYPTO'86 Proc., Springer-Verlag, 186-194.
10. D.M. Gordon, *Discrete Logarithms in $GF(p)$ using the number field sieve*, SIAM J. of Discrete Math., 6, 124-138.
11. J. Håstad, *On Using RSA with Low Exponent in a Public Key Network*, CRYPTO'85 Proc., Springer-Verlag, 403-405.
12. J. Kilian, F.T. Leighton, *Fair Cryptosystems Revisited*, Crypto'95 Proc., Springer-Verlag, 208-221.
13. A.K. Lenstra, *Using Cyclotomic Polynomials to Construct Efficient Discrete Logarithm Cryptosystems over Finite Fields*, ACISP97 Proc., Springer-Verlag, 127-138.
14. A.K. Lenstra, E.R Verheul *Selecting Cryptographic Key Sizes*, these proceedings.
15. H.W Lenstra, *Finding isomorphisms between two finite fields* Math. of Comp., 56 (1991), 329-347.
16. R. Lidl, H. Niederreiter, *Finite Fields*, Addison-Wesley, 1983.
17. S.C. Pohlig, M.E. Hellman, *An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance*, IEEE Trans. on IT, 24 (1978), 106-110.
18. J.M. Pollard, *Monte Carlo methods for index computation (mod p)*, Math. of Comp., 32 (1978), 918-924.
19. M. Naor, M. Yung, *Universal one-way functions and their cryptographic applications*, In 21st Annual ACM Symposium on Theory of Computer Science, 1997.
20. T.P. Pedersen, *Distributed Provers with Applications to Undeniable Signatures*, Eurocrypt'91 Proc., Springer-Verlag, 221-242.
21. T.P. Pedersen, *A Treshold Cryptosystem Without a Trusted Party*, Eurocrypt '91 Proc., Springer-Verlag, 522-526.

22. B. Schoenmakers, *A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting*, CRYPTO'99 Proc., 148-164.
23. C. Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology, 4, 1991, 161-174.
24. M. Stadler, *Publicly Verifiable Secret Sharing*, Eurocrypt'96 Proc., 190-199.
25. M. Stadler, J.-M. Piveteau, J. Camenisch, *Fair Blind Signatures*, Eurocrypt'95 Proc., 209-219.
26. D.R. Stinson *Cryptography: theory and practice*, CRC press, 1995.
27. A. Young, M. Yung, *Auto-Recoverable Auto-Certifiable Cryptosystems*, Eurocrypt'98 Proc., 16-31.