A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic Voting

Berry Schoenmakers

Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
berry@win.tue.nl

Abstract. A publicly verifiable secret sharing (PVSS) scheme is a verifiable secret sharing scheme with the property that the validity of the shares distributed by the dealer can be verified by any party; hence verification is not limited to the respective participants receiving the shares. We present a new construction for PVSS schemes, which compared to previous solutions by Stadler and later by Fujisaki and Okamoto, achieves improvements both in efficiency and in the type of intractability assumptions. The running time is $O(nk)$, where k is a security parameter, and n is the number of participants, hence essentially optimal. The intractability assumptions are the standard Diffie-Hellman assumption and its decisional variant. We present several applications of our PVSS scheme, among which is a new type of universally verifiable election scheme based on PVSS. The election scheme becomes quite practical and combines several advantages of related electronic voting schemes, which makes it of interest in its own right.

1 Introduction

Secret sharing and its many variations form an important primitive in cryptography. The basic model for secret sharing distinguishes at least two protocols: (i) a *distribution* protocol in which the secret is distributed by a dealer among the participants, and (ii) a *reconstruction* protocol in which the secret is recovered by pooling the shares of a qualified subset of the participants. Basic schemes (e.g., [Sha79,Bla79] for threshold secret sharing) solve the problem for the case that all players in the scheme are honest.

In verifiable secret sharing (VSS) the object is to resist malicious players, such as

(i) a dealer sending incorrect shares to some or all of the participants, and
(ii) participants submitting incorrect shares during the reconstruction protocol,

cf. [CGMA85]. In publicly verifiable secret sharing (PVSS), as introduced by Stadler [Sta96], it is an explicit goal that not just the participants can verify their own shares, but that anybody can verify that the participants received correct shares. Hence, it is explicitly required that (i) can be verified publicly. (As noted

Michael Wiener (Ed.): CRYPTO'99, LNCS 1666, pp. 148–164, 1999.

in [Sta96], the VSS scheme of [CGMA85] already achieved this property, but later VSS schemes weren't designed to be publicly verifiable.) Problem (ii) is usually dealt with implicitly though. In the schemes of [Fel87,Ped92b,Sta96,FO98] it suffices that the participants simply release their shares. Subsequently the released shares may be verified by anybody against the output of the distribution protocol.

Our PVSS schemes show that such an approach is not sufficient as a general model for PVSS. As an extension to the reconstruction protocol we will therefore require that the participants not only release their shares but also that they provide a proof of correctness for each share released (see Section 2).

For PVSS schemes it is natural to accept that the secret is computationally hidden. An information-theoretic hiding VSS scheme such as [Ped92a] requires the availability of *private* channels from the dealer to each of the participants individually. However, communication over the private channels is clearly not publicly verifiable.

Given that a PVSS scheme is computationally hiding, the question is what the exact number-theoretic assumptions are. For our new scheme, which works for any group for which the discrete log problem is intractable, we will relate the security to the Diffie-Hellman assumption and its decisional variant. These assumptions are common for encryption schemes in a discrete log setting. In fact, there is a direct connection with the security of the ElGamal cryptosystem, as, for instance, the semantic security of ElGamal encryption is equivalent to the Diffie-Hellman decision problem. So, in a sense, this type of assumption is the weakest one can hope for.

In contrast, the schemes of [Sta96,FO98] rely on special number-theoretic settings and intractability assumptions. The discrete log scheme of [Sta96] requires a special assumption involving "double discrete logarithms". Briefly, the idea is to consider expressions of the form $y = g^{(h^x)}$, where g is a generator of a group of order p, say, and h is a fixed element of high order in $\mathbb{Z}_p^*$. The "double discrete logarithm" assumption states that given y it is hard to find x, which is a non-standard assumption.

We observe that such a setting excludes the use of elliptic curves. Let us call the group generated by g the *base* group, and the group generated by h the *exponent* group. Now, since the notion of double logarithms relies on the fact that h^x can be interpreted as an integer modulo p (since $h^x \in \mathbb{Z}_p^*$), the exponent group can not simply be replaced by a group based on elliptic curves. The security of Stadler's scheme requires that the discrete problem for the exponent group is hard as well, hence p must be 1024 bits, say. We conclude that although the base group can be based on elliptic curves, its order p would be rather high (normally, elliptic curve cryptosystems attain their competitiveness by requiring the order to be of size 160 bits, say).

The scheme of [FO98] relies on what they call the "modified RSA assumption," which says that inverting the RSA function is still hard, even if the public exponent e may be chosen freely. Although this assumption is not known to be actually stronger than the RSA assumption, it potentially is. Furthermore, the number-theoretic setting for the primitives used (e.g., secure commitments) are computationally expensive and require a non-trivial set-up for the system, using zero-knowledge proofs to show that set-up is correct. Stadler [Sta96] also considers a PVSS scheme for sharing e-th roots, which at least depends on the RSA assumption and variants of the Diffie-Hellman assumption for composite moduli.

Summarizing, our PVSS construction will be much simpler than the above approaches. We only need techniques that work in any group for which the discrete log problem is intractable. The protocols consist of a few steps only, relying on simple primitives. The performance is not only asymptotically optimal, but also good in practice. And, finally, the PVSS scheme can readily be used to extend all kinds of applications without strengthening the security assumptions or essentially increasing the computational cost of the resulting scheme. This is essential in order for a primitive to be used in a modular fashion.

1.1 Overview

In Section 2 we will describe the characteristics of PVSS schemes. The model we propose follows [Sta96] with some refinements, and we consider what it means for a PVSS scheme to be homomorphic.

In Section 3 we then present our main construction of a simple PVSS scheme for sharing secrets chosen uniformly at random from a large set. The complexity of the PVSS scheme is *linear* in the security parameter k (and linear in the number of participants), which is essentially optimal. As a result our scheme is a factor of k more efficient than Stadler's discrete log scheme [Sta96], and it achieves the same asymptotic complexity as the scheme for e-th roots of [Sta96] and the PVSS scheme of [FO98]; as we will point out, however, the concrete running time for our scheme is significantly lower than for these schemes.

In Section 4 we consider extensions to the case that the secret is from a small set. Our PVSS schemes work for any group of prime order for which the discrete logarithm is hard. Its security relies on the standard Diffie-Hellman assumption and its decisional variant, which makes the PVSS scheme as secure as the ElGamal cryptosystem. We stress that, unlike [Sta96,FO98], we consider also whether a PVSS scheme leaks partial information on the secret. Similarly, we do not only consider PVSS for uniformly random secrets from a large set, but also for secrets from a small set, which need not be uniformly distributed either. This is of importance to applications such as electronic voting, in which the secret may consist of a single bit. (In Appendix A, we also show that the construction works for any access structure, as long as it admits a linear secret sharing scheme.)

In Section 5 we present a new approach to electronic voting based on PVSS. It turns out that the PVSS scheme exactly matches the discrete log setting and assumptions required for the remainder of the election scheme. The resulting scheme combines several of the advantages of previous election schemes, and fits conceptually between the schemes of [CFSY96] and [CGS97]. It achieves the same set of security properties as these schemes.

Finally, Section 6 contains several more applications of our PVSS schemes, including a generalization of Binding ElGamal [VT97] and other constructions related to software key escrow, such as [YY98]. We show how threshold variants for these schemes can be obtained easily.

2 Model for Non-interactive PVSS

In this section we first describe a model for non-interactive PVSS. We then discuss some aspects of the model, where we will also address the issue of homomorphic secret sharing for PVSS schemes.

We note that a distinctive feature of PVSS is that no private channels between the dealer and the participants are assumed. All communication is done over (authenticated) public channels using public key encryption. Consequently, the secret will only be hidden computationally.

In a PVSS scheme, a dealer D wishes to distribute shares of a secret value $s \in \Sigma$ among n participants $P_1, \ldots, P_n$. A monotone access structure describes which subsets of participants are qualified to recover the secret. For example, the access structure may be a (t, n)-threshold schemes, $1 \leq t \leq n$, which means that any subset of t or more participants will be able to recover the secret; any smaller subset will be unable to gain any information about the secret, unless a computational assumption is broken.

As a common structure for PVSS schemes we consider the following protocols. Note that initialization is done without any interaction between the dealer and the participants. In fact, participants may enter or leave the system *dynamically*; the only requirement is that a participant holds a registered public key.

Initialization All system parameters are generated as part of the initialization. Furthermore, each participant P_i registers a public key to be used with a public key encryption method E_i. The actual set of participants taking part in a run of the PVSS scheme must be a subset of the registered participants. We assume w.l.o.g. that participants $P_1, \ldots, P_n$ are the actual participants in the run described below.

Distribution The protocol consists of two steps:

1. *Distribution of the shares.* The distribution of a secret $s \in \Sigma$ is performed by the dealer D. The dealer first generates the respective shares s_i for participant P_i, for $i = 1, \ldots, n$. For each participant P_i the dealer publishes the encrypted share $E_i(s_i)$. The dealer also publishes a string $PROOF_D$ to show that each E_i encrypts a share s_i. Furthermore, the string $PROOF_D$ commits the dealer to the value of secret s, and it guarantees that the reconstruction protocol will result in the same value s.
2. *Verification of the shares.* Any party knowing the public keys for the encryption methods E_i may verify the shares. For each participant P_i a non-interactive verification algorithm can be run on $PROOF_D$ to verify that $E_i(s_i)$ is a correct encryption of a share for P_i. Since anyone may verify a share, it may be ruled out that a participant complains while it received a

correct share. In case one or more verifications fail, we therefore say that the dealer fails, and the protocol is aborted. (If some level of fault-tolerance is desired one may continue and think of it as a $(t, n-c)$-threshold scheme, where c is the number of verifications that failed.)

Reconstruction The protocol consists of two steps:

1. *Decryption of the shares.* The participants decrypt their shares s_i from $E_i(s_i)$. It is not required that all participants succeed in doing so, as long as a qualified set of participants is successful. These participants release s_i plus a string $PROOF_{P_i}$ that shows that the released share is correct.
2. *Pooling the shares.* The strings $PROOF_{P_i}$ are used to exclude the participants which are dishonest or fail to reproduce their share s_i correctly. Reconstruction of the secret s can be done from the shares of any qualified set of participants.

Compared to [Sta96], we have added the requirement for the reconstruction protocol that the participants must provide a proof of correct decryption of their shares. The proof is also non-interactive so that any party is able to sort out the correct shares and pool them together.

We have limited the description to non-interactive PVSS schemes by requiring that all $PROOF$s can be verified non-interactively. In fact, it is natural to reduce the amount of interaction between the players even more than for VSS schemes. Non-interactive VSS schemes, such as [Fel87,Ped92b], still include a stage in which participants file complaints if they received an incorrect share. Subsequently these complaints must be resolved to decide whether the distribution of the secret was successful. In non-interactive PVSS we have eliminated even this round of interaction: since any party can verify the output of the dealer, there is no need for the individual participants to check their own shares!

Homomorphic Secret Sharing The notion of homomorphic secret sharing is due to Benaloh [Ben87a], where its relevance to several applications of secret sharing is described, in particular electronic voting. Informally, homomorphic secret sharing is about combining shares of independent secrets in such a way that reconstruction from the combined shares results in a combined secret. In case of PVSS, there is an operation $\oplus$ on the shares, and an operation $\otimes$ on the encrypted shares such that for all participants

$$E_i(s_i) \otimes E_i(s_i') = E_i(s_i \oplus s_i').$$

Thus by decrypting the $\otimes$-combined encrypted shares, the recovered secret will be equal to $s \oplus s'$, assuming that the underlying secret sharing scheme is $\oplus$-homomorphic. In Section 5 we will present an electronic voting scheme that relies on a homomorphic PVSS scheme.

3 Special PVSS Scheme

We describe the construction for a (t, n)-threshold access structure, but it can be applied to any monotone access structure for which a linear secret sharing scheme exists (see Appendix A).

Let G_q denote a group of prime order q, such that computing discrete logarithms in this group is infeasible. Let g, G denote independently selected generators of G_q, hence no party knows the discrete log of g with respect to G. We solve the problem of efficiently sharing a random value from G_q. The dealer will achieve this by first selecting $s \in_R \mathbb{Z}_q$ and then distributing shares of the secret $S = G^s$. This approach allows us to keep the required proofs simple and efficient.

3.1 Protocols

We will use the protocol by Chaum and Pedersen [CP93] as a subprotocol to prove that $\log_{g_1} h_1 = \log_{g_2} h_2$, for generators $g_1, h_1, g_2, h_2 \in G_q$. We denote this protocol by $DLEQ(g_1, h_1, g_2, h_2)$, and it consists of the following steps, where the prover knows α such that $h_1 = g_1^\alpha$ and $h_2 = g_2^\alpha$:

1. The prover sends $a_1 = g_1^w$ and $a_2 = g_2^w$ to the verifier, with $w \in_R \mathbb{Z}_q$.
2. The verifier sends a random challenge $c \in_R \mathbb{Z}_q$ to the prover.
3. The prover responds with $r = w - \alpha c \pmod q$.
4. The verifier checks that $a_1 = g_1^r h_1^c$ and $a_2 = g_2^r h_2^c$.

Initialization The group G_q and the generators g, G are selected using an appropriate public procedure. Participant P_i generates a private key $x_i \in_R \mathbb{Z}_q^*$ and registers $y_i = G^{x_i}$ as its public key.

Distribution The protocol consists of two steps:

1. *Distribution of the shares.* Suppose w.l.o.g. that the dealer wishes to distribute a secret among participants $P_1, \ldots, P_n$. The dealer picks a random polynomial p of degree at most $t - 1$ with coefficients in $\mathbb{Z}_q$:

$$p(x) = \sum_{j=0}^{t-1} \alpha_j x^j,$$

and sets $s = \alpha_0$. The dealer keeps this polynomial secret but publishes the related commitments $C_j = g^{\alpha_j}$, for $0 \le j < t$. The dealer also publishes the encrypted shares $Y_i = y_i^{p(i)}$, for $1 \le i \le n$, using the public keys of the participants. Finally, let $X_i = \prod_{j=0}^{t-1} C_j^{i^j}$. The dealer shows that the encrypted shares are consistent by producing a proof of knowledge of the unique $p(i)$, $1 \le i \le n$, satisfying:

$$X_i = g^{p(i)}, \qquad Y_i = y_i^{p(i)}.$$

The non-interactive proof is the n-fold parallel composition of the protocols for $DLEQ(g, X_i, y_i, Y_i)$. Applying Fiat-Shamir's technique, the challenge c for the protocol is computed as a cryptographic hash of X_i, Y_i, a_{1i}, a_{2i}, $1 \leq i \leq n$. The proof consists of the common challenge c and the n responses r_i.

2. *Verification of the shares.* The verifier computes $X_i = \prod_{j=0}^{t-1} C_j^{i^j}$ from the C_j values. Using y_i, X_i, Y_i, r_i, $1 \leq i \leq n$ and c as input, the verifier computes a_{1i}, a_{2i} as
$$a_{1i} = g^{r_i} X_i^c, \qquad a_{2i} = y_i^{r_i} Y_i^c,$$
and checks that the hash of X_i, Y_i, a_{1i}, a_{2i}, $1 \leq i \leq n$, matches c.

Reconstruction The protocol consists of two steps:

1. *Decryption of the shares.* Using its private key x_i, each participant finds the share $S_i = G^{p(i)}$ from Y_i by computing $S_i = Y_i^{1/x_i}$. They publish S_i plus a proof that the value S_i is a correct decryption of Y_i. To this end it suffices to prove knowledge of an α such that $y_i = G^\alpha$ and $Y_i = S_i^\alpha$, which is accomplished by the non-interactive version of the protocol $DLEQ(G, y_i, S_i, Y_i)$.
2. *Pooling the shares.* Suppose w.l.o.g. that participants P_i produce correct values for S_i, for $i = 1, \ldots, t$. The secret G^s is obtained by Lagrange interpolation:
$$\prod_{i=1}^{t} S_i^{\lambda_i} = \prod_{i=1}^{t} \left(G^{p(i)}\right)^{\lambda_i} = G^{\sum_{i=1}^{t} p(i)\lambda_i} = G^{p(0)} = G^s,$$
where $\lambda_i = \prod_{j \neq i} \frac{j}{j-i}$ is a Lagrange coefficient.

Note that the participants do not need nor learn the values of the exponents $p(i)$. Only the related values $S_i = G^{p(i)}$ are required to complete the reconstruction of the secret value $S = G^s$. Also, note that participant P_i does not expose its private key x_i; consequently participant P_i can use its key pair in several runs of the PVSS scheme. The type of encryption used for the shares has been optimized for performance; however, if desired, it is also possible to use standard ElGamal encryption instead.

Clearly, the scheme is homomorphic. For example, given the dealer's output for secrets G^{s_1} and G^{s_2}, the combined secret $G^{s_1+s_2}$ can be obtained by applying the reconstruction protocol to the combined encrypted shares $Y_{i1}Y_{i2}$. We will use this to construct an election scheme in Section 5.

3.2 Performance

The dealer only needs to post $t+n$ elements of G_q (the numbers C_j and Y_i) plus $n+1$ number of size $|q|$ (the responses r_i and the challenge c). The number of exponentiations throughout the protocol is correspondingly low, and all of these exponentiations are with relatively *small* exponents from $\mathbb{Z}_q$ ($|q| = 160$ bits in practice).

Compared to Stadler's $O(k^2n)$ discrete log scheme, where k is a security parameter, we have reduced the work to $O(kn)$, which is asymptotically optimal. Compared to the e-th root scheme of Stadler [Sta96] and the scheme of [FO98], which achieve the same asymptotic complexity as our construction, our scheme is much simpler. The construction of [FO98] uses a rather complicated proof to show that a share is correctly encrypted. For example, in its simplest form, that is, using RSA with public exponent 3, the scheme requires at least 17 secure commitments per participant, where each commitment requires a two-way or three-way exponentiation. Moreover, the exponents are of full-length (1024 bits in practice). Therefore, we estimate our scheme to be faster by a factor of 25 to 50. Similarly, but to a lesser extent, the primitive operations for Stadler's e-th root scheme, are more costly than for our scheme.

3.3 Security

We first consider the security of the share-encryptions. We observe that directly breaking the encryptions used in our PVSS scheme implies breaking the Diffie-Hellman assumption. This can be seen as follows. Breaking the encryption of the shares amounts to finding $G^{p(i)}$ given g, G, X_i, y_i, Y_i, for the group G_q. Writing $G = g^\alpha$, $X_i = g^\beta$, $y_i = g^\gamma$, breaking the encryption of the shares is equivalent to computing $g^{\alpha\beta}$, given g^α, g^β, g^γ, and $g^{\beta\gamma}$, for $\alpha, \beta, \gamma \in_R \mathbb{Z}_q$. Recalling that the Diffie-Hellman assumption states that it is infeasible to compute $g^{\alpha\beta}$, given g^α and g^β, we have the following lemma.

Lemma 1. *Under the Diffie-Hellman assumption, it it infeasible to break the encryption of the shares.*

Proof. Given $x = g^\alpha$ and $y = g^\beta$, we want to obtain $z = g^{\alpha\beta}$ by using an algorithm $\mathcal{A}$ that breaks the encryption of the shares. Pick random α', β', γ, and feed $x^{\alpha'}, y^{\beta'}, g^\gamma, y^{\beta'\gamma}$ to $\mathcal{A}$. Since the input to $\mathcal{A}$ is uniformly distributed, we then obtain $z' = g^{\alpha\alpha'\beta\beta'}$ with some success probability ϵ. By taking $z'^{1/(\alpha'\beta')} = g^{\alpha\beta}$, we are thus able to compute z with the same success probability ϵ. □

A stronger result is that the secret is protected unless t or more participants cooperate. This is expressed by the following lemma.

Lemma 2. *Suppose that $t-1$ participants pool their shares and obtain the secret. Then we can break the Diffie-Hellman assumption.*

Proof. Let g^α and g^β be given, so we want to obtain $g^{\alpha\beta}$. We assume that α and β are random; if not, it is trivial to adapt the proof, as in the previous lemma. Suppose w.l.o.g. that participants $P_1, \ldots, P_{t-1}$ are able to break the scheme. We will show how to set up the system such that this fact enables us to compute $g^{\alpha\beta}$.

We put $G = g^\alpha$ and $C_0 = g^\beta$, which implicitly defines $p(0)$ as it is required that $C_0 = g^{p(0)}$. The points $p(1), \ldots, p(t-1)$ are chosen at random from $\mathbb{Z}_q$, which fixes polynomial p. This allows us to directly compute $X_i = g^{p(i)}$ and

$Y_i = y_i^{p(i)}$, for $i = 1, \dots, t-1$. Since $p(0)$ is only given implicitly, we cannot compute the points $p(t), \dots, p(n)$. It suffices, however, that we can compute $X_i = g^{p(i)}$ by Lagrange interpolation, which also yields the remaining C_j's. We now deviate from the protocol by computing the public keys y_i of participants P_i, $i = t, \dots, n$, as $y_i = g^{w_i}$ for random $w_i \in \mathbb{Z}_q$, and we set $Y_i = X_i^{w_i}$ such that $Y_i = y_i^{p(i)}$, as required.

The complete view for the system is now defined. It is consistent with the private view of participants $P_1, \dots, P_{t-1}$, and comes from the right distribution. Now, suppose that they are able to compute the secret $G^{p(0)}$. Since $G = g^\alpha$ and $p(0) = \beta$, we are thus able to compute $g^{\alpha\beta}$. This contradicts the Diffie-Hellman assumption. □

Note that we are assuming a static adversary. The above argument may be extended to the case where the static adversary is allowed to take part in the PVSS protocol K times, say, before breaking it. In that case we follow the protocol (hence we know the polynomials p) for the first K runs except that for participants $P_t, \dots, P_n$ we will set $S_i = G^{p(i)}$ directly instead of decrypting Y_i.

So far we have ignored the proofs that are required at several points in the protocol. However, in the random oracle model these proofs can easily be simulated. This leads to the following summary.

Theorem 1. *Under the Diffie-Hellman assumption, the special PVSS scheme is secure in the random oracle model. That is, (i) the reconstruction protocol results in the secret distributed by the dealer for any qualified set of participants, (ii) any non-qualified set of participants is not able to recover the secret.*

Proof. It follows from the soundness of the Chaum-Pedersen proof and the fact that the X_i's are obtained from the C_j's as $X_i = \prod_{j=0}^{t-1} C_j^{i^j}$ that the shares of the participants are consistent with the secret. It follows from Lemma 2 and the fact that the Chaum-Pedersen proof is honest-verifier zero-knowledge (hence the non-interactive version releases no information under the random oracle assumption) that no set of less than t participants can recover the secret. □

Theorem 1 does not claim that the participants cannot get any partial information on the secret G^s. This stronger result holds under the assumption that ElGamal encryption is semantically secure, which is known to be equivalent to the Decision DH assumption. The latter assumption states that it is infeasible to determine whether a given triple is of the form $(g^\alpha, g^\beta, g^{\alpha\beta})$ or $(g^\alpha, g^\beta, g^\delta)$, for random α, β, δ.

The above results are easily adapted to this case. For the equivalent of Lemma 1 we reason as follows. Suppose that an adversary is able to determine whether an encrypted share is equal to a given value g^δ or not. We then obtain a contradiction with the Decision DH assumption, closely following Lemma 1, by setting $G = g^\alpha$, $X_i = g^\beta$, and for random γ, setting $y_i = g^\gamma$ and $Y_i = (X_i)^\gamma = g^{\beta\gamma}$. Since the share is equal to $G^\beta = g^{\alpha\beta}$ it follows that we are able to distinguish $g^{\alpha\beta}$ from g^δ, if the adversary is able to distinguish the share from g^δ. The equivalent of Lemma 2 can be proved in a similar way. This leads to the following conclusion.

Theorem 2. *Under the DDH assumption and the random oracle assumption, the special PVSS scheme is secure. That is, (i) the reconstruction protocol results in the secret distributed by the dealer for any qualified set of participants, (ii) any non-qualified set of participants is not able to recover any (partial) information on the secret.*

4 General PVSS Schemes

The special PVSS scheme solves the basic problem of sharing a *random* secret from G_q. In this section we show how to extend this to any type of given secret $\sigma \in \Sigma$, where $2 \leq |\Sigma| \leq q$. Hence, a small set $|\Sigma| = 2$ is allowed, and it is not required that σ is uniformly distributed over Σ. We describe two methods.

For the first method, the general procedure is to let the dealer first run the distribution protocol for a random value $s \in \mathbb{Z}_q$, and then publish $U = \sigma \oplus \mathcal{H}(G^s)$, where $\mathcal{H}$ is an appropriate cryptographic hash function. The reconstruction protocol will yield G^s, from which we obtain $\sigma = U \oplus \mathcal{H}(G^s)$. See Section 6.1 for a similar technique. More generally, U may be viewed as an encryption of σ under the key G^s.

A second, more specific, method for the case that $\Sigma \subseteq G_q$ works as follows. The dealer runs the distribution protocol for s and also publishes $U = \sigma G^s$. Upon reconstruction of G^s, this will yield $\sigma = U/G^s$. Extending Theorem 2, security is maintained under the Decision DH assumption, even if Σ is a small set. (Recall that $C_0 = g^s$ is part of the output of the distribution protocol of our special PVSS scheme. Hence together with U this constitutes an ElGamal encryption of the form $(g^s, G^s\sigma)$.) This method also allows us to share bit strings without using a hash function as in the first method, e.g., if $\Sigma = \{1, \ldots, G^{2^u-1}\}$ we may share u bits at once, as long as $\log_G \sigma$ can be computed efficiently for $\sigma \in \Sigma$.

For the above methods, it does not necessarily follow that for a given value of U the reconstructed σ will be an element of Σ. However, it does follow for the first method in the common case that $\Sigma = \{0,1\}^u$ and the range of $\mathcal{H}$ is also equal to $\{0,1\}^u$, and also for the second method in case $\Sigma = G_q$. In other cases, an additional proof of knowledge may be required to show that indeed $\sigma \in \Sigma$ (e.g., see the next section), or, depending on the application, it may be sufficient to discard σ's outside Σ (or replace it by a default value in Σ).

5 Electronic Voting

In this section we briefly show how to construct a universally verifiable secret-ballot election scheme using PVSS as a basic tool. We show that by using our PVSS scheme we get a simple and efficient election scheme. This is not the case for the PVSS schemes of [Sta96] and [FO98].

We follow the model for universally verifiable elections as introduced by Benaloh *et al.* [CF85,BY86,Ben87b], which assumes the availability of a so-called bulletin board, to which all of the players in the scheme will post their messages. The players comprise a set of tallying authorities (talliers) $A_1, \ldots, A_n$, a set of voters $V_1, \ldots, V_m$, and a set of passive observers. These sets need not be disjoint. For example, in small-scale elections (e.g., board-room elections), each player may be both a voter and a tallier.

An election proceeds in two phases. In the first phase, the voters post their ballots, which contain the votes in encrypted form, to the bulletin board. Since the voters need not be anonymous in this scheme it is trivial to prevent double voting. Only well-formed (valid) ballots will be accepted. In the second phase, the talliers use their private keys to collectively compute the final tally corresponding with the accumulation of all valid ballots.

The protocols are as follows. Technically, each voter will act as a dealer in the PVSS scheme, where the talliers act as the participants. The initialization of the PVSS scheme is run, and we assume that each tallier A_i has registered a public key y_i.

Ballot casting A voter V casts a vote $v \in \{0, 1\}$ by running the distribution protocol for the PVSS scheme from Section 3, using a random secret value $s \in_R \mathbb{Z}_q$, and computing the value $U = G^{s+v}$. In addition, the voter constructs a proof $PROOF_U$ showing that indeed $v \in \{0, 1\}$, without revealing any information on v. $PROOF_U$ refers to the value of $C_0 = g^s$ which is published as part of the PVSS distribution protocol, and shows that:

$$\log_G U = \log_g C_0 \quad \vee \quad \log_G U = 1 + \log_g C_0$$

Such a proof can be efficiently constructed using the technique of [CDS94], see Appendix B. The ballot for voter V consists of the output of the PVSS distribution protocol, the value U, and $PROOF_U$.

Due to the public verifiability of the PVSS scheme and of $PROOF_U$, the ballots can be checked by the bulletin board when the voters submit their ballots. No involvement from the talliers is required in this stage.

Tallying Suppose voters V_j, $j = 1, \ldots, m$ have all cast valid ballots. The tallying protocol uses the reconstruction protocol of the special PVSS scheme, except that we will first exploit its homomorphic property. We first accumulate all the respective encrypted shares, that is, we compute the values Y_i^*, where the index j ranges over all voters:

$$Y_i^* = \prod_{j=1}^{m} Y_{ij} = y_i^{\sum_{j=1}^{m} p_j(i)}.$$

Next, each tallier A_i applies the reconstruction protocol to the value Y_i^*, which will produce $G^{\sum_{j=1}^{m} p_j(0)} = G^{\sum_{j=1}^{m} s_j}$, due to the homomorphic property. Combining with $\prod_{j=1}^{m} U_j = G^{\sum_{j=1}^{m} s_j + v_j}$, we obtain $G^{\sum_{j=1}^{m} v_j}$, from which the tally $T = \sum_{j=1}^{m} v_j$, $0 \leq T \leq m$, can be computed efficiently.

Conceptually, the above election scheme fits between the schemes of [CFSY96] and [CGS97]. It achieves the same level of security with regard to universal verifiability, privacy, and robustness. Voter independence can also be achieved in the same way as in those papers. The main difference with [CGS97] is that our scheme does not require a shared-key generation protocol for a threshold decryption scheme. Such a key generation protocol needs to be executed between the talliers, and requires several rounds of interaction. An example is Pedersen's key generation protocol [Ped91], or rather the improvement by [GJKR99].

For large-scale elections, such a key generation protocol is affordable, as it reduces the work for the election itself. For elections on a smaller scale, though, the cost of shared-key generation may dominate the cost of the entire election. In particular, when the group of talliers varies from election to election, the shared-key generation phase is better avoided. Further, in small-scale elections (e.g., board-room elections), it is realistic to let each voter play the role of tallier as well. Using our new election scheme, the election consists of just the two phases described above, without relying on any interaction between the voters or between the talliers.

It is possible to instantiate the scheme of [CFSY96] to get the same type of scheme. Recall that [CFSY96] requires a private channel from each of the voters to each of the talliers. By replacing the private channels by public key encryption, the information-theoretic privacy for the voters is lost, but a scheme is obtained that also avoids the use of a shared-key generation protocol. This approach has two major shortcomings:

– The shares for the talliers are not publicly verifiable, hence we have to allow for an additional stage in the election in which the talliers may file their complaints, which must then be checked by the others, and so on, to sort out which ballots and which talliers will be part of the tallying phase. In our scheme, the talliers are not involved at all during the voting stage.
– Each tallier must decrypt its shares one by one, since there is no homomorphic property. This means that all of the encrypted shares have to be communicated to the talliers, and that the talliers have to use their private keys to decrypt each of them individually, instead of decrypting just a single accumulated share.

Given that we settle for computational privacy, we thus conclude that a PVSS-based scheme can be used for elections on a smaller scale, while [CGS97] is to be used for large-scale elections.

6 Other Applications

We present a few more examples of situations in which our PVSS scheme can be applied. We expect that PVSS offers an efficient alternative in many protocols which use VSS as a subroutine.

6.1 Threshold Binding ElGamal

In [VT97] Verheul and van Tilborg present a scheme called Binding ElGamal that may be used to enhance the strength of a public-key infrastructure depending on software key escrow. The basic scheme provides a mechanism to copy a session-key not only to the intended receiver but also to n trustees. In other words, the trustees share the session-key according to a $(1, n)$-threshold structure. Extensions to (t, n)-threshold scenarios are also described in [VT97]. However, in these extensions it is required that the participants first engage in a shared-key generation protocol (as in the previous section). In principle, key generation needs to be done anew each time trustees are added or removed.

Using our new PVSS scheme, we obtain a *dynamic* version of threshold Binding Elgamal. The sender of the message acts as the dealer in the PVSS scheme. Let y denote the public key of the intended receiver. Next to the output of the distribution protocol, the sender also publishes the value $Y = y^s$ plus a proof that $\log_g C_0 = \log_y Y$, where $C_0 = g^s$ is part of the output of the distribution protocol in Section 3. This guarantees that the intended receiver and the participants will reconstruct the same value G^s. For use as a session-key in a symmetric encryption scheme, this value may be compressed to $\mathcal{H}(G^s)$. Clearly, this variant of Binding Elgamal is open to the same kind of criticism as the basic scheme (see, e.g., [PW98]).

The advantage of this approach is that, depending on the type of message, the sender may select a group of participants that will be able to reconstruct the session-key; these participants need not engage in a prior key generation protocol. Note that we are in fact combining a $(1, 1)$-threshold structure with a (t, n)-threshold structure (see also Appendix A).

6.2 Threshold Revocable Electronic Cash

Recent techniques to make privacy-protected cash revocable, are described in [CMS96] and [FTY96]. As a concrete example, in [CMS96] the customer must provide the bank with a value $d = y_T^\alpha$, where $\alpha \in_R \mathbb{Z}_q$, as part of the withdrawal protocol. Here y_T denotes the public key of the trustee. We can use our PVSS scheme to share the value d among a dynamically chosen set of trustees. Note that, as above, one can use threshold decryption with y_T representing the public key of the group as an alternative.

6.3 Threshold Software Key Escrow

Recently, Young and Yung [YY98] presented a scenario and solution for software key escrow. In their approach each user generates its own key pair, and registers it with a certification authority. The user must also encrypt shares of the secret key for a specified group of trustees. The key pair is only accepted, if the user also provides a (non-interactive) proof that the encrypted shares indeed correspond to the registered public key.

Following the model of [YY98] a possible alternative to their key escrow solution for threshold scenarios is as follows. The user generates a random private key $S = G^s$, and shares this value among the trustees using our PVSS scheme. Furthermore, the user publishes its public key $H = f^{(G^s)}$, where f is an appropriate generator (as in [Sta96]). Finally, to show that the trustees indeed receive shares of the private key $S = G^s$, the user proves knowledge of a witness s satisfying

$$H = f^{(G^s)} \ \wedge\ C_0 = g^s,$$

where C_0 is the value published in the distribution protocol in Section 3. The protocol used by Stadler in his PVSS scheme exactly matches this problem (see [Sta96, Section 3.3]). So, although we have to resort to the use of double discrete logarithms for this application, we only have to do this once (and not once per trustee!).

Decryption of an ElGamal ciphertext $(x, y) = (f^\alpha, H^\alpha m)$ is accomplished by raising x to the power G^s. We assume w.l.o.g. that the first t trustees take part in the decryption of the ciphertext. Using the fact that $G^s = \prod G^{p(i)\lambda_i}$, decryption works by setting $z_0 = x$, and letting the i-th trustee transform z_i to $z_{i+1} = z_i^{(G^{p(i)\lambda_i})} = z_i^{(S_i^{\lambda_i})}$. If desired, each trustee also produces a proof that its decryption step is correct. Again, Stadler's proof can be used for this purpose, this time to show that:

$$z_{i+1} = z_i^{(Y_i^{\lambda_i/x_i})} \ \wedge\ g^{\lambda_i} = y_i^{\lambda_i/x_i},$$

where y_i is the i-th trustee's public key and Y_i is its encrypted share of G^s.

7 Conclusion

We presented a new scheme for PVSS including several interesting applications. We stressed the virtues of the new approach, among which are its improved performance and its simplicity. Our construction hinges on the observation that it is advantageous to distribute and reconstruct a secret of the form G^s for fixed G and known random s instead of trying to reconstruct s itself. We hope to find applications of this idea in other settings too.

We have shown that many aspects play a role when selecting a secret sharing scheme for a particular application. It turns out that often we find a trade-off between the use of PVSS and the use of a threshold decryption scheme. As an example, we have considered the choice between a new PVSS based election scheme and an election scheme based on threshold decryption ([CGS97]) for large-scale elections. For elections on a smaller scale the costly key generation protocol, which is part of a threshold decryption scheme, can be avoided by using a more dynamic PVSS based approach.

Acknowledgments

Ronald Cramer, Markus Stadler, Martijn Stam, Moti Yung and the anonymous referees are gratefully acknowledged for their constructive comments.

References

Ben87a. J. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 251–260, Berlin, 1987. Springer-Verlag.

Ben87b. J. Benaloh. *Verifiable Secret-Ballot Elections.* PhD thesis, Yale University, Department of Computer Science Department, New Haven, CT, September 1987.

Bla79. G.R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the National Computer Conference 1979*, volume 48 of *AFIPS Conference Proceedings*, pages 313–317, 1979.

Bri89. E. F. Brickell. Some ideal secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 9:105–113, 1989.

BY86. J. Benaloh and M. Yung. Distributing the power of a government to enhance the privacy of voters. In *Proc. 5th ACM Symposium on Principles of Distributed Computing (PODC '86)*, pages 52–62, New York, 1986. A.C.M.

CDM99. R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret sharing scheme, 1999. Manuscript.

CDS94. R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187, Berlin, 1994. Springer-Verlag.

CF85. J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pages 372–382. IEEE Computer Society, 1985.

CFSY96. R. Cramer, M. Franklin, B. Schoenmakers, and M. Yung. Multi-authority secret ballot elections with linear work. In *Advances in Cryptology—EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 72–83, Berlin, 1996. Springer-Verlag.

CGMA85. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proc. 26th IEEE Symposium on Foundations of Computer Science (FOCS '85)*, pages 383–395. IEEE Computer Society, 1985.

CGS97. R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology—EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118, Berlin, 1997. Springer-Verlag.

CMS96. J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. In *Computer Security – ESORICS 96*, volume 1146 of *Lecture Notes in Computer Science*, pages 33–43, Berlin, 1996. Springer-Verlag.

CP93. D. Chaum and T. P. Pedersen. Transferred cash grows in size. In *Advances in Cryptology—EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 390–407, Berlin, 1993. Springer-Verlag.

Fel87. P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. 28th IEEE Symposium on Foundations of Computer Science (FOCS '87)*, pages 427–437. IEEE Computer Society, 1987.

FO98. E. Fujisaki and T. Okamoto. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In *Advances in*

Cryptology—EUROCRYPT '98, volume 1403 of *Lecture Notes in Computer Science*, pages 32–46, Berlin, 1998. Springer-Verlag.

FTY96. Y. Frankel, Y. Tsiounis, and M. Yung. "Indirect discourse proofs": Achieving efficient fair off-line e-cash. In *Advances in Cryptology—ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 286–300, Berlin, 1996. Springer-Verlag.

GJKR99. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 295–310, Berlin, 1999. Springer-Verlag.

KW93. M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference*, pages 102–111. IEEE Computer Society Press, 1993.

Ped91. T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology—EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, Berlin, 1991. Springer-Verlag.

Ped92a. T. P. Pedersen. *Distributed Provers and Verifiable Secret Sharing Based on the Discrete Logarithm Problem.* PhD thesis, Aarhus University, Computer Science Department, Aarhus, Denmark, March 1992.

Ped92b. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Berlin, 1992. Springer-Verlag.

PW98. B. Pfitzmann and M. Waidner. How to break fraud-detectable key recovery. *Operating Systems Review*, 32(1):23–28, 1998.

Sha79. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

Sta96. M. Stadler. Publicly verifiable secret sharing. In *Advances in Cryptology—EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199, Berlin, 1996. Springer-Verlag.

VT97. E. Verheul and H. van Tilborg. Binding ElGamal: A fraud-detectable alternative to key-escrow proposals. In *Advances in Cryptology—EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 119–133, Berlin, 1997. Springer-Verlag.

YY98. A. Young and M. Yung. Auto-recoverable auto-certifiable cryptosystems. In *Advances in Cryptology—EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 17–31, Berlin, 1998. Springer-Verlag.

A Extension to any Linear Secret Sharing Scheme

In Section 3 we have described a PVSS scheme based on Shamir's secret sharing scheme. However, the construction readily extends to any linear secret sharing scheme. A secret sharing scheme is linear if the dealer and the participants only use linear operations to compute with the shares and the secret. Below, we present a PVSS scheme based on Brickell's vector space construction [Bri89]. A more general class of linear secret sharing schemes is formed by the class of monotone span programs [KW93], which turns out to play an important role in the general context of secure multi-party computation (see [CDM99]).

Let Γ be a monotone access structure on $\{1,\ldots,n\}$ and consider the following secret sharing scheme with secrets and shares in $\mathbb{Z}_q$, where q is prime. Suppose we have constructed an $n \times d$ matrix M, for some d, satisfying for all $B \subseteq \{1,\ldots,n\}$:

$$B \in \Gamma \quad \equiv \quad e_1 \in \{c\, M_B \mid c \in \mathbb{Z}_q^{|B|}\}, \tag{1}$$

where $e_1 = (1,0,\ldots,0)$, and M_B denotes the submatrix consisting of all rows indexed by B. Hence, a set B of participants is qualified just when e_1 is contained in the span of the rows of M_B.

To distribute a random secret s, the dealer picks a random column vector $a \in \mathbb{Z}_q^d$, sets $s = a_1$, and computes the share for participant P_i as $s_i = M_i a$. As before, the dealer publishes the commitments $C_j = g^{a_j}$, for $0 \le j < t$, and the encrypted shares $Y_i = y_i^{s_i}$, for $1 \le i \le n$. To prove the consistency of the encrypted shares, the dealer proves that $\log_g X_i = \log_{y_i} Y_i$, where $X_i = g^{s_i}$ can be computed by anyone (due to the linearity):

$$X_i = \prod_{j=1}^{d} (C_j)^{M_{ij}} = g^{\sum_{j=1}^{d} M_{ij} a_j} = g^{s_i}$$

As before, for reconstruction the participants decrypt their shares to obtain $\{G^{s_i}\}_{i \in B}$, for some $B \in \Gamma$. Using a vector c for which $e_1 = c\, M_B$, which exists on account of (1), we then have (again due to the linearity):

$$\prod_{i \in B} (G^{s_i})^{c_i} = G^{\sum_{i \in B} (M_i a) c_i} = G^{(\sum_{i \in B} c_i M_i) a} = G^{e_1 a} = G^s.$$

Clearly, the resulting schemes are both simple and efficient. In general, only a single commitment C_j is required per random value chosen by the dealer, and one encryption Y_i per share s_i plus a response r_i for the consistency proof. Note that the construction also works for secret sharing schemes in which participants may receive multiple shares.

B Description of $\boldsymbol{PROOF_U}$

See Section 5. In order to prove that U is well-formed the prover must convince the verifier that there exists an s such that $C_0 = g^s$ and $U = G^{s+v}$ with $v \in \{0,1\}$. The protocol runs as follows:

1. The prover sets $a_v = g^w$ and $b_v = G^w$ for random $w \in_R \mathbb{Z}_q$. The prover also sets $a_{1-v} = g^{r_{1-v}} C_0^{d_{1-v}}$ and $b_{1-v} = G^{r_{1-v}} (U/G^{1-v})^{d_{1-v}}$, for random $r_{1-v}, d_{1-v} \in_R \mathbb{Z}_q$. The prover sends a_0, b_0, a_1, b_1 in this order to the verifier.
2. The verifier sends a random challenge $c \in_R \mathbb{Z}_q$ to the prover.
3. The prover sets $d_v = c - d_{1-v} \pmod q$ and $r_v = w - s d_v \pmod q$, and sends d_0, r_0, d_1, r_1 in this order to the verifier.
4. The verifier checks that $c = d_0 + d_1 \pmod q$ and that $a_0 = g^{r_0} C_0^{d_0}$, $b_0 = G^{r_0} U^{d_0}$, $a_1 = g^{r_1} C_0^{d_1}$, and $b_1 = G^{r_1} (U/G)^{d_1}$.

Clearly, the proof is honest verifier zero-knowledge, hence its non-interactive version releases no information on v in the random oracle model.