Verifiable Signature Sharing

Matthew K. Franklin Michael K. Reiter

AT&T Bell Laboratories, Holmdel, New Jersey, USA
franklin,reiter@research.att.com

Abstract. We introduce Verifiable Signature Sharing (VΣS), a cryptographic primitive for protecting digital signatures. VΣS enables the holder of a digitally signed document, who may or may not be the original signer, to share the signature among a set of proxies so that the honest proxies can later reconstruct it. We present efficient VΣS schemes for exponentiation based signatures (e.g., RSA, Rabin) and discrete log based signatures (e.g., ElGamal, Schnorr, DSA) that can tolerate the malicious (Byzantine) failure of the sharer and a constant fraction of the proxies. We also describe our implementation of these schemes and evaluate their performance. Among the applications of VΣS is the incorporation of digital cash into multiparty protocols, e.g., to enable cash escrow and secure distributed auctions.

1 Introduction

In this paper we introduce a new cryptographic primitive for protecting digital signatures, called Verifiable Signature Sharing (VΣS). VΣS enables the holder of a digitally signed document, who may or may not be the original signer, to *share* the signature among a set of *proxies* so that the honest proxies can later *reconstruct* it. At the end of the sharing phase, each proxy can verify whether a valid signature for the document can be reconstructed, even if the original signature holder and/or some proxies are malicious. In addition, malicious proxies gain no information about the signature held by an honest sharer prior to reconstruction (but do see the document itself).

While VΣS can be solved in theory using known cryptographic techniques, the approach taken here is to focus on *practical* solutions to this problem. In fact, our study of VΣS was motivated by an effort to implement common financial trading vehicles in computer systems. For instance, VΣS provides an elegant way to escrow digital cash [5, 7]: by verifiably sharing the bank's signature of a bank note, the moneyholder can escrow (in the true sense) the monetary value of the note among the proxies. That is, each proxy can verify that the proxies collectively possess the cash, but the cooperation of sufficiently many proxies is required to spend it. This ability, in turn, facilitates secure distributed financial services, such as sealed-bid auctions. By having bidders escrow cash bids with a set of auction servers, the winner's payment can be guaranteed while the losers' payments are protected. A separate paper reports on the implementation of an auction service using these techniques [14].

More generally, VΣS has applications whenever a signed document should become valid only under certain conditions (e.g., a will, a "springing power of attorney" [17], or an exchange of contracts). Verifiably sharing the document's signature, with "trigger" instructions given to all proxies, ensures that the signature will not be released until the honest proxies believe that the triggering events have occurred.

We have developed VΣS schemes for many digital signature schemes, including RSA [24] (with public exponent 3), Rabin [22], ElGamal [12], Schnorr [25], and the Digital Signature Standard [11]. Our protocols are simple and efficient. Sharing requires a single broadcast from the signature holder to the proxies, followed by a single round of broadcasts among the proxies. Reconstruction requires no interaction, beyond a single message sent from each proxy to the reconstructor. Our protocols can tolerate a malicious sharer and a constant fraction of malicious proxies. Almost all of our protocols ensure the secrecy of the signature in a strong sense (related to simulatability), but we consider a weaker, heuristic notion of secrecy as well. Some of our protocols exploit precomputation by the proxies to improve performance. We summarize our results in Table 1, which includes performance data in milliseconds for our protocols on a network of SPARCstation 10s, for 512-bit moduli and the minimum number of proxies that can tolerate one faulty proxy.

scheme	faults	precomp?	secrecy	test size	share time	recon time
RSA ($e = 3$)	$\lfloor (n-1)/5 \rfloor$	yes	strong	$n = 6$	330 ms	6 ms
Rabin	$\lfloor (n-1)/4 \rfloor$	yes	strong	$n = 5$	251 ms	3 ms
ElGamal	$\lfloor (n-1)/3 \rfloor$	no	strong	$n = 4$	862 ms	321 ms
Schnorr	$\lfloor (n-1)/3 \rfloor$	no	strong	$n = 4$	514 ms	117 ms
DSA	$\lfloor (n-1)/3 \rfloor$	no	weak	$n = 4$	587 ms	116 ms

Table 1. Summary of results (n = number of proxies)

The rest of this paper is structured as follows. In Section 2, we place VΣS in the context of related work. Section 3 presents our system model and definitions. VΣS for exponentiation-based signature schemes is presented in Section 4, and for discrete log-based schemes in Section 5. VΣS schemes with heuristic secrecy are discussed in Section 6. Section 7 addresses the performance of the schemes, including a discussion of the numbers in Table 1. We conclude in Section 8.

2 Related Work

In this section, we describe previous cryptographic research to which Verifiable Signature Sharing is connected, and from which we have borrowed many of our techniques.

VΣS is related to, but different from, the idea of threshold signature schemes (see survey [10]). Such schemes enable any subset of sufficiently many members of an organization to sign documents on behalf of the organization. Some of the functionality of our VΣS schemes could be achieved by the sharer distributing the ability to sign the document among the proxies using a threshold signature scheme. One important difference is that the sharer must be the original signer of the document. A second difference is that a too-large coalition gains the ability to forge any document, rather than just reconstruct those signatures that are shared. We can state this second difference another way. Threshold signature schemes—and, more generally, function sharing schemes [9]—enable the efficient computation of a secret function (signing) on a public input (the document). VΣS enables the efficient computation of a public function (signature verification) on a shared secret input (the signature).

VΣS is related to a distributed verification protocol for undeniable signatures, due to Pedersen [21]. In an undeniable signature scheme, each signed document has associated with it some secret information, distinct from the signer's private key. To verify an undeniable signature of a document requires interaction with the possessor of the corresponding secret information. Pedersen shows how the original signer of a document can distribute the ability to verify it. In this case, the signed document is public, the secret information for that signature is shared among the verifiers, and the private key of the signer is uninvolved in the verification. Some of his methods, which in turn are based on ideas from Feldman [13], are central to our VΣS schemes for ElGamal, Schnorr, and DSA.

VΣS is related to "fair public-key cryptosystems" [20]. These schemes also enable certain public predicates, related to encryption and decryption keys, to be evaluated efficiently on shared secret inputs. The shared secret input in this case is a private decryption key. The public predicate is that the decryption key bears the appropriate inverse relationship to the public key.

The notion of secure distributed computation, or "mental games" [15, 2, 6], is also closely related. That work builds on secret sharing [26, 4] and verifiable secret sharing [8, 13, 21] techniques. General and powerful "completeness theorems" for securely evaluating arbitrary boolean or arithmetic circuits yield VΣS as a special case. However, the message complexity of these general solutions (and the number of encryptions) is typically a large constant times the size of the circuit. The circuit size of known public-key signature schemes is too large for these methods to be feasible. By adapting some of the techniques of earlier schemes in novel ways, we achieve VΣS with low communication complexity.

3 Model and Definitions

A VΣS scheme is a pair of protocols (for "sharing" and "reconstructing") involving a dealer D (called the "sharer" in the Introduction), n proxies $P_1, \ldots, P_n$, and a reconstructor R. The dealer begins with a document m and a signature $\sigma(m)$ for the document, created by itself or by another. At the end of the sharing protocol, the honest proxies either *accept* or *reject*. If the honest proxies accept, then they know m (but not $\sigma(m)$) at the end of the sharing protocol, and the reconstructor will know $\sigma(m)$ (and m if desired) at the end of the reconstruction protocol. The reconstructor does not participate in the sharing protocol, and the dealer does not participate in the reconstruction protocol.

Faulty parties (proxies and/or the dealer) are assumed to be controlled by a single adversary, who can see the contents of their memories, and can cause them to deviate from the protocol in an arbitrary manner (Byzantine faults). The reconstructor is assumed to be honest throughout the reconstruction protocol. Since our reconstruction protocols are non-interactive, the only effect of a faulty reconstructor is to prevent its own success.

We assume that the dealer can send an authenticated message privately to any proxy, and that any proxy can send an authenticated message privately to the reconstructor. We also assume that the dealer or any proxy can reliably broadcast a message to the set of proxies (Byzantine Agreement [19]). More precisely, each reliable broadcast in our VΣS protocols results in the same message being delivered to all honest proxies (and thus is *terminating* in the sense of [16]), and this message is the same as the message broadcast if the sender is honest.

Finally, the dealer and the proxies are assumed to know the public key of the original signer, which is assumed to be a valid public key.

3.1 t-Resilience

We say that our VΣS scheme is t-resilient if the following conditions hold:

Completeness: If the dealer is honest, and at most t proxies are faulty, then each honest proxy will accept.

Soundness: If at most t proxies are faulty and any honest proxy accepts at the end of the sharing protocol, then every honest proxy accepts at the end of the sharing protocol and reconstruction will be successful (regardless of whether the dealer is honest or faulty).

Secrecy: Anything that can be computed by an adversary that controls up to t faulty proxies, after participating in k sharing protocols with an honest dealer, can also be computed by the adversary without participating in any sharing protocols (i.e., from the k unsigned documents and the public keys of the original signer).

We will say that our VΣS scheme is t-resilient with "heuristic secrecy" if only completeness and soundness can be proven, while heuristic evidence supports some weaker version of secrecy, e.g., that no "useful" information can be computed from what is seen by a computationally bounded adversary.

3.2 Secret Sharing

Our protocols make extensive use of the polynomial based secret sharing scheme due to Shamir [26]. A dealer wishes to share a secret s, from a finite field F, among n parties. The dealer chooses $a_1, \ldots, a_t$ from the uniform distribution on F; this is denoted $a_1, \ldots a_t \in_R F$. Let $g(x) = a_t x^t + \cdots + a_1 x + s$. Party i is given the share $(i, g(i))$, $1 \leq i \leq n$. The parties $S \subseteq \{1 \ldots n\}$, $|S| = t+1$, can recover the secret using the Lagrange formula for interpolation ($z = 0$): $g(z) = \sum_{i \in S} c_i g(i)$, where $c_i = \prod_{j \in S, j \neq i}(z-j)(i-j)^{-1}$. No subset of up to t shares yields any information about the secret.

Our VΣS schemes for exponentiation based signatures use polynomial based secret sharing over the ring Z_N, where N is the product of two large primes. The properties of secret sharing still hold in this case (e.g., see [9]). In particular, the Lagrange interpolation formula is well-defined as long as n is smaller than the prime factors of N (and thus $(i-j)^{-1} \bmod N$ always exists).

4 VΣS for Exponentiation Based Signatures

In this section, we present VΣS schemes for exponentiation based signature schemes, i.e., RSA with public exponent 3, and Rabin. The following is an informal description of how the protocol works for RSA. The signature is the cube root of the hash of a document. The dealer uses Shamir's polynomial based secret sharing scheme to share the signature among the n proxies, using some degree t polynomial. The dealer then broadcasts the corresponding points of the cube of this polynomial. The proxies convince themselves that (1) a value has been shared using a degree t polynomial; and (2) the cube of this polynomial shares the cube of the signature, i.e., the hash of the document itself. This is enough to ensure that reconstruction will be successful.

To improve the communication complexity of the sharing protocol, we will assume in this section that the proxies have performed some precomputation before participating in any sharing or reconstruction. Specifically, the ith proxy knows (only) the value at i of a large number of random degree t polynomials. One such "pre-shared" polynomial will be used per execution of the sharing protocol. The proxies can then convince themselves that the signature-sharing polynomial has degree t by adding one of the pre-shared polynomials to it, and verifying the degree of the sum. The proxies can precompute these pre-shared polynomials using standard multi-party secure computation protocols. The computation for all shared polynomials can proceed in parallel, requiring only a constant number of rounds of communication. Alternatively, a single trusted source can provide the proxies with such values, and then destroy itself.

4.1 RSA Signatures

Suppose that the signature scheme is ("hashed") RSA with encrypting exponent $e = 3$, decrypting exponent d, and modulus N. The dealer begins with $m, \sigma(m)$, where $\sigma(m) = (h(m))^d \bmod N$ for some publicly known one-way hash function h. Assume that $p_1(x), p_2(x), \ldots$ are a supply of random degree t polynomials shared among the proxies, i.e., each proxy P_i holds $p_1(i), p_2(i), \ldots$. No proxy knows anything further about these polynomials. The polynomial $p_j(x)$ will be used in the jth sharing protocol. We do not consider concurrent executions of the sharing protocol.

Sharing Protocol

1. D sends messages privately and by reliable broadcast to the proxies:
   a. D chooses $a_t, \ldots, a_1 \in_R Z_N$, and finds $f(x) = a_t x^t + \cdots + a_1 x + \sigma(m)$.
   b. $D \rightarrow P_i$ privately: $y_i = f(i) \bmod N$ (for all i, $1 \leq i \leq n$).
   c. $D \rightarrow P_1, \ldots P_n$ by reliable broadcast: $m, y_1^3 \bmod N, \ldots, y_n^3 \bmod N$.
2. Let the broadcast values (from 1c) be denoted $m, z_1, \ldots, z_n$. Assume that this is the jth sharing protocol in which the proxies are participating. Each proxy P_i computes $r_i = p_j(i) + y_i \bmod N$, and makes the following reliable broadcast to all proxies:
   a. $(r_i$, COMPLAIN$)$, if $z_i \neq y_i^3 \bmod N$.
   b. $(r_i$, ALLOW$)$, otherwise.
3. Without further communication, each proxy ACCEPTS if the following conditions are met (and REJECTS otherwise):
   a. The values $z_1, \ldots, z_n$ lie on a polynomial g of degree at most $3t$.
   b. $g(0) = h(m)$.
   c. The values $r_1, \ldots, r_n$ that were reliably broadcast (in 2) lie on a polynomial f^* of degree t, with at most t errors. (Any nonsensical response is counted as an error.)
   d. At most t proxies COMPLAINED (in 2a) or contributed an error to f^* (in 3c).

Reconstruction Protocol

1. Each proxy P_i sends to R the following information:
   a. The value y_i that was privately sent to it (in 1b of the sharing protocol);
   b. The values $z_1, \ldots, z_n$ that were reliably broadcast (in 1c of the sharing protocol);
   c. The error locations (proxy identities) in the degree t polynomial f^* computed at the end of the sharing protocol.
2. Without further communication, R reconstructs the signature as follows:
   a. R finds $z_1, \ldots, z_n$ by majority vote of the lists received in 1b of this protocol.
   b. R finds ERROR-LOCS, the majority vote of the lists of error locations received in 1c of this protocol.
   c. R discards every private share y_i (received in 1a of this protocol) such that $z_i \neq y_i^3 \bmod N$ or such that i is in ERROR-LOCS.
   d. R interpolates the remaining private shares (received in 1a of this protocol) to find a degree t polynomial (with *no* errors). The signature is taken to be the value of the resulting polynomial at zero.

Theorem 1 *This $V\Sigma S$ scheme is t-resilient whenever $n \geq 5t + 1$.*

Proof. It suffices to show that the three conditions for t-resilience are satisfied. Suppose at most t proxies are faulty.

Completeness: If the dealer is honest, then $z_1, \ldots, z_n$ lie on the polynomial $g(x) = (f(x))^3$, which is of degree at most $3t$ (since the degree of f is at most t). So, each honest proxy will recover g through interpolation. Only a faulty proxy will complain or give a bad r_i in step 2, so at most t are seen to do so by each honest proxy. Thus all honest proxies will accept.

Soundness: We claim that the reconstruction protocol will be successful, and all honest proxies will accept, whenever any honest proxy accepts at the end of the sharing protocol. Let $\hat{y}_i$ be the value sent privately by the (possibly faulty) dealer $\hat{D}$ to P_i, let $m, \hat{z}_1, \ldots, \hat{z}_n$ be the values broadcast by $\hat{D}$ to the proxies, and let $\hat{r}_i$ be the value broadcast by P_i to the other proxies. If any honest proxy accepts during the sharing protocol, then every honest proxy will accept (since the information on which they base their decisions was all received by reliable broadcast). It suffices to show that the reconstruction will be successful when all honest proxies accept.

If all honest proxies accept, then all found the same degree t polynomial through $\hat{r}_1, \ldots, \hat{r}_n$ with at most t errors. Since all honest proxies agree on the polynomial, all sent the same error locations to R, and so ERROR-LOCS will consist of exactly these locations (by majority vote). R will also recover $\hat{z}_1, \ldots, \hat{z}_n$ exactly as they were reliably broadcast by $\hat{D}$, again by majority vote from the honest proxies. Since a degree t polynomial passes through $\hat{r}_1, \ldots, \hat{r}_n$ minus points at ERROR-LOCS, a (different) degree t polynomial $\hat{f}$ passes through $\hat{y}_1, \ldots, \hat{y}_n$ minus points at ERROR-LOCS and faulty proxies (except with negligible probability). The points $\hat{y}_1, \ldots, \hat{y}_n$ minus points at complainer locations and faulty proxies have cubes that agree with $\hat{z}_1, \ldots, \hat{z}_n$.

Let $\hat{g}$ be the unique degree $3t$ polynomial that passes through $\hat{z}_1, \ldots, \hat{z}_n$. Then $\hat{f}^3$ and $\hat{g}$ agree at the points of all proxies minus complainers, error locations, and faulty proxies. Thus they have at least $n - 2t \geq 3t + 1$ points in common. Since both are degree $3t$, this means that $\hat{f}^3 = \hat{g}$. Thus it only remains to be shown that R recovers $\hat{f}$. The points from all honest proxies, minus error locations and complainers, are included in the interpolation by R, and a point from a faulty proxy is included only when it lies on $\hat{f}$ (since otherwise its cube will be inconsistent). Hence at least $t + 1$ points are included, and so $\hat{f}$ is recovered.

Secrecy: For every sharing protocol that is executed with an honest dealer, the adversary sees a random degree $3t$ polynomial g that passes through the hash of the document, a random degree t polynomial f^* (together with "allow" messages from the honest proxies), and at most t points on the cube root of g. This probability distribution can be sampled by an adversary who knows only the document that is being signed: choose f^* at random; choose t random values to be the cube roots, cube them, and then find a random degree $3t$ polynomial g that passes through the hash of the document and the cubes ($t + 1$ points). For any computation that involves participation in sharing protocols, the adversary could perform an equivalent computation by simulating its participation instead.

4.2 Rabin Signatures

Using Rabin's signature scheme, $\sigma(m) = \sqrt{h(m)} \bmod N$, where h is a one-way hash function and the factorization of N is the private key of the signer. The same scheme as for RSA signatures now works (with squares instead of cubes). Protection against t faulty proxies requires $n \geq 4t + 1$.

5 VΣS for Discrete Log Based Signatures

In this section, we present VΣS schemes for discrete log based signature schemes, illustrating our methods with ElGamal and Schnorr signature schemes.

5.1 Verifiable Discrete Log Sharing

Central to our VΣS schemes is a scheme to verifiably share the discrete log of a public value. This scheme is essentially the Verifiable Secret Sharing scheme due to Pedersen [21], although small differences arise from differences in security models. We assume that p, q, g are known to all parties before the start of the protocol, where p is a large prime, q is a large prime factor of $p-1$, and where g has order q in Z_p^*, i.e., $\langle g \rangle = \{g^1 \bmod p, \ldots, g^{p-1} \bmod p\}$ has q elements. The dealer broadcasts α and shares a value β, claiming that (1) $\alpha \in \langle g \rangle$, and (2) β is the discrete log of α modulo p with respect to the base g. The sharing protocol ends with the proxies convinced of this claim.

Sharing Protocol

1. D sends messages privately and by reliable broadcast to the proxies:
   a. D chooses $a_t, \ldots, a_1 \in_R Z_q$, and finds $f(x) = a_t x^t + \cdots + a_1 x + \beta$.
   b. $D \to P_i$ privately: $\beta_i = f(i) \bmod q$ (for all i, $1 \leq i \leq n$).
   c. $D \to P_1, \ldots, P_n$ by reliable broadcast: $\alpha, g^{a_1} \bmod p, \ldots, g^{a_t} \bmod p$.
2. Let the broadcast values (from 1c) be denoted $\alpha, u_1, \ldots, u_t$. Each proxy P_i makes the following reliable broadcast to all proxies:
   a. COMPLAIN if $g^{\beta_i} \not\equiv \alpha \prod_{j=1}^{t} (u_j)^{i^j} \bmod p$.
   b. ALLOW otherwise.
3. Without further communication, each proxy ACCEPTS if the following are true (and REJECTS otherwise):
   a. At most t proxies COMPLAINED (in 2a). (Any nonsensical response is counted as a COMPLAIN.)
   b. $\alpha^q \equiv 1 \bmod p$.

Reconstruction Protocol

1. Each proxy P_i sends to R the following information:
   a. The value β_i that was privately sent to it (in 1b of the sharing protocol);
   b. The values $\alpha, u_1, \ldots, u_t$ that were reliably broadcast (in 1c of the sharing protocol);

2. Without further communication, R finds the discrete log of α as follows:
   a. R finds $\alpha, u_1, \ldots, u_t$ by majority vote of the lists received in step 1b of this protocol.
   b. R discards every private share β_i (received in 1a of this protocol) that is inconsistent with $\alpha, u_1, \ldots, u_t$, i.e., if $g^{\beta_i} \not\equiv \alpha \prod_{j=1}^{t} (u_j)^{i^j} \bmod p$.
   c. R interpolates the remaining private shares (received in 1a of this protocol) to find a polynomial in $Z_q[x]$ of degree at most t (with *no* errors). The discrete log is taken to be the value at zero of this polynomial.

The definition of t-resilience for VΣS from Section 3.1 can be extended in the obvious way to log sharing schemes. For the proof (of soundness) of the following theorem, it will be useful to define the "g-log" of an element of Z_p^*. Choose $g_1, \ldots, g_k$ such that every element $y \in Z_p^*$ can be written as $g^z \prod_{i=1}^{k} g_i^{z_i} \bmod p$ for exactly one z, $0 \leq z \leq q-1$. Call z the "g-log" of $y \bmod p$.

Theorem 2 *This log sharing scheme is t-resilient whenever* $n \geq 3t+1$.

Proof. The conditions for t-resilience are met.

Completeness: No honest proxy complains in step 2a when the dealer is honest, and so at most t complaints are broadcast. Moreover, $\alpha^q \boxminus 1 \bmod p$ for any $\alpha \in \langle g \rangle$, and so each honest proxy accepts.

Soundness: If any honest proxy accepts at the end of the sharing protocol, then all honest proxies accept (since the decision is based solely on information sent to all proxies by reliable broadcast). It suffices to show that reconstruction will be successful whenever all honest proxies accept. Let the broadcast from step 1c of the sharing protocol (and of step 2a of the reconstruction protocol) be $\hat{\alpha}, \hat{u}_1, \ldots, \hat{u}_t$. Let the private share sent to each proxy P_i in step 1b be $\hat{\beta}_i$. Let $\hat{a}_j$ be the "g-log" of $\hat{u}_j$ modulo p for all j, $1 \leq j \leq t$, i.e., $g^{\hat{a}_j} \equiv \hat{u}_j \prod_{i=1}^{k} g_i^{z_{ji}} \bmod p$ (for some $z_{j1}, \ldots, z_{jk}$). Let $\hat{\beta}$ be the g-log of $\hat{\alpha}$ modulo p, i.e., $g^{\hat{\beta}} \equiv \hat{\alpha} \prod_{i=1}^{k} g_i^{z_i} \bmod p$ (for some $z_1, \ldots, z_k$). Let $\hat{f}(x) = \hat{a}_t x^t + \cdots + \hat{a}_1 x + \hat{\beta}$. Suppose that $(i, \hat{\beta}_i)$ passes the test in step 2b of the reconstruction protocol, and thus is used by R in its interpolation in step 2c. Then $g^{\hat{\beta}_i} \equiv \hat{\alpha} \prod_{j=1}^{t} (\hat{u}_j)^{i^j} \bmod p$, and so $g^{\hat{\beta}_i} \equiv (\prod_{i=1}^{k} g_i^{z'_i}) g^{\hat{\beta}} \prod_{j=1}^{t} g^{\hat{a}_j i^j} \bmod p$ (for some $z'_1, \ldots, z'_k$), and thus $g^{\hat{\beta}_i} \equiv (\prod_{i=1}^{k} g_i^{z'_i}) g^{\hat{f}(i)} \bmod p$. By uniqueness of g-logs, $\hat{\beta}_i \equiv \hat{f}(i) \bmod q$. Thus, in step 2c of reconstruction, R will interpolate only points that lie on $\hat{f}(x)$. Furthermore, the interpolation will include all points from honest proxies who did not complain in step 2a of the sharing protocol. Since there are at least $n-2t \geq t+1$ such points, R will successfully recover $\hat{f}(x)$, and thus find $\hat{f}(0) \bmod q$, the g-log of $\hat{\alpha}$. Since $\hat{\alpha}$ passed step 3b of the sharing protocol, $\hat{\alpha} \in \langle g \rangle$, i.e., the g-log of $\hat{\alpha}$ is in fact the discrete log of $\hat{\alpha}$.

Secrecy: For every sharing protocol that is initiated by an honest dealer, the adversary controlling t faulty proxies sees (1) the element $\alpha \in \langle g \rangle$ whose discrete log is to be shared, (2) $g^{a_1} \bmod p, \ldots, g^{a_t} \bmod p$, for random $a_1, \ldots, a_t \in_R Z_q^*$, and (3) t points on $f(x) = a_t x^t + \cdots + a_1 x + \beta \bmod p$ where β is the discrete log of

α. Suppose that the faulty proxies are $P_1, \ldots, P_t$. The probability distribution of the adversary's view of the protocol can be sampled by an adversary who knows only α, as follows. Choose the t points to be $(1, \beta_1), \ldots, (t, \beta_t)$, where $\beta_1, \ldots, \beta_t \in_R Z_q$. Let b_{ij} be the i, j entry of the inverse of the Van der Monde matrix whose i, j entry is i^j, $0 \leq i, j \leq t$. Then each $u_i = g^{a_i} \bmod p$ can be computed by $u_i = (\alpha)^{b_{i0}} \prod_{j=1}^{t} (g^{\beta_j b_{ij}}) \bmod p$.

5.2 ElGamal Signatures

Using the log sharing scheme from Section 5.1 (or simple variants), we can construct VΣS schemes for many variants of ElGamal signature [12]. For example, let the public key be g, p, y where p is a large prime with large prime factor q, g has order q in Z_p^*, and $y = g^x \bmod p$ for some x. Let the private key be x. The signature of a document m is given by $\sigma(m) = [r, s]$, where $r = g^k \bmod p$ for $k \in_R Z_q$, and where $s = k^{-1}(m - xr) \bmod q$. A signature can be publicly verified by checking that $g^m \equiv y^r r^s \bmod p$. This is a slight modification of the original scheme proposed by ElGamal (computing s modulo q instead of modulo $p - 1$).

Using the log sharing scheme from Section 5.1, we can construct a VΣS scheme for this signature scheme. The dealer D reliably broadcasts m, r to all proxies. Each proxy can now compute $g^m y^{-r} \equiv g^{m-xr} \equiv r^s \bmod p$. D now uses the scheme from Section 5.1 to verifiably share s to the proxies, i.e., to verifiably share the log of $r^s \bmod p$ with respect to the base r. Instead of step 3b of the sharing protocol, the proxies verify that $r^q \equiv 1 \bmod p$.

The reconstruction protocol for the VΣS scheme is essentially the same as the reconstruction protocol for the discrete log sharing scheme. The only modification is that each proxy also sends r to the reconstructor, who determines the actual r by majority vote.

Theorem 3 *This VΣS scheme is t-resilient whenever $n \geq 3t + 1$.*

Proof. The conditions for t-resilience are met.

Completeness: When the dealer is honest, each honest proxy accepts the log sharing protocol, and verifies that $r^q \boxminus 1 \bmod p$, and thus accepts.

Soundness: Suppose that the (possibly faulty) dealer broadcasts $m, \hat{r}$ to the proxies. If $\hat{r}^q \equiv 1 \bmod p$, then $\hat{r}$ has order q in Z_p^*, and thus the assumptions for the discrete log sharing scheme are met. The old step 3b of the sharing protocol of the discrete log scheme is unnecessary, since $\hat{r}^{\hat{s}} \boxminus g^m y^{-\hat{r}} \bmod p$ is known to have order q by construction. If the honest proxies accept the log sharing protocol, then reconstruction will successfully recover $\hat{s}$ such that $\hat{r}^{\hat{s}} \boxminus g^m y^{-\hat{r}} \bmod p$. Thus $g^m \equiv y^{\hat{r}} \hat{r}^{\hat{s}} \bmod p$, i.e., $[\hat{r}, \hat{s}]$ is a valid signature of m.

Secrecy: The adversary chooses a random $r \in \langle g \rangle$, and then simulates the log sharing protocol for $g^m y^{-r} \bmod p$, as described in part 3 of the proof of resilience for log sharing (Theorem 2).

5.3 Schnorr Signatures

In a Schnorr signature, the public key is g, p, y where p is a large prime, q is a large prime factor of $p-1$, g has order q in Z_p^*, and $y = g^x \bmod p$ for some x. The private key is x. The signature of a document m is given by $\sigma(m) = [c, z]$ where $c = h(g^r \bmod p, m)$ for random r and one-way hash function h, and where $z = cx + r \bmod q$. A signature can be publicly verified by checking that $c = h(g^z y^{-c} \bmod p, m)$.

Using the log sharing scheme from Section 5.1, we can construct a VΣS scheme for Schnorr signatures. The dealer D reliably broadcasts m, c, u to all proxies, where $u = g^z \bmod p$. D shares z with the proxies so that they are convinced they hold shares of the log of $u \bmod p$ with respect to the base g. The proxies accept if they accept the log sharing protocol and $c = h(uy^{-c} \bmod p, m)$. In the reconstruction protocol, R finds u, c by majority vote from the proxies, and reconstructs z as in the log sharing scheme.

Theorem 4 *This VΣS scheme is t-resilient whenever $n \geq 3t+1$.*

Proof. The conditions for t-resilience are met.

Completeness: When the dealer is honest, each honest proxy accepts the log sharing protocol, and sees that $c = h(uy^{-c} \bmod p, m)$.

Soundness: Suppose that the (possibly faulty) dealer broadcasts $m, \hat{u}, \hat{c}$ to all proxies. If the honest proxies accept, then (1) $\hat{c} = h(\hat{u}y^{-\hat{c}} \bmod p, m)$ and (2) R will recover $\hat{z}$ as the log of $\hat{u} \bmod p$ with respect to the base g. Thus $\hat{c} = h(g^{\hat{z}} y^{-\hat{c}} \bmod p, m)$, i.e., $[\hat{c}, \hat{z}]$ is a valid signature of m.

Secrecy: The adversary chooses a random r, finds $w = g^r \bmod p$, finds $c = h(w, m)$, and finds $u = wy^c \bmod p$. The adversary then follows the simulation from part 3 of the proof of resilience for log sharing (Theorem 2).

6 VΣS for the DSA with Heuristic Secrecy

In this section, we consider VΣS schemes for which the secrecy requirement is weakened. Previously, we have required that the adversary learns *nothing* from participating in these protocols, in the strong sense that its view could be simulated without participation. We replace this with a heuristic condition that the adversary *appears to learn nothing useful.* Of course, it is impossible to prove such an informal condition.

In the DSA, the public key is g, y, p, q where p is a large prime, q is a large prime factor of $p-1$, g is a generator of Z_q^*, and $y = g^x \bmod p$ for some private key $x \in Z_q^*$. The signature of a document m is $\sigma(m) = [r, s]$, where $r = (g^k \bmod p) \bmod q$ for some $k \in_R Z_q^*$, and where $s = k^{-1}(h(m) + xr) \bmod q$ for a one-way hash function h into Z_q^*. To verify a signature, anyone can check that $r = (g^{u_1} y^{u_2} \bmod p) \bmod q$, where $u_1 = h(m)s^{-1} \bmod q$, and where $u_2 = rs^{-1} \bmod q$.

A VΣS scheme with heuristic secrecy for DSA proceeds as follows. D reliably broadcasts m, s, α to all proxies, where $\alpha = y^{u_2} \bmod p$. D does a verifiable share of u_2 to the proxies so that they are convinced it is the log of $\alpha \bmod p$ with respect to the base y. The proxies accept if they accept the log sharing protocol, and if $y^v \equiv \alpha^s \bmod p$, where $v = (g^{u_1}\alpha \bmod p) \bmod q$. The signature will be easy to reconstruct since $r = u_2 s \bmod q$.

Theorem 5 *This $V\Sigma S$ scheme is t-resilient with heuristic secrecy whenever $n \geq 3t + 1$.*

Proof. The conditions for t-resilience with heuristic secrecy are met.

Completeness: When the dealer is honest, each honest proxy accepts the log sharing protocol. Furthermore, the test $y^v \equiv \alpha^s \bmod p$ will succeed for every honest proxy, since it compares values that were reliably broadcast by the dealer.

Soundness: Suppose that the (possibly faulty) dealer broadcasts $m, \hat{s}, \hat{\alpha}$ to the proxies. If the honest proxies accept the log sharing protocol, then reconstruction will successfully recover $\hat{u}_2$, the y-log of $\hat{\alpha} \bmod p$. R will reconstruct the signature to be $[\hat{u}_2\hat{s} \bmod q, \hat{s}]$. We claim that this is a valid signature of m: $y^{\hat{u}_2\hat{s}} \equiv \hat{\alpha}^{\hat{s}} \equiv y^{\hat{v}} \equiv y^{(g^{\hat{u}_1}\hat{\alpha} \bmod p) \bmod q} \bmod p$, and thus $\hat{u}_2\hat{s} \bmod q = (g^{\hat{u}_1}\hat{\alpha} \bmod p) \bmod q = (g^{\hat{u}_1}y^{\hat{u}_2} \bmod p) \bmod q$. Thus $[\hat{u}_2\hat{s} \bmod q, \hat{s}]$ passes the test for being a valid signature of m.

Heuristic Secrecy: An adversary controlling t faulty proxies sees m, s, α and t shares of u_2. The t shares reveal no useful information, since they come from a distribution that is uniformly random. There appears to be no way to recover r from m, s, α, y without an ability to take discrete logs, but it is difficult to prove this because of the complex dependencies among these values.

7 Performance

As described in Section 1, this work was motivated by a practical effort to experiment with a number of financial trading vehicles in real systems. We have implemented the $V\Sigma S$ schemes described here as part of this effort. Our present implementation uses the arbitrary precision arithmetic package of Cryptolib [18] and the reliable multicast protocol of Rampart [23]. This reliable multicast protocol, which incorporates timeouts into its methods of fault detection and recovery, satisfies the "reliable broadcast" portion of our communication model, under the assumption that messages from honest parties induce timeouts in other honest parties sufficiently infrequently.

A brief summary of the performance for our implementation, in the case of no failures, is shown in Table 1 (see Section 1). The tests described in Table 1 were performed among user processes on a network of moderately loaded, single processor SPARCstation 10s running SunOS 4.1.3. The moduli N (for RSA and Rabin) and p (for ElGamal, Schnorr, and DSA) were 512 bits long in these tests. The modulus q was 160 bits long for Schnorr and DSA, and 511 bits long for ElGamal. The column titled "share time" describes the mean latency in milliseconds (ms) of the sharing protocol for each $V\Sigma S$ scheme presented, beginning when the dealer initiates the protocol and ending when the proxies accept. This cost includes the computational costs incurred by the dealer and the proxies, as well as the communication latency of the $n+1$ reliable multicasts. The private messages from the dealer to the proxies are piggybacked on the dealer's reliable multicast, each encrypted so that only the intended proxy can decipher it (using symmetric key encryption in our implementation). The column titled "recon time" shows the computational latency incurred by the reconstructor in the reconstruction protocol of each VΣS scheme. The latency of the communication from the proxies to the reconstructor (and the accompanying voting at the reconstructor) is not included in these times, because for some applications that we envision, proxies may not communicate to the reconstructor simultaneously.

The experience of implementing these schemes revealed a number of opportunities to exploit concurrency and precomputation to improve the latency of the protocols. For instance, in each scheme a proxy can perform many of the tests for determining acceptance or rejection in parallel with reliably multicasting its ALLOW/COMPLAIN message, because many of these tests depend only on information multicast from the dealer. In addition, since polynomial interpolation is used heavily in these protocols, it is useful to compute the interpolation coefficients (i.e., the quantities c_i; see Section 3.2) prior to executing the protocol, if the modulus is known in advance. The numbers in Table 1 reflect the use of both of these optimizations. Moreover, the noisy interpolation steps in the RSA and Rabin schemes were optimized for the case $t = 1$; see [3] for an algorithm for arbitrary t. The column labeled "precomp?" in Table 1 refers only to the pre-sharing of polynomials in the RSA and Rabin schemes, and not to the local precomputation of interpolation coefficients in all of the implementations.

8 Conclusions

We have identified a new cryptographic primitive for protecting digital signatures, called Verifiable Signature Sharing, and have provided practical implementations of this primitive for RSA, Rabin, ElGamal, Schnorr, and DSA signatures. Our experimental data confirms that these techniques perform sufficiently well to be useful in a wide range of applications, including the integration of digital cash into secure protocols. In a separate paper [14], we describe the use of these techniques to construct a secure distributed auctioning system.

For some of our VΣS schemes, further speed-ups are possible by making small modifications to the underlying signature scheme. It is an intriguing open question to find new signature schemes that best balance the efficiencies of signature construction, signature verification, and verifiable signature sharing.

References

1. D. Beaver, S. Micali, and P. Rogaway, "The round complexity of secure protocols," ACM STOC 1990, 503–513.
2. M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," ACM STOC 1988, 1–9.

3. E. Berlekamp and L. Welch, "Error correction of algebraic block codes," U.S. Patent Number 4,633,470.
4. G. Blakely "Safeguarding cryptographic keys," AFIPS National Computer Conference 48 (1979), 313–317.
5. D. Chaum, "Security without identification: transaction systems to make big brother obsolete," CACM 28 (1985), 1030–1044.
6. D. Chaum, C. Crépeau, and I. Damgård, "Multiparty unconditionally secure protocols," ACM STOC 1988, 11–19.
7. D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," Crypto 1988, 319–327.
8. B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," IEEE FOCS 1985, 383–395.
9. A. DeSantis, Y. Desmedt, Y. Frankel, and M. Yung, "How to share a function securely," ACM STOC 1994, 522–533.
10. Y. Desmedt, "Threshold cryptography," European Transactions on Telecommunications and Related Technologies 5 (1994), 449–457.
11. NIST FIPS PUB 181, "Digital signature standard," U.S. Department of Commerce/National Institute of Standards and Technology.
12. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Trans. Information Theory IT-31 (1985), 469–472.
13. P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," IEEE FOCS 1987, 4427–437.
14. M. K. Franklin and M. K. Reiter, "The design and implementation of a secure auction service," IEEE Symposium on Security and Privacy, Oakland, CA, 1995 (to appear).
15. O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," ACM STOC 1987, 218–229.
16. V. Hadzilacos and S. Toueg, "Fault-tolerant broadcasts and related problems," In *Distributed Systems* (2nd edition), Chapter 5, Addison-Wesley, 1993.
17. J. Hoffman, "New power-of-attorney form is introduced," The New York Times, October 1, 1994.
18. J. Lacy, D. Mitchell, and W. Schell, "CryptoLib: cryptography in software," 4th USENIX Security Workshop, pp. 1–17, 1993.
19. L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," ACM TOPLAS 4 (1982), 382–401.
20. S. Micali, "Fair public-key cryptosystems," Crypto 1992, 113-138.
21. T. Pedersen, "Distributed provers with applications to undeniable signatures," Eurocrypt 1991, 221–242.
22. M. Rabin, "Digitalized signatures and public key functions as intractable as factorization," Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, Massachusetts Institute of Technology, 1979.
23. M. K. Reiter, "Secure agreement protocols: Reliable and atomic group multicast in Rampart," 2nd ACM Conf. Computer and Comm. Security, 68–80, 1994.
24. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," CACM 21 (1978), 120–126.
25. C. Schnorr, "Efficient signature generation by smart cards," J. Cryptology 4 (1991), 161–174.
26. A. Shamir, "How to share a secret," CACM 22 (1979), 612–613.