Threshold DSS Signatures without a Trusted Party

Susan K. Langford*

Department of Electrical Engineering
Stanford University
Stanford CA 94305-4055

Abstract. A t-out-of-l threshold signature scheme allows l members of a group to own shares of a private key such that any t of them can create a signature, while fewer than t cannot. Most of these schemes require a single trusted party to create the secret key and calculate the l shares. Harn [10] and Li, Hwang, and Lee [13] have devised threshold schemes based on the difficulty of solving the discrete logarithm problem which do not require such a trusted party. This paper extends that property to 2-out-of-l threshold signatures based on the Digital Signature Standard and describes two possible generalizations to a t-out-of-l scheme.

1 Introduction

A digital signature, like a handwritten signature, is used to verify the sender of a particular message. The signer is generally a single person. However, when the message is on behalf of an organization, a valid message may require the approval of several people. A common example of this policy is a large bank transaction, which requires signatures from two people. Such a policy could be implemented by having a separate digital signature for every required signer, but this solution increases the effort to verify the message linearly with the number of signers.

An alternative method is a threshold signature scheme. In a t-out-of-l threshold signature scheme, there is one public key for the group while the private key is shared among the l members of the group. Any t members can cooperate to create a digital signature without revealing their shares of the private key. Fewer than t members cannot create a valid signature.

Threshold signatures are closely related to the concept of threshold cryptography, first introduced by Desmedt [5]. A number of group oriented systems have been proposed and, for an overview of the field, the reader is referred to [6].

Threshold signature schemes have been proposed using RSA signatures [7, 11, 9]. These schemes require the use of a trusted center to generate the private key and calculate the share values. While the existence of such a center is not an unreasonable assumption, there are two potential problems. First, for many

* This work was supported by the National Semiconductor Corporation under the Fellow/Mentor/Advisor [FMA] program of Stanford University Center for Integrated Systems.

D. Coppersmith (Ed.): Advances in Cryptology - CRYPTO '95, LNCS 963, pp. 397-409, 1995.

applications, there is no one person or device which can be completely trusted by all members of the group. Second, the use of a key center creates a single point failure. Any security lapse at the key center can reveal the private key. Once the shares are distributed, careless handling of any one share reveals no information about the private key. At least t shares must be compromised before an attacker has any useful information.

To avoid these problems, Harn [10] introduced a scheme based on a modified ElGamal signature which does not require a trusted key center. Li, Hwang, and Lee used Harn's scheme to allow the receiver to verify which members of the group signed a given message [13]. These schemes show that it is possible to generate threshold signatures without a trusted party, but leave open the question of how to create such schemes for other signature algorithms.

This paper extends the idea of distributed generation of the private key by proposing three threshold schemes based on the Digital Signature Standard (DSS). The first is a 2-out-of-l threshold scheme which requires a small amount of interaction between the co-signers, but does not require the use of a trusted center to generate the shares. This scheme does need a third person as combiner, but the combiner need not be any more trusted than other members of the group. In fact, the combiner may be a member of the group who is not a co-signer. The second scheme uses precomputed values to create t-out-of-l threshold signatures. The key generation center for this scheme can be distributed into three centers, any two of which must be compromised to expose the private key. The third scheme does not require the use of a semi-trusted combiner or precomputed lists. However, the scheme requires $t^2 - t + 1$ signers for t-out-of-l security. These three schemes are the first threshold schemes to be proposed using DSS.

2 Digital Signature Standard

The Digital Signature Standard (DSS) [1] is based on the difficulty of computing discrete logs. For this algorithm, we need the following parameters.

1. p = a prime modulus, L bits long, where $512 \leq L \leq 1024$, and L is a multiple of 64.
2. q = a 160-bit prime divisor of $p - 1$.
3. $g = h^{(p-1)/q} \bmod p$, where h is any integer with $0 < h < p$ such that $g \bmod p > 1$ (g is an element of order q in GF(p)).
4. x = an integer with $0 < x < q$.
5. $y = g^x \bmod p$.
6. M = a one-way hash of the message to be signed.
7. k = a random integer with $0 < k < q$.

The system parameters p, q, and g are public. The user's public key and private key are y and x, respectively. To create a signature, the user chooses a random k, computes $k^{-1} \bmod q$, and calculates

$$r = (g^{k^{-1}} \bmod p) \bmod q, \tag{1}$$

and

$$s = k(M + xr) \bmod q. \tag{2}$$

The pair (r, s) is the desired signature. A new value for k must be generated for each new signature.

Note: For this paper, k and k^{-1} are switched from the standard DSS notation. This is merely to simplify the following discussion. Since k is random, this interchange is immaterial.

To verify a signature, the recipient calculates $g^{Ms^{-1}} y^{rs^{-1}} \bmod p$ and checks that the result equals r. Since $y = g^x$, when we multiply the two values together, the exponent becomes $Ms^{-1} + xrs^{-1} = (M + xr)s^{-1} \bmod q$. This expression reduces to k^{-1} in the exponent, yielding the desired result r as in equation (1).

3 Secret Sharing of Sums and Products

Shamir [15] developed a secret sharing scheme based on polynomials over a finite field. A polynomial f of degree $t-1$ is chosen such that $f(0)$ is the secret. Every participant i, $(i \neq 0)$, in the scheme is given $f(i)$, a share of the secret. Any t of these shares can be used to calculate $f(0)$. Further, if any $t-1$ of these shares are combined, then no information is revealed about $f(0)$.

In general, from any t distinct points of a degree $t-1$ polynomial, we can calculate the value of any other point of the polynomial by interpolation. For a set π of t shareholders with shares $f(x_i)$, over the field $GF(q)$, the formula for this interpolation is

$$f(x) = \sum_{i \in \pi} f(x_i) \prod_{j \in \pi, j \neq i} \frac{(x - x_j)}{(x_i - x_j)} \bmod q. \tag{3}$$

Therefore the formula for calculating the secret $f(0)$ is the linear combination of the shares,

$$f(0) = \sum_{i \in \pi} c_{i,\pi} f(x_i) \bmod q, \tag{4}$$

where $c_{i,\pi}$ is a constant which can be publicly determined because its value does not depend on any secret information.

Although generally defined over a field, this scheme can be defined mod n, where n is not prime, if the quantities $x_i - x_j$ have inverses for all i and j. If the shareholders are numbered from 1 to l, this is equivalent to requiring that n has no prime factors less than l.

It is natural to think of secret sharing in terms of an existing secret owned by an individual or device. The owner of the secret then creates and distributes the shares. However, we can also implement secret sharing so that the secret is created as part of the process of creating the shares. The value of the secret is determined by the value of the shares, but the secret itself is not explicitly calculated at the time the shares are created. This process will be useful for creating threshold signatures, where we want to create shares of a private key.

The private key itself is never used directly, so there is no reason that it ever needs to be explicitly calculated.

Shamir's scheme can be used to implement this idea of implicitly creating the secret. Suppose that a group wishes to create shares in a secret which is the sum of two numbers, without any one person ever knowing that sum. Person A creates a secret sharing polynomial $f_A(x)$ with secret S_A (i.e., $f_A(0) = S_A$), and distributes the shares. Similarly, person B creates $f_B(x)$ with secret S_B and distributes the resulting shares. Then $f(x) = f_A(x) + f_B(x)$ is a new polynomial with secret $f(0) = f_A(0) + f_B(0) = S_A + S_B$. If shareholder i has received $f_A(i)$ and $f_B(i)$, he can create the share of the new polynomial by adding the two shares, $f(i) = f_A(i) + f_B(i)$. Clearly, this scheme is not limited to two equations, the only requirement for additional equations is that the numbering of the shareholders remains consistent. Shareholder i must receive the polynomial evaluated at point i for all polynomials involved.

If we use these polynomials in the exponent, then we create shares which multiply together, rather than add. Given a prime q and an element β of order n in $\mathrm{GF}(q)$, we can create a polynomial $f(x) \bmod n$ and define the secret as $\beta^{f(0)} \bmod q$. The secret can be recovered by exponentiating the normal linear interpolation formula. As explained above, n must not have any small prime factors or this interpolation is not possible. We will use this multiplicative sharing technique to create the shares of the private key in our threshold scheme.

The methods for creating additive or multiplicative shares without a trusted center are well known. However, generating DSS threshold signatures leads to a new problem—creating shares of an arbitrary product. We would like to be able to have every member of a group generate a number and calculate shares which combine linearly to form the product of those numbers. We have not solved this problem in general. The remainder of this section explains the solution for the case of the product of two numbers.

Suppose Alice knows secret A and Bob knows secret B. Together, they would like to create shares which can produce the secret $AB \bmod q$. However, they would like to create these shares without having any one person know the secret AB. To do this, they need the help of a third person, Charles.

1. Alice generates a random number R_A and Bob generates R_B, where $0 \leq R_A, R_B < q$.
2. Alice privately sends Charles $A - R_A \bmod q$ and privately sends Bob R_A. Bob privately sends Charles $B - R_B \bmod q$ and privately sends Alice R_B.
3. Alice calculates $S_A = AR_B - \frac{1}{2}R_AR_B \bmod q$, and Bob calculates $S_B = BR_A - \frac{1}{2}R_AR_B \bmod q$. Charles calculates $S_C = (A - R_A)(B - R_B) \bmod q$.

The sum of the three secrets, S_A, S_B, and S_C, is the desired product, AB. Notice that no single person has sufficient information to calculate AB, but that any two of the three can work together to recover the secret. Thus, Charles must be trusted exactly the same amount as Alice or Bob.

4 2-out-of-l Threshold Signature Algorithm

4.1 DSS Parameters

The 2-out-of-l threshold signature scheme places an additional requirement on the parameters of DSS. We require that $(q-1)/2$ has no prime divisors smaller than l, the number of shareholders.[2] This requirement will slow the generation of the system parameters slightly. We will also need an additional parameter, β, where β is the square of a primitive element mod q. (β has order $(q-1)/2$.)

4.2 Creating the Shares

To create the 2-out-of-l threshold signature algorithm, every shareholder needs a multiplicative share of the secret key, x. We will define $x = \beta^{f(0)} \bmod q$, where f is a secret sharing polynomial. Since β is the square of a primitive element, arithmetic in the exponent is done mod $(q-1)/2$. Therefore, we will define f mod $(q-1)/2$. All necessary inverses exist since q has been chosen such that $(q-1)/2$ has no prime factors less than l.

As described in Section 3, the generation of the secret sharing polynomial f can be accomplished cooperatively, rather than relying on a trusted center. Each member of the group generates a degree one polynomial mod $(q-1)/2$. For instance, if Alice is member number 1, she generates $f_1(z) = \gamma_1 z + \delta_1 \bmod (q-1)/2$, where γ_1 and δ_1 are random. She now generates shares of f_1 and distributes them privately to each member of the group, taking care to give each member i the share $f_1(i)$.

When every member of the group has generated and distributed shares of their polynomial, Alice has $f_1(1), f_2(1), \ldots f_l(1)$. She now adds them together. Since the polynomials are linear, the result gives her $f(1)$, where $f(z) = f_1(z) + f_2(z) + \ldots + f_l(z)$. Now everyone has a share of the polynomial f, yet no single person knows $f(0)$.

To calculate the public key, member a computes $x_a = \beta^{c_{a,\pi}f(a)} \bmod q$ and member b computes $x_b = \beta^{c_{b,\pi}f(b)} \bmod q$. The product of these two numbers is the secret key since $x_a x_b = \beta^{c_{a,\pi}f(a)+c_{b,\pi}f(b)} = \beta^{f(0)} = x \bmod q$. Members a and b can use their values of x_a and x_b in a Diffie-Hellman exchange to calculate $y = g^{x_a x_b} \bmod p$. This process can be repeated for any pair, allowing group members to check their shares without revealing them.

4.3 Creating the Signature

The signature is created using the method for creating shares of a product described in Section 3. We will be creating shares of the values k and kx. Creating a signature requires two shareholders, Alice and Bob, and a trusted combiner, Charles. Any two of these three can cooperate to recover the secret, so Charles must be trusted the same amount as any shareholder.

[2] For simplicity, we could require that $(q-1)/2$ be prime. However, this stronger requirement is not necessary.

To create a signature, Alice generates three independent random numbers, k_A, R_{A1}, and R_{A2}, uniformly between 0 and $q-1$. She then sends $g^{k_A{}^{-1}} \bmod p$, R_{A1}, and R_{A2} privately to Bob. Similarly, Bob sends $g^{k_B{}^{-1}} \bmod p$, R_{B1}, and R_{B2} privately to Alice. Alice and Bob can avoid the need for a secure channel by generating R_{A1}, R_{A2}, R_{B1}, and R_{B2} by a Diffie-Hellman exchange.

Alice and Bob can now calculate $r = (g^{k_A{}^{-1}k_B{}^{-1}} \bmod p) \bmod q$, and send the result publicly to Charles. Alice calculates $x_A = \beta^{c_{A,\pi f(A)}} \bmod q$ for her private use. Note that $x_A x_B = x \bmod q$ and $k_A k_B = k \bmod q$. So Alice can use the number k_A and the method of Section 3 to create a sharing scheme for k. Similarly, she can create a sharing scheme for kx. Let S_{A1} be Alice's share of k and S_{A2} be Alice's share of kx. Alice computes

$$s_A = S_{A1}M + S_{A2}r = (k_A - \frac{1}{2}R_{A1})R_{B1}M + (k_A x_A - \frac{1}{2}R_{A2})R_{B2}r \bmod q. \quad (5)$$

She then sends s_A, $k_A - R_{A1}$, and $k_A x_A - R_{A2}$ privately to Charles. Similarly, Bob computes

$$s_B = S_{B1}M + S_{B2}r = (k_B - \frac{1}{2}R_{B1})R_{A1}M + (k_B x_B - \frac{1}{2}R_{B2})R_{A2}r \bmod q, \quad (6)$$

and sends s_B, $k_B - R_{B1}$, and $k_B x_B - R_{B2}$ privately to Charles.

Charles now computes his shares of k and kx,

$$S_{C1} = (k_A - R_{A1})(k_B - R_{B1}) \bmod q \quad (7)$$

and

$$S_{C2} = (k_A x_A - R_{A2})(k_B x_B - R_{B2}) \bmod q. \quad (8)$$

He can now calculate $s = s_A + s_B + (S_{C1}M + S_{C2}r) \bmod q$. The result (r, s) is a valid DSS signature, since

$$\begin{aligned} s &= (S_{A1} + S_{B1} + S_{C1})M + (S_{A2} + S_{B2} + S_{C2})r & (9)\\ &= kM + kxr & (10)\\ &= k(M + xr) \bmod q. & (11) \end{aligned}$$

4.4 Maintenance of Shares

One of the benefits of a threshold scheme is that if a single share is compromised, the private key is not revealed. If the shareholders know that a share has been compromised, for security they should generate a new public key and shares of the private key. However, the shareholders may not realize that a share has been compromised, or the cost of changing the public key may be quite high. To help preserve the security of the system in these cases, we can change the shares in such a way that although the public and private keys do not change, the old shares and the new shares are incompatible. This process is similar to the disenrollment capability described in [2] and [3].

To create the new shares, one of the shareholders should distribute shares of a polynomial with a "secret" of zero (ie., $g_Z(x) = Rx + 0 \bmod (q-1)/2$,

where R is random). Each shareholder then adds the share of g_Z to the share of the private key. Adding the two polynomials does not change the secret, but does change the value of all shares. Visualizing the secret sharing polynomial as a line, we have changed the slope of the line, but not the intercept. Once the shareholders confirm that their shares are still valid, they destroy their shares of g_Z. Since the old shares and the new shares now correspond to two different polynomials, they cannot be combined to calculate the private key. The private key is only compromised if someone has access to two shares which are valid at the same time.

Any two shareholders can cooperate to replace a lost or damaged share without revealing their shares or calculating the secret. To create share i, the two shareholders use the interpolation formula (3) to calculate the value of $f(i)$. Each calculates one of the terms of this sum and adds a share of a polynomial with secret zero as described in the preceding paragraph. The result is transmitted secretly to shareholder i. Share i is the sum of the these two results. The two shareholders generating these pieces must use independent polynomials. Otherwise, shareholder i would have sufficient information to calculate the secret. Once share i is created, shares of the two polynomials with secret zero should be distributed to the remaining shareholders.

4.5 Security of 2-out-of-l Scheme

Because the partial signer, Alice (or by symmetry Bob), knows only a share of x, a share of k, and four random numbers, she has no information that could not be simulated by an external attacker. She therefore cannot break the system faster than an attacker who sees only the final DSS signature which is a result of this scheme.

The only participant with information which cannot be simulated by an attacker is the combiner. Coppersmith [4] has observed that the combiner Charles can compute quadratic residues involving x_A and x_B, but that this does not seem to lead to an attack. Essentially, this "attack" manipulates the equations Charles knows to generate a quantity which is a function of x_A and x_B and known information. Charles does not know the quantity itself, merely that it is a quadratic residue. This information does not appear to help Charles calculate x_A and x_B. For a description of the method for calculating quadratic residues, see [12].

5 t-out-of-l Threshold Signatures

To create general t-out-of-l threshold signatures, we will precompute and store information for each signature. While this approach raises non-trivial implementation problems, it may be suitable for certain applications.

5.1 With a Trusted Center

For simplicity, we will first assume the existence of a trusted center to create the shares. The next section will show how to remove this requirement. The center creates the private key x, and the public key y. For a single signature, the center generates a random k and creates two independent secret sharing schemes, one with secret k and the other with secret kx. Then the center privately gives Alice her share of k, her share of kx, and the quantity $r = (g^{k^{-1}} \bmod p) \bmod q$. Since the sharing schemes for k and kx are both mod q, and q is 160 bits long, the entire share is 480 bits. The center repeats the process for the other shareholders. Notice that, unlike the 2-out-of-l scheme, this scheme does not place any additional restrictions on the parameters of the system.

In order to sign a message, any t of the shareholders must create a partial signature. Let U_A be Alice's share of k and V_A be Alice's share of kx. Then her partial signature would be $s_A = U_A M + V_A r$. She sends this partial signature to a combiner. Once the combiner has accumulated t partial signatures for a message, he can combine them into a single signature using equation (3),

$$s = \sum_{i \in \pi} c_{i,\pi} s_i \tag{12}$$

$$= \sum_{i \in \pi} (c_{i,\pi} U_i) M + (c_{i,\pi} V_i) r \tag{13}$$

$$= kM + kxr = k(M + xr). \tag{14}$$

The interpolation constant $c_{i,\pi}$ can be calculated by the combiner, so the signers do not need to know who else is signing. Notice that unlike the method from Section 4, this method does not require trusting the combiner or keeping the partial signatures secret.

Once the share has been used to create a signature, it must never be used to create another signature. Therefore, the center will create a list of shares, with each share to be used exactly once. Since about 20,000 shares would fit on a floppy disk, the storage requirements for this are not unreasonable. However, using these lists creates a new problem. The shareholders need some system to insure that two messages are never signed using the same k. One of the simplest methods would be to have one person in charge of sending out message signing requests which include the number of the share to be used to create the signature. Each signing request would be signed by the individual. The shareholders could then keep track of what requests had been sent. If the requester cheats and requests that two different messages be signed with the same k, the shareholders' records will show this, and he will be eventually caught.

Other protocols may be possible depending on the individual application. For example, the following protocol might be acceptable for a small group of shareholders which creates signatures infrequently. Whenever a message to be signed is sent to the group, each member signs with her personal key an acknowledgement giving the message and the current k value. Group members wait to receive acknowledgements from all members before partially signing the message.

5.2 Distributing the Share Generation

To avoid the single point failure of Section 5.1, we can replace the single trusted center with three share creation centers. The three centers use the method of sharing a product described in section 3 to create additive shares of k and kx, then create t-out-of-l secret sharing schemes and distribute these shares to the shareholders. This method requires a great deal of interaction between the share creation centers at the time when they create all of the shares.

If $t > 2$ then this system puts more trust in the centers than in the shareholders since any two of the centers can compromise the private key, while two shareholders cannot reveal any information about it.

As in Section 4, we need some method for checking the validity of the shares. The shareholders can check their shares by choosing random shares within the list and computing

$$y^k = \prod_{i \in \pi} y^{c_{i,\pi} U_i} \bmod p, \tag{15}$$

a function of their shares of k, and

$$g^{xk} = \prod_{i \in \pi} g^{c_{i,\pi} V_i} \bmod p, \tag{16}$$

a function of their shares of kx. Since $y = g^x \bmod p$, if equation (15) equals equation (16), then the shares are legitimate.

5.3 Security of t-out-of-l Scheme

The security of this scheme depends critically on preventing shareholders from signing more than one message with the same k. If the protocol for determining the current k value is not secure, then two messages may be signed using the same k, revealing the private key. While the reliance on lists is clearly a disadvantage of this system, the t-out-of-l scheme does have the nice property that the partial signatures are zero-knowledge. Therefore, we can prove the following theorem.

Theorem 1. *If the shareholders never partially sign different messages with the same k value, then breaking the t-out-of-l signature scheme is equivalent to breaking DSS.*

To prove this theorem, we need to show that an external attacker, Eve, can simulate the threshold signature scheme given a standard DSS signature. Without loss of generality, we show that Eve can simulate a threshold signature scheme in which she knows the $t-1$ shares $f(1), \ldots, f(t-1)$. Eve generates $t-1$ random numbers, $U_1, \ldots, U_{t-1}$, which are shares of k. For every possible value of k, there is exactly one degree $t-1$ polynomial, $f(z)$, such that $f(i) = U_i$ and $f(0) = k$. Eve's "shares" are therefore perfectly legitimate shares of the secret k. Similarly, Eve generates $t-1$ random numbers, $V_1, \ldots, V_{t-1}$, which are shares of kx. Let $s_j = f(j)M + g(j)r \bmod q$, where f is the polynomial determined by Eve's choice of U_i and the value of k, and g is the polynomial determined

by Eve's choice of V_i and the value of kx. Eve creates $t-1$ partial signatures, $s_i = U_i M + V_i r \bmod q$, for $i = 1, \ldots, t-1$. Eve does not know $f(j)$ or $g(j)$ for $t \leq j \leq l$, but she does know that

$$s = c_{j,\pi} s_j + \sum_{i=1}^{t-1} c_{i,\pi} s_i \bmod q. \tag{17}$$

Therefore, using s, she can solve for s_j, for any j of interest.

Eve can thus simulate a threshold signature scheme in which she knows $t-1$ of the shares and sees partial signatures created by all shareholders. Therefore, she can use any attack on the threshold scheme to attack standard DSS signatures.

6 $(t^2 - t + 1)$-out-of-l Scheme

This section presents an alternate way to generalize the 2-out-of-l signature scheme. This method avoids the precomputed lists of the previous section, but requires t^2-t+1 signers to participate in a signature which has t-out-of-l security. The additional signers are required because we are multiplying secret sharing polynomials. Let $h(z) = f(z) \times g(z)$, then $h(0) = f(0) \times g(0)$, and the degree of h is equal to the degree of f plus the degree of g. For the signature algorithm, t people generate shares of k, and multiply the secret sharing polynomials. While this method is clearly impractical for large t, it may be usable for small values. For example, a scheme with 2-out-of-l security requires only three signers.

For this algorithm, we will place the same requirements on q as those of Section 4 and create the private key as described in that section. An individual signature is created by the following steps:

1. Identify t of the signers to act as dealers for k. Let π represent this group.
2. Each dealer, i, generates k_i and $k_i^{-1} \bmod q$.
3. The dealers calculate $r = (g^{k_1^{-1} k_2^{-1} \cdots k_t^{-1}} \bmod p) \bmod q$.
4. Each dealer generates and distributes shares of the following polynomials mod q:
   $\kappa_i(z)$, degree $t-1$ polynomial, $\kappa_i(0) = k_i$.
   $\lambda_i(z)$, degree $t-1$ polynomial, $\lambda_i(0) = k_i x_i$, where $x_i = \beta^{c_{i,\pi} f(0)}$.
   $\gamma_i(z)$, degree $t^2 - t$ polynomial, $\gamma_i(0) = 0$.
   $\delta_i(z)$, degree $t^2 - t$ polynomial, $\delta_i(0) = 0$.
5. Signer j creates the shares

$$U_j = [\prod_{i \in \pi} \kappa_i(j)] + [\sum_{i \in \pi} \gamma_i(j)] \bmod q, \tag{18}$$

and

$$V_j = [\prod_{i \in \pi} \lambda_i(j)] + [\sum_{i \in \pi} \delta_i(j)] \bmod q. \tag{19}$$

6. Signer j creates the partial signature

$$s_j = U_j M + V_j r, \tag{20}$$

and sends the result to the combiner. (Note: the combiner is not trusted as in Section 4.)

7. After receiving $t^2 - t + 1$ partial signatures, the combiner calculates

$$s = \sum_{j \in \rho} c_{j,\rho} s_j \bmod q, \tag{21}$$

where ρ is the set of $t^2 - t + 1$ signers and the $c_{j,\rho}$'s are the interpolation constants for a $t^2 - t$ degree polynomial.

6.1 Security of the $(t^2 - t + 1)$-out-of-l Scheme

Like the algorithm of Section 5, the partial signatures of this algorithm are zero-knowledge. Therefore, we can prove the following theorem.

Theorem 2. *If $t - 1$ or fewer of the shareholders collude, then breaking the (t^2-t+1)-out-of-l signature scheme is no more than twice as difficult as breaking a standard DSS signature with the same p, q, and g parameters.*

Again, to prove this theorem, we need to show that an external attacker, Eve, can simulate the threshold signature scheme given a standard DSS signature. Recall that the method for creating the private key, x, caused x to be a quadratic residue mod q. Since x may not be a quadratic residue for a standard DSS signature, Eve first conducts her simulation using the signature (r, s), then if the attack fails she repeats the attack using the value $(r, \alpha s)$. This value corresponds to a signature with message αM and private key αx. Let α be a primitive element mod q, so either x or αx must be a quadratic residue, and one of Eve's simulations should succeed.

Now Eve can simulate the knowledge of $t-1$ signers of the threshold scheme.[3] For $1 \leq j < t$ and $1 \in \pi$, Eve generates $\kappa_i(j)$, $\lambda_i(j)$, $\gamma_i(j)$, and $\delta_i(j)$ randomly. For every value of k_i and $k_i x_i$, and any $t - 1$ values for $\kappa_i(j)$ and $\lambda_i(j)$, there is exactly one polynomial κ_i such that $\kappa_i(0) = k_i$ and one polynomial λ_i such that $\lambda_i(0) = k_i x_i$. Eve then computes U_j and V_j according to equations (18) and (19) and calculates the partial signatures s_j.

Now she generates $t^2 - 2t + 1$ additional independent, random s_j, where $j = t, \ldots, t^2 - t$. We need to show that these are valid partial signatures for those $t^2 - 2t + 1$ signers. Let $\mathcal{X}(z)$ be a $t^2 - t$ degree polynomial with $\mathcal{X}(0) = 0$ and $\mathcal{X}(j) = \sum_{i=1}^{t} \gamma_i(j)$. Similarly, let $\mathcal{Y}(z)$ be a $t^2 - t$ degree polynomial with $\mathcal{Y}(0) = 0$ and $\mathcal{Y}(j) = \sum_{i=1}^{t} \delta_i(j)$. Rewriting the formula for the partial signature, equation (20), we have

$$s_j = ([\prod_{i \in \pi} \kappa_i(j)] + \mathcal{X}(j))M + ([\prod_{i \in \pi} \lambda(j)] + \mathcal{Y}(j))r. \tag{22}$$

[3] The proof that Eve can simulate $t - 1$ dealers is a straightforward extension of this, but is more complicated to follow.

The values of $\kappa_i(j)$ and $\lambda_i(j)$ were fixed for all j by the random numbers Eve generated. The polynomials $\mathcal{X}(z)$ and $\mathcal{Y}(z)$ each have degree $t^2 - t$, and $t^2 - t$ unknown terms. (The zeroth order term is fixed at zero.) For arbitrary s_j, $j = t, \ldots, t^2 - t$, there are $t^2 - 2t + 1$ linear equations in these unknowns created by equation (22). There are an additional $2(t - 1)$ equations from our above definitions of $\mathcal{X}(z)$ and $\mathcal{Y}(z)$. Therefore, there are a total of $t^2 - 1$ linear equations in $2t^2 - 2t$ unknowns. Since $2t^2 - 2t > t^2 - 1$ for all $t > 1$, the number of unknowns is greater than the number of equations, and there must be a solution for the system. Therefore, the random s_j's are legitimate partial signatures.

Eve now has $t^2 - t$ valid partial signatures. She can use the equation

$$s = c_{i,\rho} s_i + \sum_{j=1}^{t^2-t} c_{j,\rho} s_j, \tag{23}$$

to find any other partial signature, s_i. Therefore, she can perfectly simulate the threshold signature scheme using a standard DSS signature.

7 Conclusion

This paper describes three threshold signature schemes for the Digital Signature Standard. The 2-out-of-l scheme does not require a trusted party to create the shares or to recreate lost shares. The more general t-out-of-l scheme is less practical, requiring precomputation and storage of shares for each individual signature. The $(t^2 - t + 1)$-out-of-l scheme does not require precomputed lists, but approximately t^2 signers are required for t-out-of-l security. Both generalized schemes have provable levels of security.

Acknowledgements

The author would like to thank Jimmy Upton for suggesting this problem and Don Coppersmith and Rajeev Motwani for their helpful comments.

References

1. "A proposed federal information processing standard for digital signature standard (DSS)," *Federal Register*, August 1991.
2. B. Blakley, G. Blakley, A. Chan, and J. Massey. "Threshold schemes with disenrollment. *Advances in Cryptology–Crypto '92 proceedings*, Springer-Verlag, 1993. p. 540-8.
3. C. Charnes, J. Pieprzyk, and R. Safavi-Naini, "Conditionally secure secret sharing schemes with disenrollment capability," 2nd ACM conference on Computer and Communications Security, Fairfax, Virginia, Nov. 2-4, 1994, pp. 89-95.
4. D. Coppersmith, personal communication.
5. Y. Desmedt, "Society and group oriented cryptography," *Advances in Cryptology–Crypto '87 proceedings*, Springer-Verlag, 1988, pp. 120-127.

6. Y. Desmedt, "Threshold Cryptography," *European Transactions on Telecommunications and Related Technologies*, Vol. 5, No. 4, July-August 1994, pp. 35-43.
7. Y. Desmedt and Y. Frankel, "Shared generation of authenticators and signatures," *Advances in Cryptology–Crypto '91 proceedings*, Springer-Verlag, 1992, pp. 457-269.
8. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inform. Theory*, Vol. 31, 1985, pp. 521-539.
9. Y. Frankel and Y. Desmedt, "Parallel reliable threshold multisignature," *Tech. Report TR-92-04-02*, Dep. of EE & CS, Univ. of Wisconsin-Milwaukee, April 1992.
10. L. Harn, "Group-oriented (t, n) threshold digital signature scheme and digital multisignature," *IEE Proc.-Comput. Digit. Tech.*, Vol. 141, No. 5, September 1994, pp. 307-313.
11. C. Laih and L. Harn, "Generalized threshold cyrptosystems," *Advances in Cryptology – Asiacrypt '91 proceedings*, Springer-Verlag, 1993, pp. 159-166.
12. S. Langford, *Differential-Linear Cryptanalysis and Threshold Signatures*, Ph.D. Thesis, Stanford University, June 1995.
13. C. Li, T. Hwang, N. Lee, "(t, n)-threshold signature scheme based on discrete logarithm," *Advances in Cryptology – Eurocrypt '94 proceedings*, to appear.
14. R. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, Vol. 21, April 1978, pp. 294-299.
15. A. Shamir, "How to share a secret," *Commun. ACM*, 22:612-613, November 1979.