CS276 Cryptography **Spring 2004**

Lecture 4.14.04

Lecturer: David Wagner *Scribe: Boriska Toth*

0.1 Introduction

Today's lecture concerns *secret sharing*. Consider two officers and a president, who want a protocol for launching a nuke such that if all three submit their share of some secret the nuke is launched, but if any two submits his share the nuke isn't launched. Thus, we want to share secret x such that each person i gets x_i. We denote this

$$x \Longrightarrow (x_1, x_2, x_3)$$

In fact, we want the stronger condition that given only two shares of the secret, there is not even *partial info* that can be gained about the secret that would facilitate launching the nuke.

It turns out this simple, concrete scenario has a simple solution. Pick x_1, x_2 randomly, and set $x_3 = x \oplus x_1 \oplus x_2$. It is trivial to verify that this protocol works. Given all three shares, take the xor of all three to get x. Given less than three shares, in the case that x_1, x_2, x_3 are of the same length n, x is uniformly distributed among all n-bit strings. This situation is called "3-out-of-3 sharing".

Now we wish to add a protocol to enable any one of the three participants to unarm the nuke. That is, a secret y should be distributed among the three participants as shares y_1, y_2, and y_3 such that given any y_i, the nuke is disabled. This is called a "one-out-of-three scheme". It turns out to be trivial also: set $y = y_1 = y_2 = y_3$.

In the rest of the lecture we consider the general problem of t-out-of-n secret sharing. We have n parties, and we want to distribute secrets

$$x \Longrightarrow (x_1, ..x_n)$$

The two properties we need to guarantee are:

1) **Recoverability:** Given any t shares, we can recover x.

2) **Secrecy:** Given any $< t$ shares, absolutely nothing is learned about x. In other words, the conditional distribution given the known shares for x should be the a priori distribution for x, so $Pr\,(x|shares) = Pr\,(x)$.

We have already seen schemes for 1-out-of-n and n-out-of-n secret sharing. Now we want to deal with the remaining cases.

0.2 The Shamir Secret Sharing Scheme

One idea for 2-out-of-n sharing is that your secret is the slope of a line. Now pick two random points on the line as shares x_1, x_2. Any two people can find the slope, and thus the secret, but secrecy is also preserved as knowing one point on a line tells you nothing about its slope.

In fact, we can generalize this idea to using a quadratic function for 3-out-of-n sharing, and keep going with higher degree polynomials.

Fact: Let $\mathcal{F}$ be a field (typically finite). Then $d+1$ pairs (a_i, b_i) uniquely determine a polynomial $f(z)$ of degree $\leq d$, such that $f(a_i) = b_i$. We are assuming $d < |\mathcal{F}|$, so that the a_i's can be distinct.

In the fact above, $f(z)$ has degree $\leq d$ and not $= d$ because of degeneracy issues.

So here is *Shamir's secret sharing scheme*, for t-out-of-n secret sharing:

First, choose a large prime p, and let $\mathcal{F} = \mathcal{Z}/p\,\mathcal{Z}$. To share secret x as

$$x \Longrightarrow (x_1, \ \ldots \ x_n)$$

do the following:

1. Choose coefficients $f_1, \ \ldots \ f_{t-1} \in \mathcal{Z}/p\,\mathcal{Z}$, which are to be the coefficients of degree $t-1$ polynomial f.
2. Let $f(z) = f_0 + f_1 * z + \ \ldots \ f_{t-1} * z^{t-1}$, where $f_0 = x$.
3. Give $f(i)$ to party i, $i = 1 \ \ldots \ n$.

Now we need a recovery procedure and we need to prove the secrecy condition to show that this is a secret sharing scheme.

Recovery is straightforward. When t parties have a secret, then we have t points on the curve of a $\leq (t-1)$ degree polynomial, so by the fact above, we get unique coefficients to a degree $\leq (t-1)$ polynomial. The secret is coefficient f_0.

Formalizing, we use Lagrange Interpolation over a finite field. Given (i, x_i) for $i \in \mathcal{G}$,

$$f(z) = \sum_{i \in \mathcal{G}} x_i \prod_{j \in \mathcal{G}, j \neq i} \frac{z-j}{i-j}$$

This is a linear system in t unknowns, the coefficients f_k, with t equations. The existence of a unique solution is guaranteed by the fact stated above. So Gaussian elimination can be used to solve. Then $x = f_0 = f(0) =$

$$\sum_{i \in \mathcal{G}} x_i \prod_{j \in \mathcal{G}, j \neq i} \frac{-j}{i-j}$$

Letting $\prod_{j \in \mathcal{G}, j \neq i} \frac{-j}{i-j} = c_i$, we have that $x = \sum_{i \in \mathcal{G}} c_i x_i$. Note that c_i is a constant independent of the x_i's. Thus we can compute $c_i, i \in \mathcal{G}$ ahead of time without knowing the x_i's, and then in linear time can find x, the secret, once we have the x_i's. So recoverability is quite efficient in this case.

Now we need to verify the **secrecy** of this scheme. Suppose we have only $t-1$ parties contributing shares. This corresponds to knowing $t-1$ points of a degree $t-1$-polynomial. Can we find out coefficient f_0? Or even gain partial info? It turns out we cannot. Stating this formally, given $t-1$ shares $(i, f(i))$, and a hypothetical value x^* for the secret, to test whether the secret is x^*, we need that $x^* = f(0)$, or in other words, that point $(0, x^*)$ is another correspondence. If we just know $t-1$ points, none of which have input value 0, the conditional distribution on those points for having a point $(0, x^*)$ is still uniform. This wasn't proved in class why.

Thus all x^* values for the secret are equally likely, and secrecy holds.

So it seems we have a solution for secret sharing; we have an efficient procedure to share a secret such that secrecy and recoverability both hold. Looks like we're done, right? Actually, we shouldn't be satisfied with Shamir's scheme. Here are four problems with it we can immediately see:

1. If the participants cheat in the recover phase, the secret cannot be recovered. The other participants don't even have a way of knowing if someone cheated. There is no way to recognize for the Shamir scheme whether an alleged share is valid. Signature schemes or Advanced Coding Theory can be used to address this issue.

2. There is total trust in the dealer, and thus in any single point of failure. If the application is firing the nuke, or voting, Shamir's scheme is obviously useless, as there is no single party all parties will agree to trust.

   The dealer might hand out bogus or inconsistent shares with the Shamir scheme, and the other parties don't even know what went wrong, that the problem lies with the dealer and not with not enough participants wanting to share their secret.

   Quite surprisingly, there is actually a fix to having to trust a single dealer! We can come up with a scheme where the parties can check that the dealer has shared a secret properly and does computation correctly on the shares.

3. The scheme is one-time.

4. The scheme only allows revealing a secret, and not computing with it. An example of the relevance of this problem is the following. There is a PGP private key scheme where a private key is shared across three machines, so that three machines need to get hacked into for security to be compromised. However, it should be the case that to use the private key for computation, no one machine should ever, even temporarily, hold the entire key and represent a single point of failure. If we used the Shamir scheme to share the private key as a secret among three machines, then to decrypt a message, we would undesirably need a single machine to know the entire private key.

We will fix many of these problems with a new protocol, Verifiable Secret Sharing (VSS). We will fix the first two problems directly. The version of VSS given in this lecture is one-time. The fourth problem is a little harder to fix. A whole field in cryptography that studies "threshold cryptosystems" deals with it.

0.3 Verifiable Secret Sharing

Shamir's scheme had a nice property that we will exploit with some modifications. As Shamir's scheme is linear, in a way clarified below, it has a nice homomorphism property. Given secrets x, y that are shared:

$$x \Longrightarrow^{f} (x_1, \ \ldots \ x_n)$$

$$y \Longrightarrow^{g} (y_1, \ \ldots \ y_n)$$

The key property is that given valid shares of x, y, parties can compute valid shares of

$$x + y \Longrightarrow^{f+g} (x_1 + y_1 \ \ldots \ x_n + y_n)$$

Any party with x_i, y_i has points $f(i), g(i)$, so he can compute share $\{x+y\}_i = (i, (f+g)(i)) = (i, f(i) + g(i))$. The participants can thus compute their shares of $x + y$ on their own given their shares of x, y. So a limit type of computation (addition) can be done in a distributed fashion even in the Shamir scheme.

Now we want a scheme that works much like the Shamir scheme when the dealer is honest, but unlike the Shamir scheme, does not assume an honest dealer. Again, if $\geq t$ parties are honest, the secret should be recoverable. Ideally, the secret should not be recoverable otherwise. However, if the dealer can be malicious,

we cannot always guard the secrecy of the scheme. The dealer could simply reveal the secret, for instance. So our goals are now:

All honest parties know or detect the dealer is malicious,

OR

There is a consistent sharing of the secret with a way to recover it.

Before stating the VSS scheme, we start with some preliminaries. We must make an intractability assumption, which will be the assumed hardness of the discrete log problem. The problem is as follows. Given $g \in G$, a generator in a group, and given $h \in G$, the goal is to find $x \in \mathcal{Z}$ ($\mathcal{Z}$ denotes the integers) such that $g^x = h$, with operations defined over the group. This problem is widely believed to be hard and used as a cryptographic intractability assumption, although its hardness is unproven. Note that it has a nice trapdoor structure. Given g, it is easy to get h from x but hard to get x from h.

We now go over some notation. Let p, q be primes, with $p = 2q + 1$. $\mathcal{Z}/p\,\mathcal{Z}$ forms a group. Now we restrict our attention to quadratic cyclic residues in this group. These are elements a in a group of the from $a = w^2$, for some w in the group. Thus they are the "squares" in the group, such as the element 1. They form a cyclic subgroup of $\mathcal{Z}/p\,\mathcal{Z}$ of prime order q. We call this subgroup QR. We then pick elements $g, h \in QR$ randomly.

We define a two-tuple notation. Let:

$$\mathcal{G} = (g, h), \qquad \mathcal{A} = (a, b), \qquad \mathcal{C} = (c, d)$$

Then we can define addition, scalar multiplication, and exponentiation operations as:

$$\mathcal{A} + \mathcal{C} = (a + c, b + d), \qquad n * \mathcal{A} = (n * a, n * b), \qquad \mathcal{G}^{\mathcal{A}} = g^a * h^b$$

Furthermore, if polynomials $f(z) = f_0 + \ \ldots\ + f_d * z^d$ and $f'(z) = f'_0 + \ \ldots\ + f'_d * z^d$, then define:

$$\mathcal{F} = (f, f'), \qquad \mathcal{F}(a) = (f(a), f'(a))$$

Finally, define commitments as:

$$commit(\mathcal{A}) = \mathcal{G}^{\mathcal{A}} = g^a * h^b$$

The above represents a commitment to value a. g, h are given constants, and b is a random independent value. Now we can use a homomorphism property to observe that:

$$commit(\mathcal{A} + \mathcal{B}) = \mathcal{G}^{\mathcal{A}+\mathcal{B}} = \mathcal{G}^{(a,b)+(a',b')} =$$

$$\mathcal{G}^{(a+a',b+b')} = g^{a+a'} * h^{b+b'} = g^a * h^b * g^{a'} * h^{b'} = commit(\mathcal{A}) * commit(\mathcal{B})$$

Thus, we have a homomorphic commitment scheme. It is unusual for a commitment scheme to have this property.

We also need to prove that the *commit* function above indeed represents a commitment scheme. Two properties must hold: a scheme must be *binding*, and it must be *hiding*.

commit is **hiding**:

This means that given the value $commit(A)$, which represents a commitment on a, one has no idea about a. So we would like it that for any value of $commit(A)$, any value a' could have been the value being committed. Formally, we need given $\mathcal{A} = (a, b)$, for each a', there is a b' such that

$$g^a * h^b = g^{a'} * h^{b'}$$

This means given any a, b, a', we need a b' to fit the equation above. This happens in a group of prime order if for some d:

$$g^{a+b*d*log_g h} = g^{a'+b'*d*log_g h}$$

In a group of prime order, the equation above has a unique solution for b', given a, b, a'. So the commitment scheme is hiding.

commit is **binding**:

The whole point of using a commitment scheme is that after a commitment value $commit(\mathcal{A})$ has been advertised, participants must be sure that A is the value being used later in a scheme. So it should be hard to find other values $\mathcal{A}'$ having $commit(A) = commit(A')$. Or in other words, given a, b, it should be hard to find a', b' such that

$$g^a * h^b = g^{a'} * h^{b'}$$

Equivalently it should be hard to find a', b' with

$$a + b * d * log_g h = a' + b' * d * log_g h$$

If we could solve the above equation for a pair a', b' then we could solve the discrete log problem essentially. So we will assume our commit scheme is binding.

Finally, here is the VSS scheme, to share secret x:

$$x \Longrightarrow^{\mathcal{F}} (x_1, \ \ldots \ x_n)$$

1. The dealer chooses $\mathcal{F} = (f, f')$ randomly, where f, f' are $(t-1)$- degree polynomials, and such that $f(0) = x$, and $\mathcal{F}(0) = (x, random)$.

2. The dealer computes $\mathcal{A}_i = commit(\mathcal{F}_i)$, $i = 0..(t-1)$, and thus gets a commitment for all coefficients. He broadcasts all these t commitments $\mathcal{A}_i$ to all n participants.

3. The dealer computes $\mathcal{X}_i = \mathcal{F}(i)$ and sends this value $\mathcal{X}_i$ to participant i, for each $1 \leq i \leq n$. The dealer also signs each $\mathcal{X}_i$ value and sends the signature $sig_D(\mathcal{X}_i)$ to person i.

4. Each person P_i verifies the following:

$$\mathcal{G}^{\mathcal{X}_i} = \mathcal{G}^{\mathcal{F}(i)} = \mathcal{G}^{\mathcal{F}_0 + \ .. \ + \mathcal{F}_{t-1} i^{t-1}} = \mathcal{G}^{\mathcal{F}_0} * \mathcal{G}^{\mathcal{F}_1 i} * \ .. \ * \mathcal{G}^{\mathcal{F}_{t-1} i^{t-1}} = \mathcal{A}_0 * \mathcal{A}_1^i * \mathcal{A}_2^{i^2} * \ \ldots \ * \mathcal{A}_{t-1}^{i^{t-1}}$$

   That is, he checks if the left hand side equals the right hand side, which should be equal by the homomorphism property of the commitment scheme.

5. If this check fails for party P_i, then he broadcasts to all participants an accusation. The accusation includes his share $\mathcal{X}_i$ and the signature from the dealer $sig_D(\mathcal{X}_i)$. So far, the other participants only see that something isn't kosher, but they don't know whether the dealer or P_i is at fault. It could be that the dealer never handed P_i a valid share, or that P_i is lying.

   So the dealer, to prove things are kosher from his end, broadcasts to all participants $\mathcal{X}_i$, so that the participants can check that the share the dealer now claims to have sent is a valid share.

   Now each person P_i aborts if he sees at least t such accusations, or if his share doesn't pass the check in the previous step. Otherwise he accepts this as a successful sharing.

Note that we have been assuming that all channels are secure, private, authenticated signed channels.

The point of having the dealer send an $\mathcal{X}_i$ after an accusation was broadcast by person P_i was not just so the dealer can claim he is giving valid shares, but also because then if an honest participant didn't get a valid share, then the other $t-1$ honest participants can use the valid share broadcast by the dealer to recover the secret.

We have removed a lot of the trust in the dealer already, but there are still ways the dealer can upset this scheme. For instance, if instead of choosing prime numbers to construct a quadratic residue subgroup, he might pick his phone number. How do we totally remove trust in the dealer? Here is a simple idea:

Suppose we have two instances of running the VSS scheme with the same participants:

$$x \Longrightarrow^{\mathcal{F}} (.. \mathcal{X}_i ..), \text{ with commitments } [\mathcal{A}_0 \ ... \ \mathcal{A}_{t-1}], \text{ and}$$

$$y \Longrightarrow^{\mathcal{E}} (.. \mathcal{Y}_i ..), \text{ with commitments } [\mathcal{B}_0 \ ... \ \mathcal{B}_{t-1}]$$

We can now combine the two procedures as follows. Think of the polynomial being used as $\mathcal{E}+\mathcal{F}$, then the secret is $\mathcal{F}_0+\mathcal{E}_0 = x+y$. Each person now instead of receiving $\mathcal{X}_i = \mathcal{F}(i)$ or $\mathcal{Y}_i = \mathcal{E}(i)$, receives the sum $\mathcal{X}_i+\mathcal{Y}_i = \mathcal{F}(i)+\mathcal{E}(i)$. Finally by the homomorphism property of the commitment scheme, the commitments now look like $commit(\mathcal{F}_i+\mathcal{E}_i) = \mathcal{G}^{\mathcal{F}_i+\mathcal{E}_i} = commit(\mathcal{F}_i) * commit(\mathcal{E}_i)$. So we have a procedure for sharing with characteristics:

$$x+y \Longrightarrow^{\mathcal{E}+\mathcal{F}} (.. \mathcal{X}_i+\mathcal{Y}_i ..), \text{ with commitments } [\mathcal{A}_0 * \mathcal{B}_0, \ ... \ \mathcal{A}_{t-1} * \mathcal{B}_{t-1}]$$

Finally, assume each participant acts as a dealer and picks a function $\mathcal{F}[i], 1 \le i \le n$, and the principle of adding functions to have a secret that's the sum of secrets shown above is applied n times:

$$x[1]+x[2]+ \ ... \ x[n] \Longrightarrow^{\mathcal{F}[1]+ \ ... \ +\mathcal{F}[n]} (, ... \ \mathcal{X}[1]_i + \mathcal{X}[2]_i + ... + \mathcal{X}[n]_i, \ ...),$$

$$\text{with commitments } [\mathcal{A}[1]_0 * \mathcal{A}[2]_0 * ... * \mathcal{A}[n]_0, \ ..... \ \mathcal{A}[1]_{t-1} * \mathcal{A}[2]_{t-1} * ... * \mathcal{A}[n]_{t-1}]$$

If any one of the participants if playing the role of the dealer in an honest, unbiased way, the above scheme with all players acting as a dealer in parallel will be uniform and unbiasable by the other players.

The one issue using the above scheme with all players picking a function $\mathcal{F}[i]$ is that all players must broadcast their commitments $\mathcal{F}[i]_j, \ 1 \le i \le n, \ 0 \le j \le t-1$ as soon as they pick a function and before any players are given their shares, because otherwise, the players can bias the scheme by choosing their functions based on other players' functions.