G22.3033-013 Exposure-Resilient Cryptography 17 January 2007

Lecture 1

Lecturer: Yevgeniy Dodis *Scribe: Marisa Debowsky*

1 Introduction

The issue at hand in this course is key exposure: there's a secret key to be protected, and somebody breaks in and tries to steal it. Today, we'll be looking at **secret sharing schemes** and their applications to **Threshold Cryptography**. That is to say: we take a secret key, split it into many pieces, and then let many servers share the secret. Collectively, the servers can simulate whatever one key holder can do, but if a few severs are compromised, the secret remains safe. It's a technique that has existed since the late 80's. We'll study the general case (with n servers) and also consider the special case when the number of servers is two: the "client-server model."

In this lecture, we will define secret sharing schemes and introduce the privacy, fault-tolerance, robustness and soundness thresholds. We will give two primary examples: additive secret sharing, and Shamir's secret sharing. We will then generalize Shamir's scheme to difference privacy and fault-tolerance thresholds. We give a lower bound on share size in i.t. secure schemes: $|share|(t_f - t_p) \geq |M|$. We consider one important special case: information-dispersal schemes (for which $t_p = 0$). Finally, we give an example which breaks the Shannon lower bound: Krawczyk's computational secret sharing scheme.

2 Secret Sharing Schemes (SSS)

This scheme is a main tool in building Threshold Cryptosystems and other objects: secret sharing is a pretty big field, and we'll spend several lectures describing it.

The basic Secret Sharing Scheme is given by two algorithms: sharing (Share) and recovery (Rec). It operates in the way you'd expect: Share takes a message M and split it into pieces. Since M is secret, Share must introduce randomness (that is, Share is probabilistic); we'll use an arrow ($\rightarrow$) to indicate randomness. Rec is a deterministic algorithm which recreates the message from some or all of the shares.

- The Sharing Algorithm: $\mathsf{Share}(M) \rightarrow (S_1, S_2, \ldots, S_n, \mathsf{pub})$. The secrets $S_1, \ldots, S_n$ are distributed securely among servers 1 through n, and pub is a public share. (We include pub for the sake of generality but observe that it is often empty. If pub is present, we assume that it is authenticated, so no one can change it: it's just published in the sky.)
- The Recovery Algorithm: $\mathsf{Rec}(S'_1, S'_2, \ldots, S'_n, \mathsf{pub}) = M'$. The correctness property of the algorithm says that for any message M, $\mathsf{Rec}(\mathsf{Share}(M)) = M$.

Notice that $S'_i \in \{\text{domain of shares}\} \cup \{\perp\}$ and $M' \in \{\text{domain of messages}\} \cup \{\perp\}$.

Let's introduce four threshold parameters which quantify the security of the scheme.

t_p – **the privacy threshold.** This describes the maximum number of servers that cannot determine the secret, even if they combine their shares.

t_f – **the fault-tolerance threshold.** Suppose every server is honest; this is the minimum number of servers you need to recover the secret (if the other servers are absent).

t_r – **the robustness threshold.** Arguably the most important: if some servers are compromised, this is the minimum number of correct shares you need to recover the secret (if the other servers lie about their shares).

t_s – **the soundness threshold.** This determines the minimum number of correct shares such that you don't ever recover the wrong secret. (That is, it is impossible to recover $M' \notin \{M, \perp\}$.)

We observe the following constraints off the bat: $t_p + 1 \leq t_f \leq t_r \leq n$ and $t_s \leq t_r$. We will give examples later in which the bounds are shown to be tight.

In any threshold encryption scheme, a message is encrypted such that t_f or more servers will decrypt the message, but t_p or fewer will not. We say a scheme requires **t-out-of-n** users to decrypt when $t = t_f$ and $t_p = t - 1$.

3 The Adversary: Static vs. Adaptive

An important consideration is the strength of the adversary. A **static** adversary will choose up to t_p servers to attack and then attack them. An **adaptive** adversary will corrupt one share, look at the result and then say, "Aha! I see Server A's share was 17. Based on that information, I'll corrupt Server F next." Such an adversary is more difficult to thwart. We'll start by focusing on the static adversary.

In the static adversary model, the **security condition** of a secret sharing scheme is that the share of one message is indistinguishable from another. For any two secrets M_0 and M_1 and for any $i_1, i_2, \ldots, i_{t_p}$, we have

$$(S_{i_1}, \ldots, S_{i_{t_p}}, \mathsf{pub})(M_0) \approx (S_{i_1}, \ldots, S_{i_{t_p}}, \mathsf{pub})(M_1).$$

4 Reconstruction

Another consideration is who is decoding the message – is it one of the servers, or an outside party? A server would have a strong advantage in decoding, namely knowing at least one correct share. We will distinguish between the two types of reconstruction as we design schemes.

5 Privacy: Information Theoretic vs. Computational

A cryptosystem is **information-theoretically secure** if it is secure even when the adversary is computationally unbounded. Although this is the strongest possible privacy condition we could ask for, it comes with obvious limitations: for example, we'll see in t-out-of-n schemes that each share of the secret must be at least as large as the secret itself. On the other hand, a system is **computationally secure** if it is secure against a computationally bounded adversary; such schemes may rely on the hardness of mathematical problems (for example, the privacy of a computationally secure share may rest on the assumption that factoring is hard).

Shortly, we will illustrate a special case of information-theoretic privacy called perfect privacy: a sharing algorithm Share is **perfectly private** if each S_i provides no information about the message M without knowledge of the key. It turns out that when a scheme is perfect, there will be little distinction between static and adaptable adversaries, making perfect privacy a highly desirable property.

In considering the merits of different schemes, we weigh the following:

- Is it computational secure? Statistically secure? Perfectly secure?
- How much space does it require: what is the size of $|S_i|$ compared to $|M|$? And the size of $|\mathsf{pub}|$ compared to $|M|$?

6 Examples

6.1 n-out-of-n Schemes

First, let's consider 2-out-of-2 sharing: that is, a secret is shared between two servers, and both correct shares must be present to recover the secret. Suppose $M \in G$ where G is a finite abelian group under addition. (For example, G might be the set $\{0,1\}^N$ under addition.) Define the sharing algorithm to be the following:

$$\begin{aligned}\mathsf{Share}(M) \quad : \quad & S_1 \overset{R}{\leftarrow} G \\ & S_2 = M - S_1\end{aligned}$$

Then the recovery algorithm is $\mathsf{Rec}(S_1, S_2) = S_1 + S_2$.

In this scheme, we have $t_p = 1$ (because S_1 is random, and therefore $S_2 = M - S_1$ is also random). Therefore, the scheme satisfies the security condition. The other three thresholds are $t_f = t_r = t_s = 2 = n$.

An immediate generalization of the above is the n-out-of-n scheme. Select $n-1$ shares randomly:

$$\begin{aligned}\mathsf{Share}(M) \quad : \quad & S_1, S_2, \ldots, S_{n-1} \leftarrow_R G \\ & S_n = M - (S_1 + S_2 + \cdots + S_{n-1})\end{aligned}$$

(It's clear that this scheme is n-out-of-n, since any $n-1$ shares are random, so all n shares are required to recover the secret.) As before, the recovery algorithm is then

$$\mathsf{Rec}(S'_1, \ldots, S'_n) = S'_1 + \cdots + S'_n.$$

6.2 Shamir's Secret Sharing Scheme

Assume $\mathbb{F}$ is a finite field with $|\mathbb{F}| = q$; we will need to assume that $q > n$. Let $M \in \mathbb{F}$. Let t denote t_f; we will see in this scheme that $t_p = t - 1$, so the relation between t_f and t_p will be optimal.

Choose n distinct non-zero points, $z_1, \ldots, z_n \in \mathbb{F}$. (Since $q > n$, we have $|\mathbb{F}| \geq n+1$, so it is indeed possible to find this many non-zero points.) Define the sharing algorithm as follows:

```
Share(M) : Choose $a_1, \ldots, a_{t-1} \leftarrow_R \mathbb{F}$.
           Define a polynomial $P_M(x) = M + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$.
           Let $s_i = P_M(z_i)$, $\mathsf{pub} = \emptyset$. These are your shares.
```

Before we describe the recovery algorithm, let's make a few observations about Share. First, $|S_i| = |M|$ and $|\mathsf{pub}| = 0$. (We will see examples later in which $|S_i|$ is much smaller – in fact, as small as $|M|$ divided by the number of servers.) Second, notice that $P_M(0) = M$: this shows us why $z_i \neq 0$, since having $z_i = 0$ would break the privacy condition.

The recovery algorithm will rely on the fact that any polynomial P of degree at most $t-1$ is uniquely determined by its values at any t points. (Think about small examples to convince yourself of this fact: two points determine a line, three points determine a parabola.) In fact, let us show that for any points $x_0, \ldots, x_{t-1}$, there exists a bijection $(a_0, \ldots, a_{t-1}) \leftrightarrow (p(x_0), \ldots, p(x_{t-1}))$ and this bijection is efficiently computable.

Proof. One direction is easy: given coefficients $(a_0, \ldots, a_{t-1})$, simply write down the polynomial $P(x) = a_0 + a_1 x + \cdots + a_{t-1} x^{t-1}$, evaluate it at points $x_0, \ldots, x_{t-1}$, and write down your bijection.

Conversely, suppose you are given points $x_0, \ldots, x_{t-1}$ and their evaluations under some unknown polynomial $P(x)$. For convenience, let y_i denote $P(x_i)$. We can generate the coefficients of P using Lagrange Interpolation. Define

$$P(x) = \sum_{i=0}^{t-1} y_i \prod_{\substack{0 \leq j \leq t-1 \\ j \neq i}} \frac{x - x_j}{x_i - x_j}.$$

In the product, $i \neq j$, so the denominators are nonzero. Each summand has degree $t-1$ and therefore $\deg(P) \leq t-1$ (so we didn't cheat and write down a polynomial of high degree). P satisfies the correctness property: $\forall i$, $P(x_i) = y_i$.

To show uniqueness, suppose there were polynomials p and p' with $p(x_i) = p'(x_i)$ for each i; then we would have $(p - p')(x_i) = 0$. Since $p - p'$ is a polynomial of degree $t-1$ with t roots, it is identically zero: that is, $p = p'$. Therefore, the polynomial P we constructed above is the unique polynomial with $P(x_i) = y_i$, as desired. $\square$

To complete the scheme, let's write down the recovery algorithm. First, we claim $t_f = t$; next, assume that all the shares $S_1, \ldots, S_n$ are either correct or missing (but not adversarially set). We only need t shares to recover the secret. (Think of this as needing t points to determine the polynomial of degree $t-1$.) If $S_{i_0} = y_0, S_{i_1} = y_1, \ldots, S_{i_{t-1}} = y_{t-1}$ are present and correspond to points $x_0, \ldots, x_{t-1}$, then we recover P using Lagrange Interpolation and recover M by evaluation: $M = P(0)$.

Let us argue **perfect privacy** for this scheme. For any $t-1$ indices $i_1, \ldots, i_{t-1}$ corresponding to points $z_1, \ldots, z_{t-1}$, and for any M (namely, $P(0)$), the $t-1$ shares of M look like random numbers:

$$P(z_1), \ldots, P(z_{t-1}) \equiv R_1, \ldots, R_{t-1}.$$

Why? Because for any $R_1, \ldots, R_{t-1}$, there is a unique polynomial explaining it. Bayes's theorem tells us that $\Pr(A|B) = \frac{\Pr(B|A)\cdot\Pr(A)}{\Pr(B)}$. Therefore:

$$\begin{aligned}\Pr_P\left[P(z_1)\ldots P(z_{t-1}) = R_1 \ldots R_{t-1} | P(0) = M\right] &= \frac{\Pr\left[P(z_1)\ldots P(z_{t-1}) = R_1 \ldots R_{t-1}\right]}{\Pr\left[P(0) = M\right]} \\ &= \frac{1/|\mathbb{F}|^t}{1/|\mathbb{F}|} \\ &= \frac{1}{|\mathbb{F}|^{t-1}}\end{aligned}$$

so the distribution is indeed uniform. □

6.3 How Big Are The Shares?

Recall that in Shamir's SSS, $|S_i| = |M|$. In general, for any perfect SSS with shares of size ℓ, we have

$$\ell(t_f - t_p) \geq |M|. \tag{1}$$

For Shamir's SSS, $t_f - t_p = 1$, so this bound is tight (that is, $\ell(1) \geq |M|$, so the optimal bound is achieved). We can prove a more general theorem:

Theorem 1 *For any finite field $\mathbb{F}_q$ with $q \geq n$, there exists a perfect SSS such that for all $t_p \in [0; n - \frac{|M|}{\log q}]$, $t_f = t_p + \frac{|M|}{\log q}$, with share size $\log q$.*

Before we make an argument, let us make a few modifications to generalize Shamir's scheme. Rather than using the polynomial

$$P_M(x) = M + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1},$$

consider this alternative:

$$P_M(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-2} x^{t-2} + M \cdot x^{t-1}.$$

As before, we set $S_i = P(z_i)$, where $z_1, \ldots, z_n$ are n distinct points. We no longer need to assume that $z_i \neq 0$ for all i, as choosing $z = 0$ will not compromise the security of the secret.

For any $t-1$ shares (z_i, y_i), and for any M, we have $y_i = M \cdot z_i^{t-1} + Q(z_i)$ where Q is a polynomial of degree $t-2$. Then $Q(z_i) = y_i - M \cdot z_i^{t-1} = y_i'$. Here, Q is the only element which is random, and there exists a unique polynomial Q which satisfies this relation. This follows from the same reasoning as above: once again, for all M, $z_1, \ldots, z_{t-1}$,

$$Q(z_1), \ldots, Q(z_{t-1}) \equiv R_1, \ldots, R_{t-1}.$$

□

6.4 A Generalization of Shamir's SSS

In Shamir's original SSS, the gap between t_f and t_p was 1, but this relation between t_f and t_p was tight at the expense of having $|S_i| = |M|$ for each i. In our new scheme, we will have a larger gap between the parameters (call it $t_f - t_p = g$), but in exchange, we will have $|S_i| = |M|/g$.

We now have the proper setup to describe the generalized scheme. Assume we want to achieve the gap g, so $t = t_f = t_p + g$. We divide the message into g pieces, $M = M_1, \ldots, M_g$, and set $P_M(x) = M_1 \cdot x^{t-1} + M_2 \cdot x^{t-2} + \cdots + M_g \cdot x^{t-g} + Q(x)$, where $Q(x)$ is a random polynomial of degree $t - g - 1$ (which is $t_p - 1$). Now evaluate $S_i = P_M(z_i)$.

A few observations on this scheme:

- Correctness: given any t shares, we can recover the polynomial with Lagrange Interpolation and therefore recover the message.
- Privacy: for any t_p shares, we have $M_1 \cdot z_i^{t-1} + \cdots M_g \cdot z_i^{t-g} + Q(z_i) = y_i$ by the same argument as above.
- The size of each share is $|S_i| = \frac{|M|}{g} = \frac{|M|}{t_f - t_p}$, so this is tight. See (1).
- We still have a constraint on the size q of the field $\mathbb{F}$ relative to n; we need $|S_i| \geq \log n$. We'll later examine schemes with $|S_i| = 1$.

Remark 2 If $t_p = 0$ in this scheme, then it becomes an Information Dispersal Scheme (IDS). Under such a condition, the polynomial is simply $M_{t-1}x^{t-1} + \cdots M_1 x + M_0$ (there's no privacy!); any t shares suffice to recover M, and the size of each share is $|S_i| = |M|/t$. Information theoretically, this is optimal, and is the exact opposite of Shamir's SSS.

So far in our examples, we've discussed perfect privacy and we've assumed that all the shares are correct. Let's generalize in both directions: what about computational privacy? And what if some shares are corrupted? We'll address the first question today.

7 Computational Privacy

This scheme is due to Krawczyk. The idea is the following: choose $\alpha \leftarrow_R \{0,1\}^\lambda$, where λ is the security parameter and α is the secret key of a one-time secure symmetric encryption scheme. Set $\mathsf{pub} = Enc_\alpha(M)$. We only need the encryption to be one-time secure – say, for example, $M \oplus G(\alpha)$ where G is a pseudo-random generator (PRG). Now 'secret share' the key, α. (Use Shamir's SSS, for example, or a more generalized scheme. In applications, we'll have something on the order of len$(\alpha) = 128$, in which case Shamir's scheme will work just fine.) Now $|\mathsf{pub}| \approx |M|$. There are two directions to go from here: if we have a large public share, then all of the private shares may be short.

A trivial general technique to eliminate the public share is to include it in every local share. (This is called replication.) However, replication is wasteful, and here, it makes the local shares longer than the message, which is indeed worse that Shamir's original scheme. Instead of replication, we can use information dispersal (as described above) to get rid of pub; we don't care about privacy.

That is, we do $IDS(\mathsf{pub})$ with $t_p = 0$ (and choose any t_f). Then we will have

$$|S_i| = \frac{|\mathsf{pub}|}{t_f} + \lambda = \frac{|M|}{t_f} + \lambda.$$

This is essentially optimal. Any t_f of the shares can recover α and pub, so therefore those t_f shares can recover the message. Krawczyk's scheme is a generalization of Shamir's: we settle for computational privacy, but break the Shannon lower bound.

References

[1] A. Shamir. How to Share a Secret. In *Comm. ACM* v.22 (n.11), pages 612–613, 1979.

[2] H. Krawczyk. Secret sharing made short. In *Proc. of the 13th Annual International Cryptology Conference on Advances in Cryptology*, pages 136–146, 1994.