CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

May 24, 2022

Lecture 15b

Homomorphic encryption (continued)

Homomorphic encryption

Plan

Homomorphic encryption

Homomorphic encryption

From previous lecture:

Definition (homomorphic evaluation algorithm)

$\mathcal{HE}$ is a *homomorphic evaluation algorithm* for $\mathcal{ES} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and FC if for all functions $f \in \mathrm{FC}$ and all messages $M_1, \ldots, M_m$, where m is the number of inputs of f, the following returns true with probability 1:

For $i = 1, \ldots, m$ do $C_i \xleftarrow{\$} \mathcal{E}_{ek}(M_i)$
$C \xleftarrow{\$} \mathcal{HE}_{hk}(\langle f \rangle, C_1, \ldots, C_m)$; $M \leftarrow \mathcal{D}_{dk}(C)$
Return $(M = f(M_1, \ldots, M_m))$

That is, C is an encryption of $f(M_1, \ldots, M_m)$.

Definition (homomorphic encryption scheme)

Encryption scheme $\mathcal{ES}$ is homomorphic for the class of functions FC if there is an efficient homomorphic evaluation algorithm $\mathcal{HE}$ as above.

Plan

CTR$ is XOR-homomorphic

Let F: $\{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^L$ be a family of functions and let

$$G_K(R,n) = F_K(R+1)\|\cdots\|F_K(R+n/L)\ .$$

Then recall the CTR$ symmetric encryption scheme $\mathcal{ES} = (\mathcal{K},\mathcal{E},\mathcal{D})$ has

```
Alg $\mathcal{E}_K(M)$
$R \xleftarrow{\$} \{0,1\}^\ell$ ; $n \leftarrow |M|$
$X \leftarrow G_K(R,n) \oplus M$
Return $R\|X$
```

```
Alg $\mathcal{D}_K(R\|X)$
$n \leftarrow |X|$
$M \leftarrow G_K(R,n) \oplus X$
Return $M$
```

Say $R\|X \xleftarrow{\$} \mathcal{E}_K(M)$ and $n = |M|$. If $\Delta \in \{0,1\}^n$ and $Y \leftarrow X \oplus \Delta$ then

$$\mathcal{D}_K(R\|Y) = G_K(R,n) \oplus Y = G_K(R,n) \oplus X \oplus \Delta = M \oplus \Delta\ .$$

CTR$ is XOR-homomorphic

Let $B = \{0,1\}^L$ and let B^+ be the set of all strings whose length is a positive multiple of L.

For $\Delta \in B^+$, define $f_\Delta\colon \{0,1\}^{|\Delta|} \to \{0,1\}^{|\Delta|}$ by $f_\Delta(M) = \Delta \oplus M$. Let FC be the set of all functions f_Δ as Δ ranges over B^+. Let $\langle f_\Delta \rangle = \Delta$.

- Note that f_Δ is a one-input function.

Let $\mathcal{ES}$ be the CTR$ symmetric encryption scheme based on family of functions $F\colon \{0,1\}^k \times \{0,1\}^\ell \to B$, as above.

Then the above shows that $\mathcal{ES}$ is homomorphic for FC, with homomorphic evaluation algorithm:

```
Alg $\mathcal{HE}_\varepsilon(\Delta, C)$    // $|C| = \ell + \Delta$
$R \| X \leftarrow C$    // $R = \ell$ and $X \in B^+$
$Y \leftarrow X \oplus \Delta$ ; Return $R \| Y$
```

An asymmetric XOR-homomorphic scheme

Let $G = \langle g \rangle$ be a cyclic group of order m. Let $\mathbf{H} : \{0,1\}^* \to \{0,1\}^n$. Define $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ via

```
Alg $\mathcal{K}$
    $x \xleftarrow{\$} \mathbb{Z}_m$
    $X \leftarrow g^x$
    return $(X, x)$

Alg $\mathcal{E}_X^{\mathbf{H}}(M)$    // $|M| = n$
    $y \xleftarrow{\$} \mathbb{Z}_m$ ; $Y \leftarrow g^y$ ; $Z \leftarrow X^y$
    $W \leftarrow \mathbf{H}(Y \| Z) \oplus M$
    return $(Y, W)$

Alg $\mathcal{D}_x^{\mathbf{H}}((Y, W))$
    $Z \leftarrow Y^x$
    $M \leftarrow \mathbf{H}(Y \| Z) \oplus W$
    return $M$
```

Above, X is the encryption key and x is the decryption key.

This scheme is IND-CPA in the random oracle model if CDH is hard in G.

Say $(Y, W) \xleftarrow{\$} \mathcal{E}_X(M)$. If $\Delta \in \{0,1\}^n$ and $V \leftarrow W \oplus \Delta$ then

$$\mathcal{D}_x((Y, V)) = \mathbf{H}(Y \| Y^x) \oplus V = \mathbf{H}(Y \| Y^x) \oplus W \oplus \Delta = M \oplus \Delta \ .$$

An asymmetric XOR-homomorphic scheme

For $\Delta \in \{0,1\}^n$, define $f_\Delta\colon \{0,1\}^n \to \{0,1\}^n$ by $f_\Delta(M) = \Delta \oplus M$. Let FC be the set of all functions f_Δ as Δ ranges over $\{0,1\}^n$. Let $\langle f_\Delta \rangle = \Delta$.

Let $\mathcal{ES}$ be the above asymmetric encryption scheme.

Then the above shows that $\mathcal{ES}$ is homomorphic for FC, with homomorphic evaluation algorithm:

```
$\underline{\textbf{Alg } \mathcal{HE}^{\mathbf{H}}_X(\Delta, (Y, W))}$    // $|W| = n$
$V \leftarrow W \oplus \Delta$ ; Return $(Y, V)$
```

Security of the above XOR homomorphic schemes

Both our symmetric and our asymmetric XOR homomorphic encryption schemes, above, are IND-CPA secure.

However they are not FH-secure.

Further homomorphic encryption schemes

There are many schemes that are homomorphic for operations like addition and multiplication over various groups.

The first FHE (Fully Homomorphic Encryption) scheme was given by Gentry [Ge09]. Many further and simpler ones have been proposed. Most schemes have to deal with very large ciphertexts, though.

A representative example is [GSW13].

Encryption in these schemes is based on matrices. The schemes allow homomorphic evaluation of the addition and multiplication operations on one-bit messages, which implies homomorphic evaluation of all functions.

Security (IND-CPA) is based on the Learning with Errors (LWE) assumption.

These blog posts provide an overview of the ideas: Part 1, Part 2.

There are many libraries implementing FHE schemes.

Plan

A DLog-based homomorphic scheme

Based on HW5P1.

Let $G = \langle g \rangle$ be a group of prime order ℓ.

We let G be our message space. (IOW M must be in G below.)

```
**Alg** $\mathcal{K}$
$x \xleftarrow{\$} \mathbb{Z}_\ell$
$X \leftarrow g^x$
return $(X, x)$
```

```
**Alg** $\mathcal{E}(X, M)$
if $M \notin G$ then return $\perp$
$y \xleftarrow{\$} \mathbb{Z}_\ell$ ; $Y \leftarrow g^y$
$Z \leftarrow X^y$ ; $W \leftarrow Z \cdot M$
return $(Y, W)$
```

```
**Alg** $\mathcal{D}(x, C)$
$(Y, W) \leftarrow C$
$Z \leftarrow Y^x$
return $Z^{-1} \cdot W$
```

This is called ElGamal encryption (IND-CPA under decisional DH).

Examples of groups that we can use: (prime-order subgroups of) elliptic curves, of $\mathbb{Z}_p^*$.

ElGamal encryption is homomorphic

```
**Alg** $\mathcal{K}$
$x \xleftarrow{\$} \mathbb{Z}_\ell$
$X \leftarrow g^x$
return $(X, x)$
```

```
**Alg** $\mathcal{E}(X, M)$
if $M \notin G$ then return $\perp$
$y \xleftarrow{\$} \mathbb{Z}_\ell$ ; $Y \leftarrow g^y$
$Z \leftarrow X^y$ ; $W \leftarrow Z \cdot M$
return $(Y, W)$
```

```
**Alg** $\mathcal{D}(x, C)$
$(Y, W) \leftarrow C$
$Z \leftarrow Y^x$
return $Z^{-1} \cdot W$
```

Theorem

We have $\mathcal{D}(x, \mathcal{E}(X, M_1) \cdot \mathcal{E}(X, M_2)) = M_1 \cdot M_2$. (using component-wise $\cdot$)

Proof: We have

$$\begin{aligned} C_1 &= (g^{y_1}, X^{y_1} \cdot M_1) \\ C_2 &= (g^{y_2}, X^{y_2} \cdot M_2) \\ C_1 \cdot C_2 &= (g^{y_1} \cdot g^{y_2}, X^{y_1} \cdot M_1 \cdot X^{y_2} \cdot M_2) \\ &= (g^{y_1+y_2}, X^{y_1+y_2} \cdot M_1 \cdot M_2) \end{aligned}$$

ElGamal and homomorphism

This property is often something we do not want for cryptography (it prevents IND-CCA), but it can be useful:

Use case: voting:

- Votes are YES/NO as $\mathcal{E}(X, g)$ or $\mathcal{E}(X, 1)$.
- The product of all votes is $\mathcal{E}(X, g^{\text{number of YES}})$.
- The tally can be made on the encrypted votes, and the decryption is only done in order to release the final count.

Pitfalls?

ElGamal and homomorphism

This property is often something we do not want for cryptography (it prevents IND-CCA), but it can be useful:

Use case: voting:

- Votes are YES/NO as $\mathcal{E}(X, g)$ or $\mathcal{E}(X, 1)$.
- The product of all votes is $\mathcal{E}(X, g^{\text{number of YES}})$.
- The tally can be made on the encrypted votes, and the decryption is only done in order to release the final count.

Pitfalls?

- Decryption is $H = g^N$ with $N =$ number of YES.
  Going from H to N is hard!

ElGamal and homomorphism

This property is often something we do not want for cryptography (it prevents IND-CCA), but it can be useful:

Use case: voting:

- Votes are YES/NO as $\mathcal{E}(X, g)$ or $\mathcal{E}(X, 1)$.
- The product of all votes is $\mathcal{E}(X, g^{\text{number of YES}})$.
- The tally can be made on the encrypted votes, and the decryption is only done in order to release the final count.

Pitfalls?

- Decryption is $H = g^N$ with $N =$ number of YES.
  Going from H to N is hard!
  Not as long as N is less that the earth population.

ElGamal and homomorphism

This property is often something we do not want for cryptography (it prevents IND-CCA), but it can be useful:

Use case: voting:

- Votes are YES/NO as $\mathcal{E}(X, g)$ or $\mathcal{E}(X, 1)$.
- The product of all votes is $\mathcal{E}(X, g^{\text{number of YES}})$.
- The tally can be made on the encrypted votes, and the decryption is only done in order to release the final count.

Pitfalls?

- Decryption is $H = g^N$ with $N =$ number of YES.
  Going from H to N is hard!
  Not as long as N is less that the earth population.
- A smart voter could cast a vote which is $\mathcal{E}(X, g^{10000})$. This is best addressed with zero-knowledge proofs.

CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

May 24, 2022

Lecture 16a

Finite fields

Definition and properties

Polynomials and square roots

Subgroups of $\mathbb{F}_p^*$

Math background, to this point

So far, for asymmetric cryptography, we mentioned:

- groups (in general, not really saying what group we had in mind):

  "Let G be a group generated by g, ..."

- and we sometimes instantiated things slightly more precisely:

  "Let $G = \mathbb{Z}_p^*$ and let g be a generator of G."

  Goal: deal with more concrete, less abstract examples. Use explicit functions such as, in Playcrypt:

  - `c = MOD(MULT(a, b), p)` to compute $c = a \cdot b = (a \times b \bmod p)$.
  - `c = MOD_INV(a, p)` to compute the inverse of a in $\mathbb{Z}_p^*$.
  - `c = MOD_EXP(a, k, p)` to compute $c = a^k \bmod p$.

The most complicated mathematical structure we had to deal with is $\mathbb{Z}_N^*$, for RSA. It is complicated because N is composite.

Who needs more math?

Cryptography is a vast area. In terms of underlying math structures, sticking to $\mathbb{Z}_N^*$ can be fine, but it is often not sufficient:

- If you need to implement cryptography, you will need to learn a few extra algorithms, such as the chinese remainder theorem.
- If you need to prove things about cryptographic results, you will need to know many proof-related techniques, and a good deal of extra mathematics.
- If you need to understand the inner mechanisms of more recent cryptographic primitives, you will need more math.
- If you need to do cryptanalysis, there is no limit on the amount of math that is potentially useful (the NSA hires many very good mathematicians!)

A good computer scientist can become good at math, and vice versa!

Goal today: finite fields

We introduce a few additional mathematics, such as finite fields.

- Finite fields are simple and ubiquitous computational objects.
- Cryptography uses them a lot. In fact, we've been using finite fields without saying it aloud.
- Having finite fields available will allow us to present some interesting stuff.
  - Some cryptanalysis results.
  - Elliptic curves.

Plan

Definition and properties

Polynomials and square roots

Subgroups of $\mathbb{F}_p^*$

Finite fields

A finite field:
- is a field,
- and it is finite.

Definition (Field)

Being a field means that:

- We have a group operation (notation: $+$, and neutral element is 0).
- We have another operation (notation: $\times$ or $\cdot$).
- The $\cdot$ operations is a group operation on the non-zero elements. The neutral element is 1.
- $+$ and $\cdot$ work well together (distributivity).

The definition also mandates that $+$ is commutative, and finiteness actually implies that $\cdot$ is commutative too (this is NOT an easy result!)

What can we do in a field?

If a, b, c are elements of a field K, we can compute expressions that yield other field elements, such as:

$$a \cdot (b + c), \quad a \cdot b \cdot (c + 1), \quad a^2 \cdot c + 8b, \quad a^{123} + b^{456} + c^{789}.$$

Non-zero field elements are invertible, so we can also compute

$$\frac{a+1}{a-bc} \text{ if } a - bc \neq 0.$$

What can we do in a field?

If a, b, c are elements of a field K, we can compute expressions that yield other field elements, such as:

$$a \cdot (b + c), \quad a \cdot b \cdot (c + 1), \quad a^2 \cdot c + 8b, \quad a^{123} + b^{456} + c^{789}.$$

Non-zero field elements are invertible, so we can also compute

$$\frac{a+1}{a-bc} \text{ if } a - bc \neq 0.$$

Most importantly, we can also define polynomials:

$$a + bX + cX^2$$

is a degree two polynomial, which we can evaluate at any field element.

What does finiteness imply?

Given that we're dealing with elements in a finite set K, there can't be infinitely many of the following elements of K:

- 0 (the neutral element for $+$);
- 1 (the neutral element for $\cdot$);
- $1+1$ (we call this 2);
- $1+1+1$ (we call this 3);
- $\ldots$
- $\underbrace{1+1+\cdots+1}_{k \text{ times}}$ (we call this k);
- $\ldots$

What does finiteness imply?

There has to be an integer p such that $p = \underbrace{1 + 1 + \cdots + 1}_{p \text{ times}} = 0$ in K.

Definition (Characteristic)

That integer is called the characteristic of the field K.

Fact: Because K is a field, its characteristic is a prime number.

Examples of finite fields

The most obvious example is $\mathbb{Z}_p$, for any prime number p.

- Multiple notations: $\mathbb{Z}_p$, $\mathbb{F}_p$, $\mathbb{Z}/p\mathbb{Z}$.
- The number of elements is exactly p.
- Simplest example with $p = 2$.
- Note that when N is a composite number, $\mathbb{Z}_N$ is NOT a field.

Are all finite fields of the form $\mathbb{F}_p = \mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$?

Examples of finite fields

The most obvious example is $\mathbb{Z}_p$, for any prime number p.

- Multiple notations: $\mathbb{Z}_p$, $\mathbb{F}_p$, $\mathbb{Z}/p\mathbb{Z}$.
- The number of elements is exactly p.
- Simplest example with $p = 2$.
- Note that when N is a composite number, $\mathbb{Z}_N$ is NOT a field.

Are all finite fields of the form $\mathbb{F}_p = \mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$? No!

- There are other examples where the number of elements is a power of the characteristic, but they're NOT of the form $\mathbb{Z}_{p^k}$!!!
- Such fields appear in the internal mechanisms of AES, but we will not cover them.

Working in $\mathbb{F}_p = \mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$

Since we've been dealing with $\mathbb{Z}_p$ for a while, the only extra information that we gain is that it is actually a field.

- For many things, a field is a nice playground, and many computational things work easily.
- $\mathbb{F}_p = \mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ is finite, so there is a fixed-size unambiguous representation of elements. Computer science likes that a lot!

Should we write mod p explicitly?

The mod p operation is what brings us from $\mathbb{Z}$ to $\mathbb{F}_p = \mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$. Most often, it is implicit when it is understood that we are speaking of elements of the finite field.

- "Let $g = 17 \in \mathbb{F}_p$" or "Let $a = -1 \in \mathbb{F}_p$" are perfectly fine.
- If we speak of elements of $\mathbb{F}_{17}$, we have $0 = 17 = 34 = 51 = -17$. They are just one single element!

Should we write mod p explicitly?

The mod p operation is what brings us from $\mathbb{Z}$ to $\mathbb{F}_p = \mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$. Most often, it is implicit when it is understood that we are speaking of elements of the finite field.

- "Let $g = 17 \in \mathbb{F}_p$" or "Let $a = -1 \in \mathbb{F}_p$" are perfectly fine.
- If we speak of elements of $\mathbb{F}_{17}$, we have $0 = 17 = 34 = 51 = -17$. They are just one single element!

In computer code, however, we want unambiguous representation:

- It can make sense to have finite field elements and integers be of different types. (with an available type conversion.)
- Or we have to make consistency checks on input values.

Either way, the actual implementations of $+$, $\cdot$, etc... have to know about p and do reductions.

The deal with exponents

When a is an element of a finite field K, we can compute the element:

$$\begin{tikzcd}\text{a is in K} \arrow[r] & a^k & \text{This exponent k is NOT in K!} \arrow[l]\end{tikzcd}$$

Any exponent $k \in \mathbb{Z}$ works. However, we know that $a^{p-1} = 1$, so that we can reduce the exponent modulo $\varphi(p) = p - 1$.

Plan

Polynomials and square roots

Polynomials are perfectly usable over finite fields.

- Polynomials can be added, multiplied, composed, evaluated.
- A polynomial factors uniquely into irreducible factors;
- A polynomial of degree d has at most d roots, counting multiplicities.
- It makes sense to speak of the polynomial ring $K[X]$.

Example: $X^2 - 1$ is a degree two polynomial. There are only two square roots of 1, which are 1 and -1 in $\mathbb{F}_p$.

A very special polynomial

In the finite field $\mathbb{F}_p$, we know that:

$$\forall x \in \mathbb{F}_p^*,\ x^{p-1} = 1.$$
$$\forall x \in \mathbb{F}_p,\ x^p = x.$$

So the polynomial $X^p - X$ is a non-zero polynomial, but it evaluates to zero at all points of $\mathbb{F}_p$.

Note that if $x \neq 0$ is in $\mathbb{F}_p$, we have $(x^{(p-1)/2})^2 - 1$, which means that:

$$x^{(p-1)/2} = \pm 1 \in \mathbb{F}_p.$$

Fact: $x^{(p-1)/2}$ is called the Legendre symbol.
There is a fast algorithm to compute it.

Fact: We have $x^{(p-1)/2} = 1$ IF AND ONLY IF x is a square in $\mathbb{F}_p$.

Proof: $\square \Rightarrow 1$ is easy. In the other direction: each non-zero square $b = a^2$ has exactly two square roots a and $-a$, so there are exactly $(p-1)/2$ distinct squares in total.

Computing square roots

In $\mathbb{F}_p$, the computation of square roots is possible.

We do not give the algorithm. It is not a very expensive computation.
(Thinking of it, do you actually know how to compute square roots in $\mathbb{R}$?)

Exercise: In the special case where $p \equiv 3 \pmod 4$, show that if $x \in \mathbb{F}_p$ is a square, then $x^{(p+1)/4}$ is a square root of x.

Square roots modulo N

In contrast, computing square roots modulo a composite N is HARD!

Theorem (square roots modulo N are hard)

If there is an algorithm A that can compute arbitrary square roots modulo N, then there is an algorithm B with similar running time that can factor N with high probability.

Fact: In the other direction, if the factors of N are known, it is easy.

This dissymetry is the root of several nice cryptographic things!

- Identification protocols;
- Cryptanalysis.

Plan

$\mathbb{F}_p^* = \mathbb{Z}_p^* = (\mathbb{Z}/p\mathbb{Z})^*$

The non-zero elements of $\mathbb{F}_p$ form its multiplicative group.

$$\mathbb{F}_p^* = \mathbb{Z}_p^* = (\mathbb{Z}/p\mathbb{Z})^*.$$

Fact: Because $\mathbb{F}_p$ is a field, this group is cyclic.

A slightly annoying aspect is that $p - 1$ is never prime (for $p > 3$).

Fact: For each divisor q of $p - 1$, there is a (unique) subgroup of $\mathbb{F}_p^*$ that has order exactly q.

Prime-order subgroups

Cryptography likes prime-order subgroups.

We often let q be a large prime factor of $p-1$, and consider the subgroup generated by an element of order exactly q.

Example: HW5P2

Benefit: if h generates a subgroup of prime order q of $\mathbb{F}_p^*$, then the powers h^x of h can use reduction modulo q in the exponent, and $\mathbb{F}_q$ is a field.

- Many things are possible in the exponent (as in HW5P2).
- This is often used in signature schemes (examples: Schnorr, DSA).

CSE107: Intro to Modern Cryptography

https://cseweb.ucsd.edu/classes/sp22/cse107-a/

Emmanuel Thomé

May 24, 2022

Lecture 16b

Secret sharing

Secret sharing

Plan

Secret sharing

Secret sharing

Let $1 \leq t < n$. A (t, n)-secret-sharing scheme allows an entity, called the dealer, to split s into shares $s_1, \ldots, s_n$ such that:

- Given any $t + 1$ shares, one can recover s, but
- Given any t or less shares, one learns nothing about s.

Secret sharing is useful in its own right and also a tool in secure multi-party computation.

Secret sharing syntax

Let $1 \leq t < n$. A (t, n)-secret-sharing scheme $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$ is a pair of algorithms that operate as follows:

- $\{(i, s_i)\}_{i=1}^{n} \overset{\$}{\leftarrow} \mathcal{SH}(s)$ — the dealer runs the sharing algorithm $\mathcal{SH}$ on input a secret $s \in \{0,1\}^*$ to get a list of n shares.
- $s \leftarrow \mathcal{RE}(\, R, \{(i, s_i)\}_{i \in R}\,)$ — apply deterministic recovery algorithm $\mathcal{RE}$ to a set $R \subseteq \{1, \ldots, n\}$, and a list of shares for players in R, to obtain output $s \in \{0,1\}^* \cup \{\perp\}$.

The correctness requirement is that, for all s in the underlying message space and all $R \subseteq \{1, \ldots, n\}$ with $|R| \geq t + 1$ we have:

If $\{(i, s_i)\}_{i=1}^{n} \overset{\$}{\leftarrow} \mathcal{SH}(s)$ and $s' \leftarrow \mathcal{RE}(\, R, \{(i, s_i)\}_{i \in R}\,)$ then $s' = s$.

Secret sharing privacy

Let $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$ be a (t, n)-secret-sharing scheme.

Game $\mathrm{IND}_{\mathcal{SS}}$

```
procedure Initialize
$b \xleftarrow{\$} \{0,1\}$

procedure Finalize($b'$)
return $(b = b')$

procedure LR($T, s_0, s_1$)    // $|s_0| = |s_1|$ and $T \subseteq \{1,\ldots,n\}$ with $|T| = t$
$\{(i, s_i)\}_{i=1}^{n} \xleftarrow{\$} \mathcal{SH}(s_b)$ ; return $\{(i, s_i)\}_{i \in T}$
```

Definition (ind-advantage of a secret sharing scheme)

The ind-advantage of an adversary A is

$$\mathbf{Adv}^{\mathrm{ind}}_{\mathcal{SS}}(A) = 2 \cdot \Pr\left[\mathrm{IND}^{A}_{\mathcal{SS}} \Rightarrow \mathsf{true}\right] - 1 .$$

Secret sharing privacy

IND-privacy for a secret-sharing scheme asks that knowledge of up to t (out of n) shares, of a sharing of a secret, not yield any partial information about the secret.

We refer to T as the set of corrupted players. It must have size at most t.

We say that secret-sharing scheme $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$ is *perfect*, or has perfect privacy, if $\mathbf{Adv}^{\mathrm{ind}}_{\mathcal{SS}}(A) = 0$ for all A.

A $(n-1, n)$ perfect secret-sharing scheme

Let G be a commutative group whose group operation we denote by "$+$", the inverse operation being "-". Examples are:

- $G = \mathbb{Z}_M$ with $+$ being addition modulo M.
- $G = \{0,1\}^m$ with $+$ being bitwise XOR.

The shares $\{(i, s_i)\}_{i=1}^{n}$ of a secret $s \in G$ are chosen so that $s_1, \ldots, s_n$ are random elements of G subject to the condition that

$$s = s_1 + \cdots + s_n \ .$$

Recovery: Given $s_1, \ldots, s_n$ we can recover s via the above equation.

Privacy: Any $n-1$ of $s_1, \ldots, s_n$ are randomly and independently distributed over G, so we have perfect privacy.

A $(n-1, n)$ perfect secret-sharing scheme

Let G be a commutative group whose group operation we denote by "$+$", the inverse operation being "-".

We define secret-sharing scheme $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$ via

```
Alg $\mathcal{SH}(s)$    // $s \in G$
For $i = 1, \ldots, n-1$ do $s_i \xleftarrow{\$} G$
$s_n \leftarrow s - (s_1 + \cdots + s_{n-1})$
return $\{(i, s_i)\}_{i=1}^{n}$
```

```
Alg $\mathcal{RE}(R, \{(i, s_i)\}_{i \in R})$
$s \leftarrow 0$    // Identity element of the group
For $i \in R$ do $s \leftarrow s + s_i$
return $s$
```

This is a $(n-1, n)$ perfect secret-sharing scheme.

For correctness, we are only concerned with $R = \{1, \ldots, n\}$.

Perfect privacy is because any $n-1$ of $s_1, \ldots, s_n$ are randomly and independently distributed over G.

Shamir's (t, n) perfect secret-sharing scheme

Let F be a finite field of size at least $n+1$. Examples are:

- $\mathsf{F} = \mathbb{Z}_p$ where $p \geq n+1$ is a prime.
- $\mathsf{F} = \mathrm{GF}(2^k)$ where $2^k \geq n+1$.

Fix some distinct points $e_1, \ldots, e_n \in \mathsf{F}$.

To share secret $s \in \mathsf{F}$, Shamir's sharing algorithm $\mathcal{SH}$ picks $a_1, \ldots, a_t \overset{\$}{\leftarrow} \mathsf{F}$ and defines the degree $\leq t$ polynomial $f\colon \mathsf{F} \to \mathsf{F}$ via

$$f(x) = s + \sum_{i=1}^{t} a_i x^i \ ,$$

so that $f(0) = s$. The share of player $i \in \{1, \ldots, n\}$ is $(i, f(e_i))$.

This is a perfect (t, n)-secret sharing scheme.

The fact that a degree t polynomial is determined by any $t+1$ points on it allows recovery, which is done via the polynomial interpolation algorithm.

Privacy is because the values of f on any t distinct points are random and independent elements of F.

Background on polynomials

For $a_0, a_1, \ldots, a_t \in \mathsf{F}$, define $P_{a_0,a_1,\ldots,a_t}\colon \mathsf{F} \to \mathsf{F}$ by

$$P_{a_0,a_1,\ldots,a_t}(x) = \textstyle\sum_{i=0}^{t} a_i x^i \; .$$

For $R \subseteq \{1,\ldots,n\}$ and $i \in R$ define $\delta_{R,i}\colon \mathsf{F} \to \mathsf{F}$ by

$$\delta_{R,i}(x) = \prod_{j \in R \setminus \{i\}} \frac{x - e_j}{e_i - e_j} \; .$$

Then for all $\ell \in R$ we have

$$\delta_{R,i}(e_\ell) = \begin{cases} 1 & \text{if } \ell = i \\ 0 & \text{if } \ell \neq i \; . \end{cases}$$

Now for $S = \{\, s_i \; : \; i \in R \,\} \subseteq \mathsf{F}$ define $Q_{R,S}\colon \mathsf{F} \to \mathsf{F}$ by

$$Q_{R,S}(x) = \textstyle\sum_{i \in R} s_i \, \delta_{R,i}(x) \; .$$

Then $Q_{R,S}(e_\ell) = s_\ell$ for all $\ell \in R$.

Polynomial interpolation theorem

The polynomial interpolation theorem is the following.

Let F be a finite field of size at least $n + 1$. Fix some distinct points $e_1, \ldots, e_n \in \mathsf{F}$.

Let $a_0, a_1, \ldots, a_t \in \mathsf{F}$ and let $s_i \leftarrow P_{a_0,a_1,\ldots,a_t}(e_i)$ for $1 \leq i \leq n$.

Let $R \subseteq \{1, \ldots, n\}$ have size $|R| \geq t + 1$ and let $S = \{\, s_i \,:\, i \in R \,\}$.

Then $Q_{R,S} = P_{a_0,a_1,\ldots,a_t}$.

In particular $Q_{R,S}(0) = a_0$.

Shamir's (t, n) perfect secret sharing scheme

Let F be a finite field of size at least $n+1$. Fix some distinct points $e_1, \ldots, e_n \in \mathsf{F}$.

We define secret-sharing scheme $\mathcal{SS} = (\mathcal{SH}, \mathcal{RE})$ via

```
**Alg** $\mathcal{SH}(s)$    // $s \in \mathsf{F}$
For $i = 1, \ldots, t$ do $a_i \xleftarrow{\$} \mathsf{F}$
For $i = 1, \ldots, n$ do $s_i \leftarrow P_{s,a_1,\ldots,a_t}(e_i)$
return $\{(i, s_i)\}_{i=1}^{n}$
```

```
**Alg** $\mathcal{RE}(R, \{(i, s_i)\}_{i \in R})$
$S \leftarrow \{\, s_i \,:\, i \in R \,\}$
$s \leftarrow Q_{R,S}(0)$
return $s$
```

This is a (t, n) perfect secret-sharing scheme.

Correctness, which assumes $|R| \geq t+1$, follows from the polynomial interpolation theorem.

Perfect privacy is because any t of $s_1, \ldots, s_n$ are randomly and independently distributed over F.