MÁSTER INTERUNIVERSITARIO EN CIBERSEGURIDAD

Information Security
Lab assignment II

1 Problem Description: Verifiable Secret Sharing

A (n, t)-secret sharing scheme, for $n > t$, is a protocol such that a dealer can share a secret s among n participants in a way that any subset of $t + 1$ participants can compute the secret s, but subsets of size t or lower cannot. In many applications, the participants may benefit from a procedure for verifying the correctness of the dealt values in order to prevent malicious behavior by the dealer. The addition of verifiability to a secret sharing scheme upgrades it to a *Verifiable Secret Sharing* protocol (VSS).

A (n, t)-VSS consists of two phases: the Sharing phase and the Reconstruction phase.

- In the Shanring phase, the dealer distributes a secret s (for instance, an integer number, a key, a bitstring, a hash, etc.) among n nodes. At the end of the Sharing phase, each honest node i holds a share s_i of the distributed secret s.
- In the Reconstruction phase, each node reveals its secret share s'_i and a reconstruction function is applied in order to compute the secret s. For honest nodes $s'_i = s_i$ while for malicious nodes s'_i may be different from s_i or be absent.

In this assignment, you will build a VSS based on a powerful cryptographic primitive: *polynomial commitments*, which are explained in detail in Section 3. Fortunately, you do not need to understand in full detail how a polynomial commitment is constructed, but you must understand its definition and main properties. Our (n, t)-VSS will also be t-Byzantine resilient: the protocol will be correct even if an attacker is able to compromise and control t or less of the nodes. To guarantee this resilience, we need $n \geq 2t + 1$

2 Description

Your task is to implement and test a (n, t)-VSS scheme. The behavior of the protocol is described in Algorithms 1 and 2. To simplify the task, we will skip the possibility that the dealer P_d is corrupted or malicious (this simplifies the Sharing phase, overcoming the check that the dealer is honest).

For the implementation, follow these guidelines:

- You are not required to implement the code for the polynomial commitment scheme. We will provide you with a library for this if you use Python, C++, or Rust. For other languages, please contact the instructors.
- Write two programs, one for simulating the dealer P_d and another one to simulate an arbitrary client ℓ. At the end of the sharing phase, the dealer can write the shares in a file; the client can read the shares from this file and, as long as there are t honest shares in it, reconstruct the shared secret s.
- Alternatively, you can program the dealer and the clients in full and use the MQTT server from Assignment I to distribute the messages. This is more complex, however. Anyway, the protocol is the same, of course.

Algorithm 1 Sharing phase.

```
1: Input: a secret $s \in \mathbb{Z}_p^*$, a resilience threshold $t$
2: The dealer $P_d$ chooses a random degree $t$ polynomial $\phi(x) = \sum_{j=0}^{t} \phi_j x^j \in \mathbb{Z}_p[x]$ such that $\phi(0) = s$
3: $P_d$ broadcasts $\mathcal{C} = \mathsf{Commit}(\mathsf{PK}, \phi(x))$
4: For $\ell = 1, \ldots, n$, $P_d$ computes a share $s_\ell = \phi(\ell)$, a witness $w_\ell = \mathsf{CreateWitness}(\mathsf{PK}, \phi(x), \ell)$ and sends $\langle \ell, \phi(\ell), w_\ell \rangle$ to node $\ell$
```

Algorithm 2 Reconstruction phase.

```
1: Input: A set $\mathcal{T}$ of $t + 1$ or mores shares and witnesses, arbitrarily chosen among the clients.
2: Each listener verifies each of the broadcast shares $\langle i, \phi(i), w_i \rangle$ for $i \in \mathcal{T}$ using VerifyEval with the polynomial commitment scheme
3: If verification of the shares and witnesses is successful, the clients use the values $\{\langle i, \phi(i) : i \in \mathcal{T}\}$ to interpolate the polynomial $\phi(x)$ and get the secret $s = \phi(0)$.
```

3 Commitment schemes

Commitment schemes are fundamental components of many cryptographic protocols. A secure commitment scheme allows a committer to publish a value, called the *commitment*, which binds her to a message (*binding*) without revealing it (*hiding*). Later, she may *open* the commitment and reveal de committed message to a verifier, who can check that the message is consistent with the commitment.

There exist three well-known ways a committer can commit to a message. Let g and h be two random generators of a group $\mathbb{G}$ of prime order p. The committer can commit to a random message $m \in \mathbb{Z}_p$ simply as $\mathcal{C}_{\langle g \rangle}(m) = g^m$. This scheme is unconditionally binding, and computationally hiding under the assumption that the discrete logarithm (DL) problem is hard in $\mathbb{G}$. The second scheme, known as Pedersen commitment, is of the form $\mathcal{C}_{\langle g,h \rangle}(m, r) = g^m h^r$, where r is a randomly chosen value in $\mathbb{Z}_p$. Pedersen commitments are unconditionally hiding, and computationally binding under the DL assumption. Third, the commiter may publish $H(m)$ or $H(m||r)$ for a one-way function H. In practice a collision-resistant hash function is often used.

Now, consider committing to a polynomial $\phi(x) \in \mathbb{Z}_p[x]$ of degree t, with coefficients $\phi_0, \ldots, \phi_t$. We could commit to the string $\phi_0||\phi_1||\ldots||\phi_t$, for instance, but this limits the options for opening the commitment, since the opening reveals the entire polynomial. In many applications, most notably secret sharing, it is necessary to open just evaluations of the polynomial (i.e., $\phi(i)$ for some $i \in \mathbb{Z}_p$), that are to be revealed to different parties. One could also commit individually to each of the coefficients, but then the commitment size grows linearly in t.

We describe in this Section a polynomial commitment scheme —the KGZ polynomial commitment— that is more efficient than the previously commented methods, and give also some applications of this cryptographic tool in modern systems.

3.1 Definition

A polynomial commitment scheme consist of six algorithms: Setup, Commit, Open, VerifyPoly, Create-Witness, and VerifyEval.

- $\mathsf{Setup}(1^\lambda, t)$ generates an appropriate algebraic structure and a commitment public-private key pair $\langle \mathsf{PK}, \mathsf{SK} \rangle$ to commit to a polynomial of degree $\leq t$. SK is not required in the rest of the scheme and can be deleted.
- $\mathsf{Commit}\big(\mathsf{PK}, \phi(x)\big)$ outputs a commitment $\mathcal{C}$ to a polynomial $\phi(x)$ for the public key PK, and possibly some associated decommitment information d.
- $\mathsf{Open}\big(\mathsf{PK}, \mathcal{C}, \phi(x), d\big)$ outputs the polynomial $\phi(x)$ used while creating the commitment, with decommitment information d.
- $\mathsf{VerifyPoly}\big(\mathsf{PK}, \mathcal{C}, \phi(x), d\big)$ verifies that $\mathcal{C}$ is a commitment to $\phi(x)$, created with decommitment information d. If so, the algorithm outputs 1, otherwise it outputs 0.
- $\mathsf{CreateWitness}\big(\mathsf{PK}, \phi(x), i, d\big)$ outputs $\langle i, \phi(i), w_i \rangle$, where w_i is a witness for the evaluation $\phi(i)$ of $\phi(x)$ at the index i and d is the decommitment information.

- $\mathsf{VerifyEval}\big(\mathsf{PK}, \mathcal{C}, i, \phi(i), w_i\big)$ verifies that $\phi(i)$ is the evaluation at the index i of the polynomial committed in $\mathcal{C}$. If so, the algorithm outputs 1, otherwise it outputs 0.

A polynomial commitment scheme is secure (informally) if (i) the opening of a commitment is successfully verified; (ii) the output of CreateWitness is successfully verified by VerifyEval; (iii) it is computationally infeasible to generate a single commitment for two different polynomials (binding property); (iv) it is not possible to create two evaluation witnesses for different polynomials with the same commitment (evaluation binding property); (v) it is not possible to learn nothing about the evaluation of $\phi(j)$ given a commitment and a witness w_i for index $i \neq j$.

3.2 Construction 1

We now explain an efficient construction of a polynomial commitment scheme, $\mathsf{PolyCommit}_{\mathrm{DL}}$ based on the following elementary property: the monomial $x - i$ divides the polynomial $\phi(x) - \phi(i)$, for $i \in \mathbb{Z}_p$ and $\phi(x) \in \mathbb{Z}_p[x]$.

- $\mathsf{Setup}(1^\lambda, t)$ computes two groups, $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p such that there exists a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$. Let g be a generator of $\mathbb{G}$ and choose $a \in \mathbb{Z}_p^*$ as the secret key SK. Generate the $t+1$-tuple $\langle g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^t}\rangle \in \mathbb{G}^{t+1}$ and output $\mathsf{PK} = \langle g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^t}\rangle$.

- $\mathsf{Commit}\big(\mathsf{PK}, \phi(x)\big)$ computes the commitment $\mathcal{C} = g^{\phi(\alpha)}$ for the polynomial $\phi(x) \in \mathbb{Z}_p[x]$ of degree t or less, as
$$\mathcal{C} = \prod_{j=0}^{\deg(\phi)} \big(g^{\alpha^j}\big)^{\phi_j}.$$

- $\mathsf{Open}\big(\mathsf{PK}, \mathcal{C}, \phi(x)\big)$ outputs the committed polynomial $\phi(x)$.

- $\mathsf{VerifyPoly}\big(\mathsf{PK}, \mathcal{C}, \phi(x)\big)$ verifies that $\mathcal{C} \stackrel{?}{=} g^{\phi(\alpha)}$. Thus, if
$$\mathcal{C} = \prod_{j=0}^{\deg(\phi)} \big(g^{\alpha^j}\big)^{\phi_j}$$
the algorithm outputs 1, otherwise the output is 0. This only works if $\deg(\phi) \leq t$.

- $\mathsf{CreateWitness}\big(\mathsf{PK}, \phi(x), i\big)$ computes $\psi_i(x) = \frac{\phi(x)-\phi(i)}{x-i} \in \mathbb{Z}_p[x]$ and outputs $\langle i, \phi(i), w_i\rangle$, where the witness $w_i = g^{\psi(\alpha)}$ is computed in a manner similar to $\mathcal{C}$.

- $\mathsf{VerifyEval}\big(\mathsf{PK}, \mathcal{C}, i, \phi(i), w_i\big)$ verifies that $\phi(i)$ is the evaluation at the index i of the polynomial committed to by $\mathcal{C}$. The verification condition is
$$e(\mathcal{C}, g) \stackrel{?}{=} e(w_i, g^\alpha/g^i)e(g,g)^{\phi(i)}.$$
VerifyEval is correct because
$$\begin{aligned} e(w_i, g^\alpha/g^i)e(g,g)^{\phi(i)} &= e(g^{\psi_i(\alpha)(}, g^{\alpha-i})e(g,g)^{\phi(i)} = e(g,g)^{\psi_i(\alpha)(\alpha-i)+\phi(i)} \\ &= e(\mathcal{C}, g), \qquad \text{as } \phi(x) = \psi_i(x)(x-i) + \phi(i). \end{aligned}$$

$\mathsf{PolyCommit}_{\mathrm{DL}}$ is a secure polynomial commitment scheme (provided the DL and t-SDH assumptions hold in $\mathbb{G}$).

3.3 Construction 2

$\mathsf{PolyCommit}_{\mathrm{KZG}}$ is also based on the same algebraic property of polynomials, namely $x - i$ divides exactly $\phi(x) - \phi(i)$ for any $\phi(x) \in \mathbb{Z}_p[x]$ and $i \in \mathbb{Z}_p$. However, it uses an additional random polynomial $\hat{\phi}(x)$ to achieve unconditional hiding.

We also use the fact that $\mathsf{PolyCommit}_{\mathrm{DL}}$ is homomorphic: given two commitments $\mathcal{C}_{\phi_1}$ and $\mathcal{C}_{\phi_2}$ associated with polynomials $\phi_1(x)$ and $\phi_2(x)$, respectively, one can compute the commitment $\mathcal{C}_\phi$

for $\phi(x) = \phi_1(x) + \phi_2(x)$ as $\mathcal{C}_\phi = \mathcal{C}_{\phi_1}\mathcal{C}_{\phi_2}$. Further, given two witness-tuples $\langle i, \phi_1(i), w_{\phi_1 i}\rangle$ and $\langle i, \phi_2(i), w_{\phi_{2i}}\rangle$ at index i associated with polynomials $\phi_1(x)$ and $\phi_2(x)$ respectively, the corresponding tuple for index i for the polynomial $\phi(x) = \phi_1(x) + \phi_2(x)$ can be given as $\langle i, \phi_1(i) + \phi_2(i), w_{\phi_{1i}} w_{\phi_{2i}}\rangle$. The $\mathsf{PolyCommit}_{\mathrm{KZG}}$ uses this homomorphic property to combine two commitments, although each commitment uses a different generator., The definition of the scheme is as follows.

- $\mathsf{Setup}(1^\lambda, t)$ computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p such that there exists a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G}_T \longrightarrow \mathbb{G}_T$. Choose two generators $g, h \in \mathbb{G}$, and let $\alpha \in \mathbb{Z}_p^*$ be the secret key SK. Setup also generates the $2t+2$-tuple $\langle g, g^\alpha, g^{\alpha^2}, \dots, g^{\alpha^t}, h, h^\alpha, \dots, h^{\alpha^t}\rangle \in \mathbb{G}^{2t+2}$ and outputs the public key $\mathsf{PK} = \langle g, g^\alpha, g^{\alpha^2}, \dots, g^{\alpha^t}, h, h^\alpha, \dots, h^{\alpha^t}\rangle \in \mathbb{G}^{2t+2}$.

- $\mathsf{Commit}\big(\mathsf{PK}, \phi(x)\big)$ chooses $\hat{\phi}(x) \in \mathbb{Z}_p[x]$ of degree t and computes the commitment $\mathcal{C} = g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)} \in \mathbb{G}$ for the polynomial $\phi(x) \in \mathbb{Z}_p[x]$ of degree t or less. More precisely, it outputs
$$\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$$
as the commitment to $\phi(x)$.

- $\mathsf{Open}\big(\mathsf{PK}, \phi(x), \hat{\phi}(x)\big)$ outputs the committed polynomials $\phi(x)$ and $\hat{\phi}(x)$.

- $\mathsf{VerifyPoly}\big(\mathsf{PK}, \mathcal{C}, \phi(x), \hat{\phi}(x)\big)$ verifies that $\mathcal{C} \stackrel{?}{=} g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)}$. If
$$\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$$
the algorithm outputs 1, otherwise it outputs 0. This only works when both polynomials have degree less than or equal to t.

- $\mathsf{CreateWitness}\big(\mathsf{PK}, \phi(x), \hat{\phi}(x), i\big)$ computes $\psi_i(x) = \frac{\phi(x)-\phi(i)}{x-i}$ and $\hat{\psi}_i(x) = \frac{\hat{\phi}(x)-\hat{\phi}(i)}{x-i}$, and outputs $\langle i, \phi(i), \hat{\phi}(i), w_i\rangle$. Here, the witness is $w_i = g^{\psi_i(\alpha)} h^{\hat{\psi}_i(\alpha)}$.

- $\mathsf{VerifyEval}\big(\mathsf{PK}, i, \phi(i), \hat{\phi}(i), w_i\big)$ verifies that $\phi(i)$ is the evaluation at the index i of the polynomial committed to by $\mathcal{C}$. If $e(\mathcal{C}, g) \stackrel{?}{=} e(w_i, g^\alpha/g^i) e(g^{\phi(i)} h^{\hat{\phi}(i)}, g)$, the algorithm outputs 1, else ot outputs 0.

Suppose $h = g^\lambda$ for some unknown λ. Then $\mathsf{VerifyEval}$ is correct because

$$\begin{aligned}
e(w_i, g^\alpha/g^i) e(g^{\phi(i)} h^{\hat{\phi}(i)}, g) &= e(g^{\psi_i(\alpha)+\lambda\hat{\psi}_i(\alpha)}, g^{\alpha-i}) e(g,g)^{\phi(i)+\lambda\hat{\phi}(i)} \\
&= e(g,g)^{(\psi_i(\alpha)(\alpha-i)+\phi(i)+\lambda(\hat{\psi}_i(\alpha)(\alpha-i)+\hat{\phi}(i))} \\
&= e(g,g)^{\phi(\alpha)+\lambda\hat{\phi}(\alpha)} \quad \text{as } \phi(x) = \psi_i(x)(x-i)+\phi(i) \text{ and } \hat{\phi}(x) = \hat{\psi}_i(x)(x-i)+\hat{\phi}(i) \\
&= e(g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)}, g) = e(\mathcal{C}, g).
\end{aligned}$$

$\mathsf{PolyCommit}_{\mathrm{KZG}}$ is a secure polynomial commitment scheme provided the t-SDH assumption holds in $\mathbb{G}$.