Secret sharing based secure multiparty computation protocols
Sharing the contents of wires

- Assume that the only operations of circuits are $\oplus$ and $\&$.
  - These are the addition and multiplication in the field $\mathbb{Z}_2$.
  - They, together with the constant 1, are sufficient to express any functionality.
    - Let there be an extra 1-gate that takes no inputs.
- During the protocol, the parties compute for all gates $g$ the values $a^1_g$ and $a^2_g$, such that
  - the real value computed by $g$ is $a^1_g + a^2_g$;
  - $P_1$ knows $a^1_g$ and $P_2$ knows $a^2_g$.
- The protocol is well-suited for functionalities with separate outputs.
- At the end of the protocol, $P_1$ will send to $P_2$ the values $a^1_g$ corresponding to the output gates $g$ of $P_2$.
- $P_2$ behaves similarly.
The protocol

- **Sharing the inputs**
  - For $P_i$’s input $b$ at the input gate $g$ generate a random bit $r$ and send $(g, r)$ to the other party. Let $a^i_g = b + r$.
  - For the 1-gate $g$ let $P_1$ generate a random bit $r$ and send $(g, r)$ to $P_2$. Let $a^1_g = r + 1$.
  - When $P_i$ receives $(g, r)$ from the other party, set $a^i_g = r$.

- **Evaluating an addition gate** Let $g = g_1 + g_2$. Define $a^i_g = a^i_{g_1} + a^i_{g_2}$.

- **Communicating the outputs** If $g$ is an output gate for $P_i$, then the other party $P_j$ will send $(g, a^j_g)$ to $P_i$ once he has it. $P_i$ outputs $a^i_g + a^j_g$ as the output of that gate.
Evaluating a multiplication gate

- Let $g = g_1 \cdot g_2 = (a^1_{g_1} + a^2_{g_1}) \cdot (a^1_{g_2} + a^2_{g_2})$.
- We’ll define a protocol for finding $a^1_g$ and $a^2_g$, such that
  - $P_i$ does not learn anything besides $a^i_g$;
  - $a^i_g$ is uniformly distributed;
  - $a^1_g + a^2_g = a_g$. 

Exercise. Correctness and security?
Evaluating a multiplication gate

- Let \( g = g_1 \cdot g_2 = (a_{g_1}^1 + a_{g_1}^2) \cdot (a_{g_2}^1 + a_{g_2}^2) \).
- We’ll define a protocol for finding \( a_g^1 \) and \( a_g^2 \), such that
  - \( P_i \) does not learn anything besides \( a^i_g \);
  - \( a_g^i \) is uniformly distributed;
  - \( a_g^1 + a_g^2 = a_g \).
- \( a_g^1 \) is picked uniformly from \( \{0, 1\} \).
- \( P_1 \) defines
  \[
  m_1 = a_g^1 + a_g^1 a_{g_1}^1 a_{g_2}^1 \\
  m_2 = a_g^1 + a_g^1 (a_{g_2}^1 + 1) \\
  m_3 = a_g^1 + (a_g^1 + 1) a_{g_2}^1 \\
  m_4 = a_g^1 + (a_g^1 + 1)(a_{g_2}^1 + 1)
  \]
- \( P_2 \) defines \( a_g^2 = m_2 a_{g_1}^2 + a_{g_2}^2 + 1 \).
  - Use oblivious transfer to transmit that \( m \) to \( P_2 \).

Exercise. Correctness and security?
Exercise

How can one party simulate the computation of this protocol?
Multi-party semi-honest case

- A protocol \( \Pi \) securely evaluates the function \((y_1, \ldots, y_n) = f(x_1, \ldots, x_n)\) if
  - there exists a PPT simulator \( S \), such that
  - for each \( I = \{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\} \)
  - for all \( x_1, \ldots, x_n \)
  - the distribution \( S(I, (x_{i_1}, \ldots, x_{i_m}), (y_{i_1}, \ldots, y_{i_m})) \) equals
  - the joint view of the parties \( P_{i_1}, \ldots, P_{i_m} \) in the execution of \( \Pi(x_1, \ldots, x_n) \).
The protocol

- Most steps of the two-party case easily generalize to the multi-party case.
- How about multiplication?
The protocol

- Most steps of the two-party case easily generalize to the multi-party case.
- How about multiplication?
- We have $a_1^{g_1}, \ldots, a_n^{g_1}, a_1^{g_2}, \ldots, a_n^{g_2}$ with $\sum_j a_j^{g_i} = a_i$.
- We want $a_1^{g_1}, \ldots, a_n^{g_1}$ that sum up to $a_1^{g_1} \cdot a_2^{g_2}$.

\[
\left(\sum_{j=1}^{n} a_j^{g_1}\right) \cdot \left(\sum_{j=1}^{n} a_j^{g_2}\right) = \sum_{k=1}^{n} a_k^{g_1} a_k^{g_2} + \sum_{1 \leq i < j \leq n} (a_i^{g_1} a_j^{g_2} + a_i^{g_2} a_j^{g_1}) = \\
(1-(n-1)) \sum_{k=1}^{n} a_k^{g_1} a_k^{g_2} + \sum_{1 \leq i < j \leq n} (a_i^{g_1} a_j^{g_2} + a_i^{g_2} a_j^{g_1} + a_i^{g_1} a_j^{g_1} + a_j^{g_1} a_i^{g_2}) = \\
(2-n) \sum_{k=1}^{n} a_k^{g_1} a_k^{g_2} + \sum_{1 \leq i < j \leq n} (a_i^{g_1} + a_i^{g_1})(a_j^{g_1} + a_j^{g_2})
\]
Each party $P_i$ engages in the **two-party** multiplication protocol with all other parties $P_j$.

As result, party $P_i$ learns the values $c^{i,j}$, such that

$$c^{i,j} + c^{j,i} = (a^i_{g_1} + a^i_{g_2}) \cdot (a^j_{g_1} + a^j_{g_2})$$

$P_i$ defines $a^i_g = n \cdot a^i_{g_1} a^i_{g_2} + \sum_{j \neq i} c^{i,j}$.

**Exercise.** Correctness, security? Uniformity of the values $a^i_g$?
Secret-sharing schemes

- There is a set of parties $\mathbf{P} = \{ P_1, \ldots, P_n \}$.
- There is some (secret) value $v$.
  - Shares of $v$ are distributed among $P_1, \ldots, P_n$.
- There is a set of subsets of parties $\mathcal{A} \subseteq \mathcal{P}(\mathbf{P})$.
  - $\mathcal{A}$ is upwards closed — if $P_1 \in \mathcal{A}$ and $P_1 \subseteq P_2$, then also $P_2 \in \mathcal{A}$.
  - $\mathcal{A}$ is called an access structure.
  - Let us call the elements of $\mathcal{A}$ privileged sets.
- Certain parties $P_{i_1}, \ldots, P_{i_k}$ have come together and are trying to find out $v$.
- They must succeed only if $\{ P_{i_1}, \ldots, P_{i_k} \} \in \mathcal{A}$. 

15.10.2019 9
General solution

- Let $v$ be an element of some (additive) group $G$.
- Express $\varphi$ as a propositional formula $\overline{\varphi}(x_1, \ldots, x_n)$, such that for each $Q \subseteq P$
  \[
  \overline{\varphi}(P_1 \in Q, \ldots, P_n \in Q) \iff Q \in \varphi.
  \]
- Use only operations AND and OR (of arbitrary arity) in $\overline{\varphi}$.
- Define a share for each node in the syntax tree of $\overline{\varphi}$:
  - The share of the root node is $v$.
  - If the share of an OR-node is $x$, then the shares of all its immediate descendants are $x$, too.
  - If the share of an AND-node of arity $m$ is $x$, then generate $r_1, \ldots, r_{m-1} \in_R G$ and put $r_m = x - \sum_{i=1}^{m-1} r_i$. The shares of the immediate descendants are $r_1, \ldots, r_m$.
- Give the party $P_i$ the shares of all leaf nodes marked with $x_i$. 

15.10.2019
Example

- Let \( P = \{ P_1, P_2, Q_1, Q_2, Q_3 \} \).
  - Let \( P_1 \) and \( P_2 \) be allowed to know the secret.
  - Let two \( Q \)-s be allowed to replace one of the \( P \)-s.

\[
\overline{\phi}(P_1, P_2, Q_1, Q_2, Q_3) = P_1 \& P_2 \lor \\
P_1 \&(Q_1 \& Q_2 \lor Q_1 \& Q_3 \lor Q_2 \& Q_3) \lor P_2 \&(Q_1 \& Q_2 \lor Q_1 \& Q_3 \lor Q_2 \& Q_3)
\]
Example

\[ P_1 \land P_2 \land (Q_1 \land (Q_2 \land Q_3)) \land (Q_1 \land Q_2) \land (Q_1 \land Q_3) \land Q_2 \land Q \]
Example
Example
Example
Example
Example

- We generate the values $r_1, \ldots, r_9 \in \mathbb{R}$ and give the following values to following parties:
  - $P_1$ learns $s_{11} = v - r_1$ and $s_{12} = v - r_2$;
  - $P_2$ learns $s_{21} = r_1$ and $s_{22} = v - r_3$;
  - $Q_1$ learns $t_{11} = r_4$, $t_{12} = r_5$, $t_{13} = r_7$ and $t_{14} = r_8$;
  - $Q_2$ learns $t_{21} = r_2 - r_4$, $t_{22} = r_6$, $t_{23} = r_3 - r_7$ and $t_{24} = r_9$;
  - $Q_3$ learns $t_{31} = r_2 - r_5$, $t_{32} = r_2 - r_6$, $t_{33} = r_3 - r_8$ and $t_{34} = r_3 - r_9$.

- When a privileged set of parties meet then they figure out which of the values to add up to recover $v$.
- A non-privileged set gets no information about $v$.
The components

- Number of parties $n$.
- The secret $v$.
- The parties $P_1, \ldots, P_n$ holding the shares of $v$, and the dealer $D$ that originally knows $v$.
- The access structure $\wp$.
  - $\wp$ is a $t$-threshold structure if all minimal elements in $\wp$ have the cardinality $t$.
- The dealing protocol, where $D$ distributes the shares among $P_1, \ldots, P_n$.
- The recovery protocol, where a privileged set computes $v$. 
Shamir’s threshold secret sharing scheme

- Let \( v \in \mathbb{F} \) for some (finite) field \( \mathbb{F} \).
  - In practice, \( \mathbb{F} \) is \( \mathbb{Z}_p \) for some suitable prime \( p \).
- Shamir’s \((n, t)\)-scheme is for \( n \) parties, where \( \mathcal{O} \) is the \( t \)-threshold structure and \( n < |\mathbb{F}| \).

**Dealing:**
- The dealer randomly chooses values \( a_1, \ldots, a_{t-1} \in \mathbb{F} \).
- He defines the polynomial \( q(x) = v + a_1x + a_2x^2 + \cdots + a_{t-1}x^{t-1} \).
- The dealer securely sends to each \( P_i \) his share \( s_i = q(i) \).

**Recovering \( v \):**
- The parties \( P_{i_1}, \ldots, P_{i_t} \) together know that
  - \( q(i_1) = s_i, \ldots, q(i_t) = s_t \);
  - The degree of \( q \) is at most \( t - 1 \).
- This information is sufficient to recover the coefficients of \( q \).
Interpolating polynomials

**Theorem.** Let $x_1, y_1, \ldots, x_t, y_t \in \mathbb{F}$, such that the values $x_1, \ldots, x_t$ are all different. Then there exists exactly one polynomial $q$ of degree at most $t - 1$, such that $q(x_i) = y_i$ for all $i \in \{1, \ldots, t\}$.

**Proof.** This polynomial $q$ is (Lagrange interpolation formula)

\[ q(x) = \sum_{j=1}^{t} y_j \prod_{x_k \in \{x_1, \ldots, x_t\} \setminus \{x_j\}} \frac{x - x_k}{x_j - x_k}. \]

It’s degree is $\leq t - 1$ and it satisfies $q(x_i) = y_i$ for all $i$.

There cannot be more than one: if $q'(x_i) = y_i$ for all $i \in \{1, \ldots, t\}$ and $\deg q' \leq t - 1$, then $(q - q')$ is a polynomial of degree at most $t - 1$ with at least $t$ roots $(x_1, \ldots, x_t)$. Hence $q - q' = 0$. □
Shamir’s scheme: simpler recovery

- The parties $P_{i_1}, \ldots, P_{i_t}$ are not interested in the entire polynomial, but just the secret value $v = q(0)$.
- According to Lagrange interpolation formula

$$v = \sum_{j=1}^{t} s_{ij} \prod_{i_k \in \{i_1, \ldots, i_t\} \setminus \{ij\}} \frac{i_k}{i_k - i_j}.$$

- In particular, note that $v$ is computed as a linear combination of the shares $s_{ij}$ with public coefficients.
Security of Shamir’s scheme

- Suppose that we are given shares $s_{i_1}, \ldots, s_{i_{t-1}}$.
- Then for each possible value of $v$, there exists exactly one polynomial $q$ of degree at most $t$, such that

$$q(0) = v, \quad q(i_1) = s_{i_1}, \ldots, \quad q(i_{t-1}) = s_{i_{t-1}}.$$

- Hence all values of $v$ are possible. Moreover, they are equally possible.
  - There is the same number of suitable polynomials for each value of $v$.
- Similarly, if we have even less shares then all values of $v$ are equally possible.
Exercise

Let two secrets be shared:
- the shares of $v$ are $s_1, \ldots, s_n$;
- the shares of $v'$ are $s'_1, \ldots, s'_n$.

Let $a \in \mathbb{F}$. How can the parties $P_1, \ldots, P_n$ obtain shares for the values
- $v + v'$?
- $av$?
- $v + a$?
Gennaro-Rabin-Rabin multiplication protocol

- Assume $t - 1 < n/2$. (in other words, $t - 1 \leq (n - 1)/2$)
- Let $f, f'$ be polynomials of degree $\leq t - 1$ used to share $v, v'$.
- $f(0) = v$. $f'(0) = v$. Let $f'' = f \cdot f'$. Then $f''(0) = v \cdot v''$.
- The degree of $f''$ is $\leq 2(t - 1) \leq n - 1$.
- The values of $f''$ on $n$ points suffice to reconstruct $f''$.
  - Party $P_i$ can compute $f''(i)$ as $s_i \cdot s'_i$.
  - But we don't want to use $f''$ to share $v''$.
- There exist (public) $r_1, \ldots, r_n$, such that $f''(0) = \sum_{i=1}^{n} r_i(s_i \cdot s'_i)$.
- By Lagrange interpolation formula $r_i = \prod_{1 \leq j \leq n, j \neq i} j/(j - i)$.
- At least $t$ of $r_1, \ldots, r_n$ are non-zero.
  - If only $r_{i_1}, \ldots, r_{i_{t-1}}$ were non-zero, then
  
  $$v = (f \cdot 1)(0) = \sum_{i=1}^{n} r_i f(i) 1(i) = \sum_{j=1}^{t-1} r_j s_j,$$

  allowing $P_{i_1}, \ldots, P_{i_{t-1}}$ to determine $v$. 

15.10.2019
Gennaro-Rabin-Rabin multiplication protocol

- Each party $P_i$ randomly generates a polynomial $f_i$ of degree at most $t - 1$, such that $f_i(0) = s_i \cdot s'_i$.
- Party $P_i$ sends to party $P_j$ the value $u_{ij} = f_i(j)$.
- Party $P_i$ receives the values $u_{1i}, \ldots, u_{ni}$.
- $P_i$ defines $s_i'' = \sum_{j=1}^{n} r_j u_{ji}$.
- The shares $s_1'', \ldots, s_n''$ correspond to the polynomial $\hat{f} = \sum_{j=1}^{n} r_j f_j$.
  - It is a random polynomial because $f_i$-s were randomly generated.
  - It is independent from any $f_1, \ldots, f_{i-1}$, because at least $t$ of the values $r_1, \ldots, r_n$ are non-zero.
- This polynomial shares the value
  \[ \hat{f}(0) = \sum_{j=1}^{n} r_j \cdot f_j(0) = \sum_{j=1}^{n} r_j s_j s'_j = f''(0) = v'' . \]
Secure computation with multiplication triples

- Let $[v]$ denote the value $v$ that is protected (e.g. secret-shared)
- Multiplication triple: $([a], [b], [c])$, where $a, b$ are random and $c = ab$
- Given $[u], [v]$, we compute $[uv]$ as follows:
  - Declassify (i.e. open) $\alpha = [u] - [a]$ and $\beta = [v] - [b]$
  - Compute
    $$[uv] = \alpha [v] + \beta [u] + [c] - \alpha \beta$$
    This is a linear combination of protected values
- The multiplication triple is thereby consumed
- The computation is split to offline and online parts
  - Online part will be fast(er)
Inputting a value

- A party $P$ wants to input the value $d$ into the private computation
- Let $[r]$ be a random protected value
- Open $[r]$ to $P$
- $P$ publishes $r' = d - r$
- Compute $[d] = [r'] + [r]$
Over half of the parties must be honest (for inf.-theor. security)

- Consider a two-party protocol $\Pi$ for computing the AND of two bits.
- Let $\Pi(b_1, r_1, b_2, r_2)$ be the sequence of messages exchanged for party $P_i$’s bit $b_i$ and random coins $r_i$.

$$\forall r_1, r_2^0 \exists r_2^1 : \Pi(0, r_1, 0, r_2^0) = \Pi(0, r_1, 1, r_2^1)$$
$$\forall r_1, r_2^1 \exists r_2^0 : \Pi(0, r_1, 0, r_2^0) = \Pi(0, r_1, 1, r_2^1)$$
$$\forall r_1, r_2^0, r_2^1 : \Pi(1, r_1, 0, r_2^0) \neq \Pi(1, r_1, 1, r_2^1)$$

- Party $P_2$ whose input is $b_2 = 0$ and random coins $r_2^0$ can find $b_1$ as follows:
  - Let $\mathcal{T}$ be the exchanged sequence of messages.
  - Try to find such $(b', r', r_2^1)$, that $\Pi(b', r', 1, r_2^1) = \mathcal{T}$.
  - If such triple exists then $b_1 = 0$. If not, then $b_1 = 1$.

**Exercise.** Generalize this result to more than 2 parties.
Verifiable secret sharing

- If some party $P_i$ is malicious, then it can input a wrong share to the recovery protocol.
- The recovered secret $v$ will then be incorrect.
- Also, a malicious dealer may give inconsistent shares to the parties $P_i$.
- In verifiable secret sharing the parties commit to the shares they have received.
Verifiable secret sharing

- If some party $P_i$ is malicious, then it can input a wrong share to the recovery protocol.
- The recovered secret $v$ will then be incorrect.
- Also, a malicious dealer may give inconsistent shares to the parties $P_i$.
- In verifiable secret sharing the parties commit to the shares they have received.
- A malicious party $P_i$ may also send $s_{it}$ to one party, but $s'_{it}$ to some other party.
- In multi-party protocols with malicious participants, a broadcast channel is often needed.
- We thus assume the existence of a broadcast channel.
- It can be implemented using point-to-point channels and the Byzantine agreement.
Feldman’s scheme

- Let $\mathbb{F} = \mathbb{Z}_p$. Let $G$ be a group with hard discrete log., such that $|G|$ is divisible by $p$. Let $g \in G$ have order $p$.
- Let $D$ use Shamir’s scheme to share $v$. When $D$ has constructed the polynomial $q(x) = v + \sum_{i=1}^{t-1} a_i x^i$, he (authentically) broadcasts

$$y_0 = g^v, \quad y_1 = g^{a_1}, \ldots, \quad y_{t-1} = g^{a_{t-1}}$$

in addition to sending the shares to the parties $P_i$.
- Whenever a party sees a share $s_j$ he checks its consistency:

$$g^{s_j} \overset{?}{=} \prod_{i=0}^{t-1} y_j^i$$

Exercise. What does the consistency check do?
Security of Feldman’s scheme

- Nobody can cheat — the “commitments” $y_0, \ldots, y_{t-1}$ fix the polynomial $q$.
  - Everybody can check whether $q(i)$ equals a given value.
- Something about the secret can be leaked, because $y_0 = g^v$ does not fully hide $v$.
  - Use only the hard-core bits of discrete logarithm to store the “real” secret in $v$.
    - This makes the shares larger.
Pedersen’s scheme

Recall Pedersen’s commitment scheme:

- Let $h \in G$ be another element of order $p$, such that nobody knows $\log_g h$.
- To commit $m \in \mathbb{Z}_p$, the committer randomly generates $r \in \mathbb{Z}_p$ and sends $g^m h^r$ to the verifier.
- To open the commitment, send $(m, r)$ to the verifier.
- The commitment is unconditionally hiding, because $g^m h^r$ is a random element of $\langle g \rangle$.
- The commitment is computationally binding, because the ability to open a commitment in two different ways allows to compute $\log_g h$.

In Pedersen’s VSS, the dealer commits to the coefficients of the polynomial $q$. 
Pedersen’s scheme

- Dealing protocol
  - $D$ randomly chooses $a_1, \ldots, a_{t-1}, a'_0, \ldots, a'_{t-1} \in \mathbb{Z}_p$. Also defines $a_0 = v$.
  - Define $q(x) = \sum_{i=0}^{t-1} a_i x^i$ and $q'(x) = \sum_{i=0}^{t-1} a'_i x^i$.
  - The share $(s_i, s'_i)$ of $P_i$ is $(q(i), q'(i))$.
  - $D$ broadcasts $y_i = g^{a_i} h^{a'_i}$ for $i \in \{0, \ldots, t-1\}$.

- Verification: when somebody sees a share $(s_j, s'_j)$, he verifies

\[
g^{s_j} h^{s'_j} \overset{?}{=} \prod_{i=0}^{t-1} y_j^i
\]
Security of Pedersen’s scheme

- The broadcast value $y_0$ hides $v$ unconditionally.
- Ability to change a share (or the pair $(v, a'_0)$) implies the knowledge of $\log_g h$.
- Having less than $t$ shares allows one to freely choose the secret $v$. Then there exists an $a'_0$ that is consistent with $y_0$.

**Exercise.** How to construct linear combinations of shared secrets when using Feldman’s or Pedersen’s secret sharing scheme? I.e. how do the dealer’s commitments change?
Exercise

Repeat the previous MPC construction, but using a verifiable secret sharing scheme. This exercise shows the possibility of MPC, where

- security is computational;
- the number of corrupted parties is strictly less than $n/2$;
- the adversary is malicious;
- there is a broadcast channel;
- the adversary can shut down the computation.

The security can be made unconditional and shutdown possibilities can be eliminated.
Threshold encryption

- Public-key encryption system.
- The public key is a single value.
- The secret key is distributed among several authorities.
- To decrypt a ciphertext $c$:
  - Each authority computes $D(sk_i, c)$ and broadcasts it.
  - If at least $t$ authorities have broadcast the share of the decrypted ciphertext, the plaintext can be reconstructed from them.
ElGamal encryption scheme

Let $G$, $g$, $p$ be as before.

- Secret key — $\alpha \in R \mathbb{Z}_p$. Public key — $\chi := g^\alpha$.
- Plaintext space: $G$. Ciphertext space: $G \times G$.
- To encrypt a plaintext $m \in G$:
  - randomly generate $r \in \mathbb{Z}_p$;
  - output $(g^r, m \cdot \chi^r)$.
- To decrypt a ciphertext $(c_1, c_2)$:
  - output $c_2 \cdot c_1^{-\alpha}$.
- Note, that after the decryption, the value $c_1^\alpha = \chi^r$ is not sensitive any more.
Threshold scheme

- Use ElGamal scheme. Distribute the secret key $\alpha$ among the $n$ authorities $P_1, \ldots, P_n$ using Shamir’s $(n, t)$-scheme.
  - Let the shares be $s_1, \ldots, s_n$.
  - Recall that for each $Q = \{i_1, \ldots, i_t\}$ there exist coefficients $\gamma^Q_{i_1}, \ldots, \gamma^Q_{i_t} \in \mathbb{Z}_p$, depending only on $Q$, such that $\alpha = \sum_{j=1}^t \gamma^Q_{i_j} s_{i_j}$.

- Decryption:
  - given $(c_1, c_2)$, the authority $P_i$ broadcasts $d_i = c_1^{s_i}$.
  - given $d_{i_1}, \ldots, d_{i_t}$, where $\{i_1, \ldots, i_t\} = Q$, we find

$$c_1^\alpha = \prod_{j=1}^t d_{i_j}^{\gamma^Q_{i_j}}$$

and the plaintext is $m = c_2 \cdot (c_1^\alpha)^{-1}$.

Exercise. How could we use Feldman’s scheme for verifiability?