International Standard

ISO/IEC 4922-2

First edition
2024-03

Information security — Secure multiparty computation —

Part 2:
Mechanisms based on secret sharing

Sécurité de l'information — Calcul multipartite sécurisé —

Partie 2: Mécanismes basés sur le partage de secret

Reference number
ISO/IEC 4922-2:2024(en)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

A list of all parts in the ISO/IEC 4922 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

Secure multiparty computation is a cryptographic technique that computes a function on a message while maintaining the confidentiality of the message. The technique is used to outsource computations to two or more stakeholders while preserving privacy. To facilitate the effective use of secure multiparty computation and maintain interoperability, the ISO/IEC 4922 series specifies secure multiparty computation and related technologies.

Secure multiparty computation often uses cryptographic mechanisms as building blocks. For secure multiparty computation which is based on secret sharing, secret sharing schemes are used as building blocks.

Secret sharing is a cryptographic technique used to protect the confidentiality of a message by dividing it into pieces called shares. A secret sharing scheme has two main parts: a message sharing algorithm for dividing the message into shares and a message reconstruction algorithm for recovering the message from all or a subset of the shares. The ISO/IEC 19592 series specifies secret sharing and related technologies. In secure multiparty computation based on secret sharing, a message is shared among participants called parties via a message sharing algorithm. The parties compute a function on the shared message while maintaining its confidentiality and obtain shares of the function output. The function output can be obtained using a message reconstruction algorithm taking as input all or a subset of the output shares. This document specifies secure multiparty computation based on secret sharing, especially mechanisms to compute a function on the shared secret.

Information security — Secure multiparty computation —

Part 2: Mechanisms based on secret sharing

1 Scope

This document specifies the processes for secure multiparty computation mechanisms based on the secret sharing techniques which are specified in ISO/IEC 19592-2. Secure multiparty computation based on secret sharing can be used for confidential data processing. Examples of possible applications include collaborative data analytics or machine learning where data are kept secret, secure auctions where each bidding price is hidden, and performing cryptographic operations where the secrecy of the private keys is maintained.

This document specifies the mechanisms including but not limited to addition, subtraction, multiplication by a constant, shared random number generation, and multiplication with their parameters and properties. This document describes how to perform a secure function evaluation using these mechanisms and secret sharing techniques.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 4922-1, *Information security — Secure multiparty computation — Part 1: General*

ISO/IEC 19592-1, *Information technology — Security techniques — Secret sharing — Part 1: General*

ISO/IEC 19592-2:2017, *Information technology — Security techniques — Secret sharing — Part 2: Fundamental mechanisms*

3 Terms and definitions

For this document, the terms and definitions given in ISO/IEC 4922-1, ISO/IEC 19592-1, ISO/IEC 19592-2 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
group
set of elements G and an operation + defined on the set of elements such that: (i) $a + (b + c) = (a + b) + c$ for every a, b and c in G; (ii) there exists an identity element e in G such that $a + e = e + a = a$ for every a in G; (iii) for every a in G there exists an inverse element $-a$ in G such that $a + (-a) = (-a) + a = e$

[SOURCE: ISO/IEC 19592-2:2017, 3.8, modified — the notation "a^{-1}" has been replaced by "$-a$".]

3.2
finite cyclic group
abelian group (G,+) that is a *group* (3.1) and $a + b = b + a$ for every a and b in G (with identity element 0), containing a finite number of elements, such that there exists g in G, where every a in G is equal to g or g added to itself a finite number of times

Note 1 to entry: Definition adapted from ISO/IEC 19592-2:2017, 3.6.

3.3
ring
set of elements R and a pair of operations (+, *) defined on R such that: (i) $a * (b + c) = a * b + a * c$ for every a, b and c in R; (ii) R together with + forms an abelian group that is a *group* (3.1) and $a + b = b + a$ for every a and b in R (with identity element 0); (iii) R excluding 0 together with * forms a monoid such that: (i) $a * (b * c) = (a * b) * c$ for every a, b and c in R; (ii) there exists an identity element e in R such that $a * e = e * a = a$ for every a in R

3.4
finite ring
ring (3.3) containing a finite number of elements

3.5
field
set of elements K and a pair of operations (+, *) defined on K such that: (i) $a * (b + c) = a * b + a * c$ for every a, b and c in K; (ii) K together with + forms an abelian group that is a *group* (3.1) and $a + b = b + a$ for every a and b in K (with identity element 0); (iii) K excluding 0 together with * forms an abelian group that is a *group* and $a * b = b * a$ for every a and b in K

[SOURCE: ISO/IEC 19592-2:2017, 3.5, modified — the phrases "that is a *group* (3.1) and $a + b = b + a$ for every a and b in K" and "that is a *group* and $a * b = b * a$ for every a and b in K" have been added.]

3.6
finite field
field (3.5) containing a finite number of elements

[SOURCE: ISO/IEC 19592-2:2017, 3.7]

3.7
deterministic random bit generator
DRBG
random bit generator that produces a random-appearing sequence of bits by applying a deterministic algorithm to a suitably random initial value called a seed and, possibly, some secondary inputs upon which the security of the random bit generator does not depend

Note 1 to entry: A DRBG takes a high-entropy, secret random string as input and outputs a longer string of bits, which is computationally indistinguishable from random data to adversaries not knowing the input.

[SOURCE: ISO/IEC 18031:2011, 3.10, modified — the original note to entry has been replaced.]

3.8
replicated additive secret sharing scheme
secret sharing scheme in which shares are specified as subsets of a set of random values that sum to the secret

Note 1 to entry: The replicated additive secret sharing scheme is specified in ISO/IEC 19592-2.

3.9
Shamir secret sharing scheme
secret sharing scheme in which shares are specified as points on a random polynomial for which the secret is the constant

Note 1 to entry: The Shamir secret sharing scheme is specified in ISO/IEC 19592-2.

4 Symbols and abbreviated terms

$\boldsymbol{A}$	adversary structure of threshold k
A^t	set of t-tuples of elements of A
$A \subset B$	A is a subset of B
$a \in A$	a is an element of A
$A \times B$	direct product of A and B, i.e. the set of all ordered pairs (a, b), where $a \in A$ and $b \in B$
$\|A\|$	number of elements in A
$[a]_i$	i-th share of a message a
$[a]$	vector of shares $([a]_1, ..., [a]_n)$
${}_i\mathrm{C}_j$	binomial coefficient, namely i choose j
G	finite cyclic group
K	finite field
$K[x]$	set of all polynomials in x with coefficients in K
k	threshold of shares
m	number of sub-shares for each party in an instance of the replicated additive secret sharing scheme
n	number of shares
P_i	i-th computing party of secure multiparty computation
R	finite ring
Recover	message reconstruction algorithm of a secret sharing scheme
r_Z	sub-share of the replicated additive secret sharing scheme corresponding to $Z \in \boldsymbol{A}$
Share	message sharing algorithm of a secret sharing scheme
x_i	non-zero fixed field element corresponding to party P_i, where the value x_i are distinct and known to all computing parties

5 Secure multiparty computation based on secret sharing

5.1 General

This clause specifies fundamental concepts for secure multiparty computation based on secret sharing. The secret sharing schemes and the parameters used in this document are described in 5.2. The process flow and parameters for secure multiparty computation based on secret sharing are described in 5.3. Annex A lists the object identifiers which shall be used to identify the mechanisms specified in this document. Annex B provides numerical examples for the mechanisms specified in this document, which can be used for checking the correctness of implementations. Annex C provides security considerations that can be used to obtain additional information regarding the security of all the mechanisms specified in this document.

5.2 Secret sharing

The secure multiparty computation schemes based on secret sharing specified in this document use the Shamir and replicated additive secret sharing schemes. These secret sharing schemes are defined in ISO/IEC 19592-2 and employ the following algorithms and parameters.

— Message space: the set of possible messages that can be input to the message sharing algorithm.

— Share space: the set of possible shares that can be output by the message sharing algorithm.

— Number of shares: the range of possible values of n supported by the scheme.

— Threshold: the range of possible values of k supported by the scheme.

— Adversary structure: the set of all maximal coalitions of participants that are not sufficient to reconstruct the message. For a threshold secret sharing scheme with threshold k, the adversary structure $\mathcal{A}$ is $\{Z \mid Z \subset \{1,\ldots,n\}, |Z| = k-1\}$.

— Message sharing algorithm: an algorithm that divides a message into n shares.

— Message reconstruction algorithm: an algorithm that reconstructs a message from k shares.

— Lagrange interpolation coefficients: the coefficients used in the reconstruction algorithm of the Shamir secret sharing scheme.

5.3 Secure multiparty computation based on secret sharing

The secure multiparty computation schemes based on secret sharing specified in this document are intended to be used for performing a secure function evaluation. The process of a secure function evaluation is as follows.

a) Input parties run the message sharing algorithm on their function inputs and then send the resulting shares to the computing parties.

b) The computing parties evaluate the function using one or more of the multiparty protocols specified in this document.

c) The computing parties send the result of the evaluation to the result parties, and the result parties then run the message reconstruction algorithm to obtain the function output.

NOTE The notions of input parties, computing parties, result parties, and multiparty protocols are defined in ISO/IEC 4922-1.

The following parameters apply to all the mechanisms specified in this document.

— Input message space: the same as the message space of the secret sharing scheme (see 5.2).

— Output message space: the same as the message space of the secret sharing scheme (see 5.2).

— Encoded message space: the same as the share space of the secret sharing scheme (see 5.2).

— Restriction of roles: there are no restrictions on the roles of a party, i.e. one party can take multiple roles.

— Communication channel: a point-to-point secure channel between each pair of parties.

Clauses 6, 7 and 8 specify mechanisms for computing parties that can be used to build a multiparty protocol for secure function evaluation. For each mechanism, the following items are listed.

d) Parameters

1) Number of computing parties: the number of computing parties n', supported by the protocol. In this document, n is used instead of n' since all mechanisms specified in this document assume that each computing party holds a single share, meaning that n equals to n'.

   2) Threshold: the number of computing parties k' such that even against an adversary corrupting fewer than k' parties, the input privacy defined in ISO/IEC 4922-1 holds. In this document, k is used instead of k' since all the mechanisms specified in this document assume that each computing party holds a single share, meaning that k equals to k'.

   3) Other parameters (if applicable).

e) Protocol description: the protocol that jointly computes a function on the input shares among the computing parties.

f) Properties

   1) Communication complexity: the total number of elements communicated among the computing parties.

   2) Round complexity: the number of communication rounds, where communication is as parallelized as possible.

   3) Tolerable adversary behaviour: the type of adversary against which the protocol will remain secure. The protocols specified in this document are secure against either passive adversaries (adversaries that only observe the protocol execution), or active adversaries (adversaries can interrupt or modify communications).

6 Addition, subtraction, and multiplication by a constant

6.1 General

This clause contains protocols which achieve secure multiparty computation for addition, subtraction, addition and subtraction of a constant, and multiplication by a constant based on the Shamir and replicated additive secret sharing schemes. These protocols involve only local computations, i.e. they do not require communication. Therefore, discussion of communication and round complexities is omitted in this clause. The protocols are a detailed description of the homomorphic operations of the secret sharing schemes described in ISO/IEC 19592-2.

6.2 Addition

6.2.1 Addition for the Shamir secret sharing scheme

6.2.1.1 Parameters

Number of computing parties: n, satisfying $n<|K|$.

Threshold: k, satisfying $k \leq n$.

6.2.1.2 Addition protocol

Input: share vectors $([a]_1,\ldots,[a]_n),([a']_1,\ldots,[a']_n) \in K^n$.

Output: share vector $([a+a']_1,\ldots,[a+a']_n) \in K^n$.

a) Each P_i for $1 \leq i \leq n$ computes $[a+a']_i = [a]_i + [a']_i \in K$.

b) Output $([a+a']_1,\ldots,[a+a']_n) \in K^n$.

6.2.1.3 Properties

Tolerable adversary behaviour: active if $k-1<\frac{n}{2}$, otherwise passive.

6.2.2 Addition of a constant for the Shamir secret sharing scheme

6.2.2.1 Parameters

Number of computing parties: n, satisfying $n<|K|$.

Threshold: k, satisfying $k\le n$.

6.2.2.2 Addition-of-a-constant protocol

Input: a share vector $([a]_1,\dots,[a]_n)\in K^n$, and a constant $c\in K$.

Output: share vector $([a+c]_1,\dots,[a+c]_n)\in K^n$.

a) Each P_i for $1\le i\le n$ computes $[a+c]_i=[a]_i+c\in K$.

b) Output $([a+c]_1,\dots,[a+c]_n)\in K^n$.

6.2.2.3 Properties

Tolerable adversary behaviour: active if $k-1<\frac{n}{2}$, otherwise passive.

6.2.3 Addition for the replicated additive secret sharing scheme

6.2.3.1 Parameters

Number of computing parties: n.

Threshold: k, satisfying $k\le n$.

Form of shares: $[a]_i=\{r_Z\,|\,i\notin Z\in \boldsymbol{A},\ 1\le i\le n\}$ and $[a']_i=\left\{r'_Z\,|\,i\notin Z\in \boldsymbol{A},\ 1\le i\le n\right\}$.

6.2.3.2 Addition protocol

Input: share vectors $([a]_1,\dots,[a]_n),([a']_1,\dots,[a']_n)\in G^{m\times n}$, where $m={}_{n-1}\mathrm{C}_{k-1}$.

Output: share vector $([a+a']_1,\dots,[a+a']_n)\in G^{m\times n}$.

a) Each P_i for $1\le i\le n$ computes $[a+a']_i=\{r_Z+r'_Z\,|\,i\notin Z\in \boldsymbol{A}\}\in G^m$.

b) Output $([a+a']_1,\dots,[a+a']_n)\in G^{m\times n}$.

6.2.3.3 Properties

Tolerable adversary behaviour: active if $k-1<\frac{n}{2}$, otherwise passive.

6.2.4 Addition of a constant for the replicated additive secret sharing scheme

6.2.4.1 Parameters

Number of computing parties: n.

Threshold: k, satisfying $k \leq n$.

Form of shares: $[a]_i = \{r_Z | i \notin Z \in \mathcal{A}, 1 \leq i \leq n\}$.

Representative share: a sub-share r_{Z^*} to be used in a special way in the protocol, where $Z^* \in \mathcal{A}$ is a set of participants that shall be agreed by the parties prior to the protocol execution.

6.2.4.2 Addition-of-a-constant protocol

Input: share vector $([a]_1, \ldots, [a]_n) \in G^{m \times n}$ and a constant $c \in G$, where $m = {}_{n-1}C_{k-1}$.

Output: share vector $([a+c]_1, \ldots, [a+c]_n) \in G^{m \times n}$.

a) Each P_i for $1 \leq i \leq n$ sets $[a+c]_i = \{r_{Z'} | i \notin Z' \in \mathcal{A}, 1 \leq i \leq n\}$, where $r_{Z'} = r_Z + c \in G$ if $Z' = Z^*$, otherwise $r_{Z'} = r_Z$.

b) Output $([a+c]_1, \ldots, [a+c]_n) \in G^{m \times n}$.

6.2.4.3 Properties

Tolerable adversary behaviour: active if $k - 1 < \frac{n}{2}$, otherwise passive.

6.3 Subtraction

6.3.1 Subtraction for the Shamir secret sharing scheme

6.3.1.1 Parameters

Number of computing parties: n, satisfying $n < |K|$.

Threshold: k, satisfying $k \leq n$.

6.3.1.2 Subtraction protocol

Input: share vectors $([a]_1, \ldots, [a]_n), ([a']_1, \ldots, [a']_n) \in K^n$.

Output: share vector $([a-a']_1, \ldots, [a-a']_n) \in K^n$.

a) Each P_i for $1 \leq i \leq n$ computes $[a-a']_i = [a]_i + (-[a']_i) \in K$, where $-[a']_i$ is an additive inverse of $[a']_i$ in K.

b) Output $([a-a']_1, \ldots, [a-a']_n) \in K^n$.

6.3.1.3 Properties

Tolerable adversary behaviour: active if $k - 1 < \frac{n}{2}$, otherwise passive.

6.3.2 Subtraction of a constant for the Shamir secret sharing scheme

6.3.2.1 Parameters

Number of computing parties: n, satisfying $n < |K|$.

Threshold: k, satisfying $k \leq n$.

6.3.2.2 Subtraction-of-a-constant protocol

Input: a share vector $([a]_1,\ldots,[a]_n)\in K^n$, and a constant $c\in K$.

Output: share vector $([a-c]_1,\ldots,[a-c]_n)\in K^n$.

a) Each P_i for $1\le i\le n$ computes $[a-c]_i=[a]_i-c\in K$.

b) Output $([a-c]_1,\ldots,[a-c]_n)\in K^n$.

6.3.2.3 Properties

Tolerable adversary behaviour: active if $k-1<\frac{n}{2}$, otherwise passive.

6.3.3 Subtraction for the replicated additive secret sharing scheme

6.3.3.1 Parameters

Number of computing parties: n.

Threshold: k, satisfying $k\le n$.

Form of shares: $[a]_i=\{r_Z \mid i\notin Z\in \boldsymbol{A}, 1\le i\le n\}$ and $[a']_i=\{r'_Z \mid i\notin Z\in \boldsymbol{A}, 1\le i\le n\}$.

6.3.3.2 Subtraction protocol

Input: share vectors $([a]_1,\ldots,[a]_n),([a']_1,\ldots,[a']_n)\in G^{m\times n}$, where $m={}_{n-1}\mathrm{C}_{k-1}$.

Output: share vector $([a-a']_1,\ldots,[a-a']_n)\in G^{m\times n}$.

a) Each P_i for $1\le i\le n$ computes $[a-a']_i=\{r_Z+(-r'_Z) \mid i\notin Z\in \boldsymbol{A}\}\in G^m$, where $-r'_Z\in G$ is an additive inverse of $r'_Z\in G$.

b) Output $([a-a']_1,\ldots,[a-a']_n)\in G^{m\times n}$.

6.3.3.3 Properties

Tolerable adversary behaviour: active if $k-1<\frac{n}{2}$, otherwise passive.

6.3.4 Subtraction of a constant for the replicated additive secret sharing scheme

6.3.4.1 Parameters

Number of computing parties: n.

Threshold: k, satisfying $k\le n$.

Form of shares: $[a]_i=\{r_Z \mid i\notin Z\in \boldsymbol{A}, 1\le i\le n\}$.

Representative share: a sub-share r_{Z^*} to be used in a special way in the protocol, where $Z^*\in \boldsymbol{A}$ is a set of participants that shall be agreed by the parties prior to the protocol execution.

6.3.4.2 Subtraction-of-a-constant protocol

Input: share vector $([a]_1,\ldots,[a]_n)\in G^{m\times n}$ and a constant $c\in G$, where $m={}_{n-1}\mathrm{C}_{k-1}$.

Output: share vector $([a-c]_1, \dots, [a-c]_n) \in G^{m \times n}$.

a) Each P_i for $1 \le i \le n$ sets $[a-c]_i = \{r_{Z'} \mid i \notin Z' \in \mathcal{A}, 1 \le i \le n\}$, where $r_{Z'} = r_Z - c \in G$ if $Z' = Z^*$ and $r_{Z'} = r_Z$ otherwise.

b) Output $([a-c]_1, \dots, [a-c]_n) \in G^{m \times n}$.

6.3.4.3 Properties

Tolerable adversary behaviour: active if $k-1 < \frac{n}{2}$, otherwise passive.

6.4 Multiplication by a constant

6.4.1 Multiplication by a constant for the Shamir secret sharing scheme

6.4.1.1 Parameters

Number of computing parties: n, satisfying $n < |K|$.

Threshold: k, satisfying $k \le n$.

6.4.1.2 Multiplication by a constant protocol

Input: share vector $([a]_1, \dots, [a]_n) \in K^n$, and a constant $c \in K$.

Output: share vector $([ca]_1, \dots, [ca]_n) \in K^n$.

a) Each P_i for $1 \le i \le n$ computes $[ca]_i = c[a]_i \in K$.

b) Output $([ca]_1, \dots, [ca]_n) \in K^n$.

6.4.1.3 Properties

Tolerable adversary behaviour: active if $k-1 < \frac{n}{2}$, otherwise passive.

6.4.2 Multiplication by a constant for the replicated additive secret sharing scheme

6.4.2.1 Parameters

Number of computing parties: n.

Threshold: k, satisfying $k \le n$.

Form of share: $[a]_i = \{r_Z \mid i \notin Z \in \mathcal{A}\}$.

6.4.2.2 Multiplication by a constant protocol

Input: share vector $([a]_1, \dots, [a]_n) \in G^{m \times n}$ where $m = {}_{n-1}\mathrm{C}_{k-1}$, and a constant c such that $0 \le c \le |G|-1$.

Output: share vector $([ca]_1, \dots, [ca]_n) \in G^{m \times n}$.

a) Each P_i for $1 \le i \le n$ computes $[ca]_i = \{cr_Z \mid i \notin Z \in \mathcal{A}\} \in G^m$, where $cr_Z \in G$ means recursive $c-1$ additions of r_Z as $r_Z + r_Z + \dots + r_Z$ when $1 \le c \le |G|-1$, or the substitution by 0 when $c = 0$.

b) Output $([ca]_1, \dots, [ca]_n) \in G^{m \times n}$.

NOTE If G is a ring and $c \in G$, each P_i can compute cr_Z by multiplying c by r_Z in a).

6.4.2.3 Properties

Tolerable adversary behaviour: active if $k-1<\frac{n}{2}$, otherwise passive.

7 Shared random number generation

7.1 General

This clause contains secure multiparty computation protocols for generating shares of a random number. No computing party shall know the generated shared random number.

7.2 Information-theoretically secure shared random number generation

7.2.1 General-purpose shared random number generation scheme

7.2.1.1 General

This subclause contains the parameters (7.2.1.2), protocol (7.2.1.3) and properties (7.2.1.4) of the information-theoretically secure shared random number generation for secret sharing schemes with homomorphic operations, as described in Reference [4]. The protocol works on secret sharing schemes with homomorphic operations, including the Shamir and replicated additive secret sharing schemes. In this protocol, k random numbers are chosen by k distinct computing parties and shared using the message sharing algorithm. The parties then sum their received shares and output the result.

NOTE The number of random numbers k is equal to the threshold of the secret sharing scheme.

7.2.1.2 Parameters

Number of computing parties: n, which shall satisfy $n<|K|$ if the Shamir secret sharing scheme is used.

Threshold: k, satisfying $k \leq n$.

Subset of parties: $\{P_{i_1},\ldots,P_{i_k}\} \subseteq \{P_1,\ldots,P_n\}$.

7.2.1.3 Shared random number generation protocol

Input: none.

Output: share vector $([w]_1,\ldots,[w]_n)$.

a) Each P_i for $P_i \in \{P_{i_1},\ldots,P_{i_k}\}$

   1) randomly selects w_i',

   2) computes $([w_i']_1,\ldots,[w_i']_n)=\mathrm{Share}(w_i')$,

   3) sends $[w_i']_j$ to each party P_j for $1 \leq j \leq n$.

b) Compute $[w]=\left[\sum_{t=1}^{k} w_{i_t}'\right]$ by addition described in 6.2.

c) Output $([w]_1,\ldots,[w]_n)$.

7.2.1.4 Properties

Communication complexity: $k(n-1)$ elements in K for the Shamir secret sharing scheme and $km(n-1)$ elements in G for the replicated additive secret sharing.

Round complexity: 1 round.

Tolerable adversary behaviour: passive.

7.2.2 Shared random number generation for the replicated additive secret sharing scheme

7.2.2.1 General

This subclause contains the parameters (7.2.2.2), protocol (7.2.2.3) and properties (7.2.2.4) of the information-theoretically secure shared random number generation for the replicated additive secret sharing scheme. The protocol is optimized for the replicated additive secret sharing scheme and has lower communication complexity than the protocol specified in 7.2.1.

7.2.2.2 Parameters

Number of computing parties: n.

Threshold: k, satisfying $k \leq n$.

Subset of parties: a party P_Z such that $P_Z = P_{i_Z}$ for some $i_Z \notin Z$ for each $Z \in \boldsymbol{A}$.

7.2.2.3 Shared random number generation protocol

Input: none.

Output: a share vector $([w]_1, \ldots, [w]_n) \in G^{m \times n}$, where $m = {}_{n-1}\mathrm{C}_{k-1}$.

a) P_Z randomly selects $r_Z \in G$ for each $Z \in \boldsymbol{A}$.

b) P_Z sends r_Z to P_j for all $j \notin Z \cup \{i_Z\}$ for each $Z \in \boldsymbol{A}$.

c) P_j for $1 \leq j \leq n$ sets $[w]_j = \{r_Z \mid j \notin Z \in \boldsymbol{A}\}$.

d) Output $([w]_1, \ldots, [w]_n) \in G^{m \times n}$.

7.2.2.4 Properties

Communication complexity: $\sum_{Z \in A} (n - |Z| - 1)$ elements of G.

Round complexity: 1 round.

Tolerable adversary behaviour: passive.

7.2.3 Shared random number generation for the Shamir secret sharing scheme

7.2.3.1 General

This subclause contains the parameters (7.2.3.2), protocol (7.2.3.3) and properties (7.2.3.4) of the information-theoretically secure shared random number generation protocol[10] for the Shamir secret sharing scheme. This protocol generates share vectors for $n-k+1$ random numbers simultaneously from share vectors for n random numbers chosen by the computing parties. The protocol has low communication complexity when generating multiple share vectors of a random number.

7.2.3.2 Parameters

Number of computing parties: n, satisfying $n < |K|$.

Threshold: k, satisfying $k \leq n$.

Vandermonde matrix: M, where M is an $n \times (n-k+1)$ matrix with i-th row $\left(1, a_i, a_i^2, \ldots, a_i^{n-k}\right)$ with the a_i distinct and non-zero, where a_i for $1 \leq i \leq n$ shall be agreed by the parties prior to executing the protocol.

7.2.3.3 Shared random number generation protocol

Input: none.

Output: share vectors $([w_1]_1, \ldots, [w_1]_n), \ldots, ([w_{n-k+1}]_1, \ldots, [w_{n-k+1}]_n) \in K^n$.

a) Each P_i for $1 \leq i \leq n$:

   1) randomly selects $w_i' \in K$,

   2) computes $([w_i']_1, \ldots, [w_i']_n) = \mathrm{Share}(w_i')$,

   3) sends $[w_i']_j$ to P_j for $1 \leq j \leq n$.

b) Each P_j for $1 \leq j \leq n$:

   1) receives all $[w_1']_j, \ldots, [w_n']_j$ from P_i for $1 \leq i \leq n$,

   2) computes $\left([w_1]_j, \ldots, [w_{n-k+1}]_j\right)^{\mathrm{T}} = M^{\mathrm{T}} \left([w_1']_j, \ldots, [w_n']_j\right)^{\mathrm{T}}$.

c) Output $([w_1]_1, \ldots, [w_1]_n), \ldots, ([w_{n-k+1}]_1, \ldots, [w_{n-k+1}]_n) \in K^n$.

7.2.3.4 Properties

Communication complexity: $n(n-1)$ elements of K.

Round complexity: 1 round.

Tolerable adversary behaviour: passive.

7.3 Computationally secure shared random number generation

7.3.1 General

7.3 describes the parameters, protocols and properties of the computationally secure shared random number generation protocols.[9] The security of these protocols depends on the computational hardness assumption of the deterministic random bit generator. The protocols consist of two phases.

a) Seed sharing phase: The parties share seeds among adequate sets of parties. This phase requires communication.

b) Shared random number generation phase: The parties generate a share vector of a random number. This phase can be performed without communication.

The parties execute the seed sharing phase first and then execute the random share generation phase repeatedly until the seeds are updated. The seed sharing phase is common between the Shamir and replicated additive secret sharing schemes.

7.3.2 Seed sharing phase

7.3.2.1 General

The seed sharing phase is desired to provide the computing parties with a replicated seed vector $(S_1,\ldots,S_n)\in X^{m\times n}$, where X is a seed space and $S_i=\{s_Z \mid i\notin Z\in \boldsymbol{A}\}$ for a random number s_Z. The replicated seed vector can be regarded as a share vector of a random number in the replicated additive secret sharing scheme in *X*. Therefore, the protocols specified in 7.2.2 can be used to implement this phase.

7.3.2.2 Parameters

Number of computing parties: n.

Threshold: k, satisfying $k\leq n$.

Subset of parties: a party P_Z such that $P_Z=P_{i_Z}$ for some $i_Z\notin Z$ for each $Z\in \boldsymbol{A}$.

7.3.2.3 Seed sharing phase

Input: none.

Output: a share vector $(S_1,\ldots,S_n)\in X^{m\times n}$, where $m={}_{n-1}\mathrm{C}_{k-1}$.

a) P_Z randomly selects $s_Z\in X$ for each $Z\in \boldsymbol{A}$.

b) P_Z sends s_Z to P_j for all $j\notin Z\cup\{i_Z\}$ for each $Z\in \boldsymbol{A}$.

c) P_j for $1\leq j\leq n$ sets $S_j=\{s_Z \mid j\notin Z\in \boldsymbol{A}\}$.

d) Output $(S_1,\ldots,S_n)\in X^{m\times n}$.

NOTE There are alternative ways. An input party can create all the s_Z and send them to the corresponding computing parties, where the input party is expected to be different from any computing party, or the computing parties can get together offline and share the s_Z.

7.3.2.4 Properties

Communication complexity: $\sum_{Z\in A}(n-|Z|-1)$ elements of X.

Round complexity: 1 round.

Tolerable adversary behaviour: passive.

7.3.3 Shared random number generation phase for the replicated additive secret sharing scheme

7.3.3.1 Parameters

Number of computing parties: n.

Threshold: k, satisfying $k\leq n$.

Seed space: a group *X*.

Deterministic random bit generator: DRBG takes a seed in *X* and state *t* as inputs and outputs a pseudo-random element in *G* and an updated state *t'*.

NOTE Both the input and output of an ordinary DRBG are bit strings, but a DRBG with group element output can be constructed following guidance in ISO/IEC 18031:2011, Annex B.

7.3.3.2 Shared random number generation phase

Input: a replicated seeds vector $(S_1,\ldots,S_n)\in X^{m\times n}$, where $m={}_{n-1}\mathrm{C}_{k-1}$ and $S_i=\{s_Z \mid i\notin Z\in \mathcal{A}\}$.

Output: a share vector $([w]_1,\ldots,[w]_n)\in G^{m\times n}$.

a) Each P_i for $1\leq i\leq n$:

   1) computes $(r_Z,t')=\mathrm{DRBG}(s_Z,t)$ for all $s_Z\in S_i$.

   2) computes $[w]_i=\{r_Z \mid i\notin Z\in \mathcal{A}\}$.

   3) sets t' as a new state of DRBG.

b) Output $([w]_1,\ldots,[w]_n)\in G^{m\times n}$.

NOTE 1 The output is a share vector of a random value $w=\sum_{Z\in \mathcal{A}} r_Z$.

NOTE 2 DRBG requires a state t as an input and the security strength. It is assumed that t is predetermined and renewed by t' for each execution. The formal definition and concrete examples of DRBGs are given in ISO/IEC 18031. The security strength can be determined by referring to Reference [2].

7.3.3.3 Properties

Tolerable adversary behaviour: active.

7.3.4 Shared random number generation phase for the Shamir secret sharing scheme

7.3.4.1 Parameters

Number of computing parties: n, satisfying $n\leq|K|$.

Threshold: k, satisfying $k\leq n$.

Seed space: a group X.

Deterministic random bit generator: DRBG takes a seed in X and state t as input and outputs a pseudo-random element in K and an updated state t'.

Other parameters: for each $Z\in \mathcal{A}$, the polynomial $f_Z(x)$ of degree $k-1$ such that $f_Z(0)=1$ and $f_Z(x_i)=0$ for all $i\in Z$.

NOTE The polynomial $f_Z(x)$ can be obtained by using the Lagrange interpolation as $f_Z(x)=\prod_{i\in Z}\frac{x_i-x}{x_i}$.

7.3.4.2 Shared random number generation phase

Input: a replicated seed vector $(S_1,\ldots,S_n)\in X^{m\times n}$.

Output: a share vector $([w]_1,\ldots,[w]_n)\in K^n$.

a) Each P_i for $1\leq i\leq n$:

   1) computes $(r_Z,t')=\mathrm{DRBG}(s_Z,t)$ for all $s_Z\in S_i$.

   2) sets t' as a new state.

   3) computes $[w]_i=\sum_{Z;i\notin Z\in \mathcal{A}} r_Z f_Z(x_i)$.

b) Output $([w]_1,\ldots,[w]_n)\in K^n$.

NOTE 1 The output is a share vector of a random value $w = \sum_{Z \in A} r_Z$.

NOTE 2 DRBG requires a state t as an input and the security strength. It is assumed that t is predetermined and renewed by t' for each execution. The formal definition and concrete examples of DRBGs are given in ISO/IEC 18031. The security strength can be determined by referring to Reference [2].

7.3.4.3 Properties

Tolerable adversary behaviour: active.

8 Multiplication

8.1 General

This clause contains secure multiparty computation protocols for multiplication based on the Shamir and replicated additive secret sharing schemes. In this clause, the message and share space of the replicated additive secret sharing scheme is assumed to be a finite ring R to define multiplication. The secret sharing scheme used in each protocol is specified as a scheme in the parameters. Part of the protocols supports the dot product[7] that outputs the sum of element-wise multiplications of share-vectors with the same communication and round complexities as the (plain) multiplication protocol.

8.2 GRR-multiplication for the Shamir secret sharing scheme

8.2.1 General

This clause describes the parameters (8.2.2), protocol (8.2.3), and properties (8.2.4) of the Gennaro-Rabin-Rabin (GRR)-multiplication.[11] It is an information-theoretically secure multiplication protocol for the Shamir secret sharing scheme.

8.2.2 Parameters

Scheme: the Shamir secret sharing scheme.

Number of computing parties: n, satisfying $n < |K|$.

Threshold: k, satisfying $2k - 1 \le n$.

Subset of parties: $\{P_{i_1}, \ldots, P_{i_{2k-1}}\} \subseteq \{P_1, \ldots, P_n\}$.

Other parameters: Lagrange interpolation coefficients $\lambda_{i_t} = \prod_{u=1, u \ne t}^{2k-1} \frac{(0 - x_{i_u})}{(x_{i_t} - x_{i_u})}$ so that $aa' = \sum_{t=1}^{2k-1} \lambda_{i_t} [a]_{i_t} [a']_{i_t}$, where x_{i_u} is defined as a non-zero fixed field element corresponding to P_{i_u} in the subset of parties. All the parties compute the Lagrange interpolation coefficients before the step b) in the mechanisms described in 8.2.3 and 8.2.4.

8.2.3 Multiplication protocol

Input: share vectors $([a]_1, \ldots, [a]_n), ([a']_1, \ldots, [a']_n) \in K^n$.

Output: share vector $([aa']_1, \ldots, [aa']_n) \in K^n$.

a) Each P_i for $i = i_1, \ldots, i_{2k-1}$:

1) computes $d_i = [a]_i [a']_i \in K$,

2) computes $([d_i]_1,\ldots,[d_i]_n) = \mathrm{Share}(d_i)$,

3) sends $[d_i]_j$ to P_j for $1 \le j \le n$.

b) Compute $[\lambda_i d_i]$ for all $i = i_1,\ldots,i_{2k-1}$ by multiplication by a constant described in 6.4.1.

c) Compute $[aa'] = \left[\sum_{t=1}^{2k-1} \lambda_{i_t} d_{i_t}\right]$ by addition described in 6.2.1.

d) Output $([aa']_1,\ldots,[aa']_n)$.

8.2.4 Dot product protocol

Input: share vector $\{([a_1]_1,\ldots,[a_\ell]_1),\ldots,([a_1]_n,\ldots,[a_\ell]_n)\},\{([a'_1]_1,\ldots,[a'_\ell]_1),\ldots,([a'_1]_n,\ldots,[a'_\ell]_n)\} \in K^{\ell \times n}$.

Output: share vector $([c]_1,\ldots,[c]_n) \in K^n$ such that $c = \sum_{u=1}^{\ell} a_u a'_u$.

a) Each P_i for $i = i_1,\ldots,i_{2k-1}$:

1) computes $d_i = \sum_{u=1}^{\ell} [a_u]_i [a'_u]_i \in K$,

2) computes $([d_i]_1,\ldots,[d_i]_n) = \mathrm{Share}(d_i)$,

3) sends $[d_i]_j$ to P_j for $1 \le j \le n$.

b) Compute $[\lambda_i d_i]$ for all $i = i_1,\ldots,i_{2k-1}$ by multiplication by a constant described in 6.4.1.

c) Compute $[c]_j = \left[\sum_{t=1}^{2k-1} \lambda_{i_t} d_{i_t}\right]$ for $1 \le j \le n$ by addition described in 6.2.1.

d) Output $([c]_1,\ldots,[c]_n)$.

NOTE There is an alternative way, where each P_i for $i = i_1,\ldots,i_{2k-1}$ multiplies $[d_i]_j$ by λ_i in a) 3), and P_j for $1 \le j \le n$ computes the sum of received shares in b).

8.2.5 Properties

Communication complexity: $(n-1)(2k-1)$ elements of K.

Round complexity: 1 round.

Tolerable adversary behaviour: passive.

8.3 DN-multiplication for the Shamir secret sharing scheme

8.3.1 General

This subclause describes the parameters (8.3.2), protocols (8.3.3 and 8.3.4) and properties (8.3.4) of the Damgård-Nielsen (DN)-multiplication.[10] It is an information-theoretically and passively secure multiplication protocol based on the Shamir secret sharing scheme. It requires two share vectors of a secret random number r as an additional input but is more efficient in communication complexity than GRR-multiplication for large n.

NOTE The additional input can be given by an input party or generated by computing parties executing the shared random number generation protocols specified in 7.2.1.3 or 7.2.3.3 twice. If an input party gives the additional input, the input party cannot become a computing party to keep the random number secret from the computing parties. If computing parties generate the additional input, the computing parties can obtain the two share-vectors of the same random number by using the same random numbers in 7.2.1.3 and 7.2.3.3 step a) 1).

8.3.2 Parameters

Scheme: the Shamir secret sharing scheme.

Number of computing parties: n, satisfying $n<|K|$.

Threshold: k, satisfying $2k-1\leq n$.

Subset of parties: $\{P_{i_1},\ldots,P_{i_{2k-1}}\}\subseteq\{P_1,\ldots,P_n\}$.

Other parameter: A party upon agreement $P_A\in\{P_1,\ldots,P_n\}$.

8.3.3 Multiplication protocol

Input: share vectors $([a]_1,\ldots,[a]_n),([a']_1,\ldots,[a']_n)\in K^n$, and additional shared random number vectors $\left([w]_1^{(k)},\ldots,[w]_n^{(k)}\right)$ and $\left([w]_1^{(2k-1)},\ldots,[w]_n^{(2k-1)}\right)$, where $[w]_i^{(k)}$ and $[w]_i^{(2k-1)}$ denotes a share vector of w with the threshold k and $2k-1$, respectively.

Output: share vector $([aa']_1,\ldots,[aa']_n)\in K^n$.

a) Each P_i for $i=i_1,\ldots,i_{2k-1}$:

   1) computes $b_i=[a]_i[a']_i+[w]_i^{(2k-1)}\in K$,

   2) sends b_i to P_A.

b) P_A computes $b=\text{Recover}\left(b_{i_1},\ldots,b_{i_{2k-1}}\right)$.

c) P_A sends b to P_j for $1\leq j\leq n$.

d) Set $[aa']=-[w-b]$ as an additive inverse of $[w-b]=[w]^{(k)}-b$ computed by subtraction of a constant described in 6.3.2.

e) Output $([aa']_1,\ldots,[aa']_n)\in K^n$.

8.3.4 Dot product protocol

Input: share vector $\{([a_1]_1,\ldots,[a_\ell]_1),\ldots,([a_1]_n,\ldots,[a_\ell]_n)\},\{([a'_1]_1,\ldots,[a'_\ell]_1),\ldots,([a'_1]_n,\ldots,[a'_\ell]_n)\}\in K^{\ell\times n}$, and additional share vectors $\left([w]_1^{(k)},\ldots,[w]_n^{(k)}\right)$ and $\left([w]_1^{(2k-1)},\ldots,[w]_n^{(2k-1)}\right)$.

Output: share vector $([c]_1,\ldots,[c]_n)\in K^n$ such that $c=\sum_{u=1}^{\ell}a_u a'_u$.

a) Each P_i for $i=i_1,\ldots,i_{2k-1}$:

   1) computes $b_i=\sum_{u=1}^{\ell}[a_u]_i[a'_u]_i+[w]_i^{(2k-1)}\in K$,

   2) sends b_i to P_A.

b) P_A computes $b=\text{Recover}\left(b_{i_1},\ldots,b_{i_{2k-1}}\right)$.

c) P_A sends b to P_j for $1\leq j\leq n$.

d) Set $[c]=-[w-b]$ as an additive inverse of $[w-b]=[w]^{(k)}-b$ computed by subtraction of a constant described in 6.3.2.

e) Output $([c]_1,\ldots,[c]_n)\in K^n$.

8.3.5 Properties

Communication complexity: $n+2k-2$ elements in K.

Round complexity: 2 rounds.

Tolerable adversary behaviour: passive.

8.4 CHIKP-multiplication for the replicated additive secret sharing scheme

8.4.1 General

This subclause describes the parameters (8.4.2), protocol (8.4.3), and properties (8.4.4) of Chida-Hamada-Ikarashi-Kikuchi-Pinkas (CHIKP)-multiplication.[5,8] It is an information-theoretically and passively secure multiplication protocol based on the replicated additive secret sharing scheme, where the threshold and the number of computing parties are limited to two and three, respectively. The protocol requires a share vector of a secret random number as an additional input.

NOTE The additional input can be given by an input party or generated by computing parties executing the shared random number generation protocols specified in 7.2.1.3, 7.2.2.3 or 7.3.3.2. If an input party gives the additional input, the input party cannot become a computing party to keep the random number secret from the computing parties.

8.4.2 Parameters

Scheme: the replicated additive secret sharing scheme.

Share space: finite ring R.

Number of computing parties: 3.

Threshold: 2.

Adversary structure: $\mathcal{A}=\{\{1\},\{2\},\{3\}\}$.

Form of shares: $[x]_1=(x_1,x_2),[x]_2=(x_2,x_3),[x]_3=(x_3,x_1)\in R^2$ where $x=x_1+x_2+x_3\in R$.

8.4.3 Multiplication protocol

Input: share vectors $([a]_1,[a]_2,[a]_3),([a']_1,[a']_2,[a']_3)$, and an additional shared random number vector $([w]_1,[w]_2,[w]_3)\in R^{2\times 3}$.

Output: share vector $([aa']_1,[aa']_2,[aa']_3)\in R^{2\times 3}$.

a) Each P_i for $i\in\{1,2,3\}$:

   1) computes $b_{i,i+1}=(a_i+a_{i+1})(a'_i+a'_{i+1})-a_ia'_i+w_i-w_{i+1}$,

   2) sends $b_{i,i+1}$ to P_{i+1} and receives $b_{i-1,i}$ from P_{i-1},

   3) sets $[aa']_i=(b_{i-1,i},b_{i,i+1})\in R^2$.

b) Output $([aa']_1,[aa']_2,[aa']_3)\in R^{2\times 3}$.

8.4.4 Properties

Communication complexity: 3 elements of R.

Round complexity: 1 round.

Tolerable adversary behaviour: passive.

8.5 Beaver-multiplication

8.5.1 General

This subclause describes the parameters (8.5.2), protocol (8.5.3), and properties (8.5.4) of Beaver-multiplication.[4] It is an information-theoretically and passively secure multiplication protocol based on both the Shamir and replicated additive secret sharing schemes. It requires share vectors of random numbers w, w', and ww', all of which are unknown to the computing parties, as additional inputs.

NOTE The additional input can be given by an input party. If an input party gives the additional input, the input party cannot become a computing party to keep the random number secret from the computing parties.

8.5.2 Parameters

Scheme: the Shamir or replicated additive secret sharing scheme.

Share space: finite field K for the Shamir secret sharing scheme, and finite ring R for the replicated additive secret sharing scheme.

Number of computing parties: n, satisfying $n < |K|$ only if the Shamir secret sharing scheme is used.

Threshold: k, satisfying $k \leq n$.

Subset of parties: $\{P_{i_1}, \ldots, P_{i_k}\} \subseteq \{P_1, \ldots, P_n\}$.

Other parameter: A party upon agreement $P_A \in \{P_1, \ldots, P_n\}$.

8.5.3 Multiplication protocol

Input: share vectors $([a]_1, \ldots, [a]_n), ([a']_1, \ldots, [a']_n)$, additional shared random number vectors $([w]_1, \ldots, [w]_n), ([w']_1, \ldots, [w']_n)$, and an additional shared vector $([ww']_1, \ldots, [ww']_n)$.

Output: share vector $([aa']_1, \ldots, [aa']_n)$.

a) Each P_i for $i = i_1, \ldots, i_k$:

  1) computes $[b]_i = [a]_i + [w]_i$ and $[b']_i = [a']_i + [w']_i$,

  2) sends $[b]_i$ and $[b']_i$ to P_A.

b) P_A computes $b = \text{Recover}([b]_{i_1}, \ldots, [b]_{i_k})$ and $b' = \text{Recover}([b']_{i_1}, \ldots, [b']_{i_k})$.

c) P_A sends b and b' to P_i for $1 \leq i \leq n$.

d) Compute $[a'b]$ and $[ab']$ by multiplication by a constant described in 6.4.

e) Compute $[aa'] = [ww' + a'b + ab' - bb']$ by addition and subtraction of a constant described in 6.2.

f) Output $([aa']_1, \ldots, [aa']_n)$.

8.5.4 Properties

Communication complexity: $2(n+k-1)$ elements of K and $2(n+km-1)$ elements of R if the Shamir and replicated additive secret sharing schemes are used, respectively.

Round complexity: 2 rounds.

Tolerable adversary behaviour: passive.

9 Secure function evaluation

This clause shows how to securely compute a function represented as a combination of arithmetic operations in a multiparty computation based on secret sharing. An arithmetic circuit consists of addition, subtraction, multiplication-by-a-constant, and multiplication gates. The parties shall agree on a function to be evaluated, a secret sharing scheme, the number of shares, the threshold, and other relevant parameters beforehand. Using protocols to compute the function on shares, the parties can securely compute any arithmetic circuit as follows.

a) Input parties run the message sharing algorithm (Share) on their inputs and then send the shares of the inputs to the computing parties.

b) The computing parties evaluate the function using a multiparty protocol. If the function is represented as an arithmetic circuit, the function can be evaluated by repeatedly executing addition, addition of a constant, subtraction, subtraction of a constant, multiplication-by-a-constant, and multiplication protocols specified in Clauses 6, 7 and 8.

c) The computing parties send the result of the evaluation to result parties, and then the result parties run the message reconstruction algorithm (Recover), to obtain the function output.

Annex A
(normative)

Object identifiers

This annex lists the object identifiers assigned to the secure multiparty computation mechanisms based on secret sharing specified in this document.

```
Secure-multiparty-computation-mechanisms-based-on-secret-sharing{
iso(1) standard(0)   secure-multiparty-computation(4922)   mechanisms-based-on-secret-
sharing(2)
asn1-module(0) object-identifiers(0) }
DEFINITIONS EXPLICIT TAGS::= BEGIN
-- EXPORTS All; --
-- IMPORTS None; --
OID::= OBJECT IDENTIFIER -- Alias
-- Synonyms --
id-mpc-ss OID::= {
iso(1) standard(0)   secure-multiparty-computation(4922)   mechanisms-based-on-secret-
sharing(2)}
-- Assignments --
id-mpc-ss-1 OID::= { id-mpc-ss local(1) }
id-mpc-ss-2 OID::= { id-mpc-ss rand(2) }
id-mpc-ss-3 OID::= { id-mpc-ss mult(3) }
-- Linear operation --
id-mpc-ss-1-1 OID::= { id-mpc-ss-1 add(1) }
id-mpc-ss-1-2 OID::= { id-mpc-ss-1 sub(2) }
id-mpc-ss-1-3 OID::= { id-mpc-ss-1 mult-by-constant(3) }
-- Addition --
id-mpc-ss-1-1-1 OID::= { id-mpc-ss-1-1 shamir-add(1) }
id-mpc-ss-1-1-2 OID::= { id-mpc-ss-1-1 shamir-add-of-constant(2) }
id-mpc-ss-1-1-3 OID::= { id-mpc-ss-1-1 replicated-add(3) }
id-mpc-ss-1-1-4 OID::= { id-mpc-ss-1-1 replicated-add-of-constant(2) }
-- Subtraction --
id-mpc-ss-1-2-1 OID::= { id-mpc-ss-1-2 shamir-sub(1) }
id-mpc-ss-1-2-2 OID::= { id-mpc-ss-1-2 shamir-sub-of-constant(2) }
id-mpc-ss-1-2-3 OID::= { id-mpc-ss-1-2 replicated-sub(3) }
id-mpc-ss-1-2-4 OID::= { id-mpc-ss-1-2 replicated-sub-of-constant(4) }
-- Multiplication by a constant --
id-mpc-ss-1-3-1 OID::= { id-mpc-ss-1-3 shamir-mult-by-constant(1) }
id-mpc-ss-1-3-2 OID::= { id-mpc-ss-1-3 replicated-mult-by-constant(2) }
-- Shared random number generation --
id-mpc-ss-2-1 OID::= { id-mpc-ss-2 it-rand(1) }
id-mpc-ss-2-2 OID::= { id-mpc-ss-2 comp-rand(2) }
-- Information-theoretically secure shared random number generation --
id-mpc-ss-2-1-1 OID::= { id-mpc-ss-2-1 hom-it-rand(1) }
id-mpc-ss-2-1-2 OID::= { id-mpc-ss-2-1 replicated-it-rand(2) }
id-mpc-ss-2-1-3 OID::= { id-mpc-ss-2-1 shamir-it-rand(3) }
-- Computationally secure shared random number generation --
id-mpc-ss-2-2-1 OID::= { id-mpc-ss-2-2 seed(1) }
id-mpc-ss-2-2-2 OID::= { id-mpc-ss-2-2 replicated-comp-rand(2) }
id-mpc-ss-2-2-3 OID::= { id-mpc-ss-2-2 shamir-comp-rand(3) }
-- Multiplication --
id-mpc-ss-3-1 OID::= { id-mpc-ss-3 grr(1) }
id-mpc-ss-3-2 OID::= { id-mpc-ss-3 dn(2) }
id-mpc-ss-3-3 OID::= { id-mpc-ss-3 chikp(3) }
id-mpc-ss-3-4 OID::= { id-mpc-ss-3 beaver(4) }
-- GRR --
id-mpc-ss-3-1-1 OID::= { id-mpc-ss-3-1 grr-mult(1) }
id-mpc-ss-3-1-2 OID::= { id-mpc-ss-3-1 grr-dot-prod(2) }
-- DN --
id-mpc-ss-3-2-1 OID::= { id-mpc-ss-3-2 dn-mult(1) }
id-mpc-ss-3-2-2 OID::= { id-mpc-ss-3-2 dn-dot-prod(2) }
-- CHIKP --
id-mpc-ss-3-3-1 OID::= { id-mpc-ss-3-3 chikp-mult(1) }
```

```
-- Beaver --
id-mpc-ss-3-4-1 OID::= { id-mpc-ss-3-4 beaver-mult(1) }
END -- secure-multiparty-computation-mechanisms-based-on-secret-sharing --
```

Annex B
(informative)

Numerical examples

B.1 Common parameters and share examples

B.1.1 General

The following parameters and share examples are used for numerical examples.

B.1.2 Shamir secret sharing scheme

Parameters:

— Finite field K is a field of a prime order $p = 2^{61} - 1$

— $(k, n) = (2, 3)$

— $(x_1, x_2, x_3) = (2, 3, 4)$

Share examples:

a) $a =$ `0x0000000000000100`

$([a]_1, [a]_2, [a]_3) =$ `(0x14722c1bc0ca1ee9, 0x0eab4229a12f2dde, 0x08e4583781943cd3)`

Random coefficient (of the first-degree term of the polynomial used in sharing):

`0x1a39160de0650ef4`

b) $a' =$ `0x0000000000000050`

$([a']_1, [a']_2, [a']_3) =$ `(0x01746f1c18b85a07, 0x122ea6aa251486e2, 0x02e8de383170b3be)`

Random coefficient: `0x10ba378e0c5c2cdb`

B.1.3 Replicated additive secret sharing scheme

Parameters:

— Finite group G is a cyclic finite group of an order $p = 2^{64}$

— $(k, n) = (2, 3)$

— $\boldsymbol{A} = \{Z \mid Z \subset \{1, \ldots, n\}, |Z| = 1\} = \{\{1\}, \{2\}, \{3\}\}$

Share examples:

a) $b =$ `0x0000000000000100`

$[b]_1 = (r_{\{2\}}, r_{\{3\}}) =$ `(0x10ba528baa79794d, 0x99cc3c534b4e6bdd)`

$[b]_2 = (r_{\{3\}}, r_{\{1\}}) =$ `(0x99cc3c534b4e6bdd, 0x557971210a381bd6)`

$[b]_3 = (r_{\{1\}}, r_{\{2\}}) =$ `(0x557971210a381bd6, 0x10ba528baa79794d)`

b) $b' =$ `0x0000000000000050`

$[b']_1 = \left(r_{\{2\}}, r_{\{3\}}\right) =$ `(0xa5fb9c848074a05d, 0x1ad0e8a1d95f00ce)`

$[b']_2 = \left(r_{\{3\}}, r_{\{1\}}\right) =$ `(0x1ad0e8a1d95f00ce, 0x3f337ad9a62c5f25)`

$[b']_3 = \left(r_{\{1\}}, r_{\{2\}}\right) =$ `(0x3f337ad9a62c5f25, 0xa5fb9c848074a05d)`

B.2 Addition, subtraction, and multiplication by a constant

B.2.1 Addition for the Shamir secret sharing scheme

Input: $[a]$ and $[a']$

Output: $\left([a+a']_1, [a+a']_2, [a+a']_3\right) =$ `(0x15e69b37d98278f0, 0x00d9e8d3c643b4c1, 0x0bcd366fb304f091)`

B.2.2 Addition to a constant for the Shamir secret sharing scheme

Input: $[a]$ and a constant $c =$ `0x0000000000000050`

Output: $\left([a+c]_1, [a+c]_2, [a+c]_3\right) =$ `(0x14722c1bc0ca1f39, 0x0eab4229a12f2e2e, 0x08e4583781943d23)`

B.2.3 Addition for the replicated secret sharing scheme

Input: $[b]$ and $[b']$

Output:

$[b+b']_1 =$ `(0xb6b5ef102aee19aa, 0xb49d24f524ad6cab)`

$[b+b']_2 =$ `(0xb49d24f524ad6cab, 0x94acebfab0647afb)`

$[b+b']_3 =$ `(0x94acebfab0647afb, 0xb6b5ef102aee19aa)`

B.2.4 Addition to a constant for the replicated secret sharing scheme

Input: $[b]$ and a constant $c =$ `0x0000000000000050`

Output:

$[b+c]_1 =$ `(0x10ba528baa79799d, 0x99cc3c534b4e6bdd)`

$[b+c]_2 =$ `(0x99cc3c534b4e6bdd, 0x557971210a381bd6)`

$[b+c]_3 =$ `(0x557971210a381bd6, 0x10ba528baa79799d)`

B.2.5 Subtraction for the Shamir secret sharing scheme

Input: $[a]$ and $[a']$

Output: $\left([a-a']_1, [a-a']_2, [a-a']_3\right) =$ `(0x12fdbcffa811c4e2, 0x1c7c9b7f7c1aa6fb, 0x05fb79ff50238915)`

B.2.6 Subtraction of a constant for the Shamir secret sharing scheme

Input: $[a]$ and a constant $c =$ `0x0000000000000050`

Output: $\left([a-c]_1, [a-c]_2, [a-c]_3\right) =$ `(0x14722c1bc0ca1e99, 0x0eab4229a12f2d8e, 0x08e4583781943c83)`

B.2.7 Subtraction for the replicated secret sharing scheme

Input: $[b]$ and $[b']$

Output:

$[b-b']_1 =$ `(0x6abeb6072a04d8f0, 0x7efb53b171ef6b0f)`

$[b-b']_2 =$ `(0x7efb53b171ef6b0f, 0x1645f647640bbcb1)`

$[b-b']_3 =$ `(0x1645f647640bbcb1, 0x6abeb6072a04d8f0)`

B.2.8 Subtraction of a constant for the replicated secret sharing scheme

Input: $[b]$ and a constant $c =$ `0x0000000000000050`

Output:

$[b-c]_1 =$ `(0x10ba528baa7978fd, 0x99cc3c534b4e6bdd)`

$[b-c]_2 =$ `(0x99cc3c534b4e6bdd, 0x557971210a381bd6)`

$[b-c]_3 =$ `(0x557971210a381bd6, 0x10ba528baa7978fd)`

B.2.9 Multiplication by a constant for the Shamir secret sharing scheme

Input: $[a]$ and a constant $c =$ `0x0000000000000003`

Output: $([ca]_1, [ca]_2, [ca]_3) =$ `(0x1d568453425e5cbc, 0x0c01c67ce38d899b, 0x1aad08a684bcb679)`

B.2.10 Multiplication by a constant for the replicated secret sharing scheme

Input: $[b]$ and a constant $c =$ `0x0000000000000003`

Output:

$[cb]_1 =$ `(0x322ef7a2ff6c6be7, 0xcd64b4f9e1eb4397)`

$[cb]_2 =$ `(0xcd64b4f9e1eb4397, 0x006c53631ea85382)`

$[cb]_3 =$ `(0x006c53631ea85382, 0x322ef7a2ff6c6be7)`

B.3 Shared random number generation

B.3.1 General-purpose shared random number generation scheme

Random numbers and their shares:

$w'_1 =$ `0x0fb8b15f8cc757d1`

$([w'_1]_1, [w'_1]_2, [w'_1]_3) =$ `(0x0303e0bafe8b1508, 0x0ca97868b76cf3a3, 0x164f1016704ed23e)`

Random coefficients: `0x09a597adb8e1de9b`

$w'_2 =$ `0x1f64f2672f0d4013`

$([w'_2]_1, [w'_2]_2, [w'_2]_3) =$ `(0x0ebd0f99f6f8da02, 0x16691e335aeea6f9, 0x1e152cccbee473f0)`

Random coefficients: `0x07ac0e9963f5ccf7`

$w'_3 =$ `0x03640488ded60fe0`

$([w'_3]_1, [w'_3]_2, [w'_3]_3) =$ `(0x1ada929646fbf710, 0x0695d99cfb0eeaa9, 0x125120a3af21de41)`

Random coefficients: `0x0bbb4706b412f398`

Output: $([w]_1, [w]_2, [w]_3) =$ `(0x0c9b82eb3c7fe61b, 0x09a870390d6a8546, 0x06b55d86de552471)`

B.3.2 Shared random number generation for the replicated additive secret sharing scheme

Random numbers and shares:

$r_{\{1\}} =$ `0xc2f79b9b76593dd3`

$r_{\{2\}} =$ `0x5c9fd9754d70ca65`

$r_{\{3\}} =$ `0xa3a646ff6aab5d8b`

Output:

$[w]_1 = (r_{\{2\}}, r_{\{3\}}) =$ `(0x5c9fd9754d70ca65, 0xa3a646ff6aab5d8b)`

$[w]_2 = (r_{\{3\}}, r_{\{1\}}) =$ `(0xa3a646ff6aab5d8b, 0xc2f79b9b76593dd3)`

$[w]_3 = (r_{\{1\}}, r_{\{2\}}) =$ `(0xc2f79b9b76593dd3, 0x5c9fd9754d70ca65)`

B.3.3 Shared random number generation for the Shamir secret sharing scheme

Random numbers and shares:

$w'_1 =$ `0x0fb8b15f8cc757d1`

$([w'_1]_1, [w'_1]_2, [w'_1]_3) =$ `(0x0303e0bafe8b1508, 0x0ca97868b76cf3a3, 0x164f1016704ed23e)`

Random coefficients: `0x09a597adb8e1de9b`

$w'_2 =$ `0x1f64f2672f0d4013`

$([w'_2]_1, [w'_2]_2, [w'_2]_3) =$ `(0x0ebd0f99f6f8da02, 0x16691e335aeea6f9, 0x1e152cccbee473f0)`

Random coefficients: `0x07ac0e9963f5ccf7`

$w'_3 =$ `0x03640488ded60fe0`

$([w'_3]_1, [w'_3]_2, [w'_3]_3) =$ `(0x1ada929646fbf710, 0x0695d99cfb0eeaa9, 0x125120a3af21de41)`

Random coefficients: `0x0bbb4706b412f398`

Vandermonde matrix: $M = \begin{bmatrix} 1 & 5 \\ 1 & 6 \\ 1 & 7 \end{bmatrix}$

Output:

$([w_1]_1, [w_1]_2, [w_1]_3) =$ `(0x0c9b82eb3c7fe61b, 0x09a870390d6a8546, 0x06b55d86de552471)`

$([w_2]_1, [w_2]_2, [w_2]_3) =$ `(0x037bc35eb37046ad, 0x13df028a942116ab, 0x044241b674d1e6aa)`

B.3.4 Computationally secure shared random number generation

B.3.4.1 Seed sharing phase

Parameter: Each seed s_i for $i \in \{\{1\},\{2\},\{3\}\}$ is 256 bits.