Blockchain e Smart contract per giuristi 3.0

Biagio Distefano

Ethereum

Nella Blockchain Ethereum, oltre a operazioni lineari come “Da Tizio a Caio 10 Ether”, è possibile eseguire operazioni complesse, condizionali, che costituiscono la c.d. business logic

Queste operazioni complesse vengono eseguite da tutti i nodi (peer) della rete nella propria Ethereum Virtual Machine (EVM), e vengono validate e inserite nella blockchain (circa) come le transazioni normali

Smart Contracts

Le istruzioni delle operazioni complesse sono contenute in
portafogli speciali

I portafogli speciali che contengono questa logica sono detti
Smart Contracts

Smart Contracts

Gli **Smart Contract** vengono scritti in un linguaggio di programmazione che viene tradotto in istruzioni per la **EVM**

Una volta scritto e compilato lo **Smart Contract**, questo viene immesso nella **Blockchain** (cd. *deployment*) e gli verrà assegnato un suo indirizzo portafogli

Conoscendone l'indirizzo, è possibile interagire con lo **Smart Contract** attraverso il proprio portafogli

Smart Contracts

Scriveremo il nostro **Smart Contract** in un linguaggio di programmazione chiamato
Solidity

Per scrivere lo smart contract useremo un ambiente di sviluppo integrato (integrated development environment, IDE) chiamato **Remix** direttamente sul browser

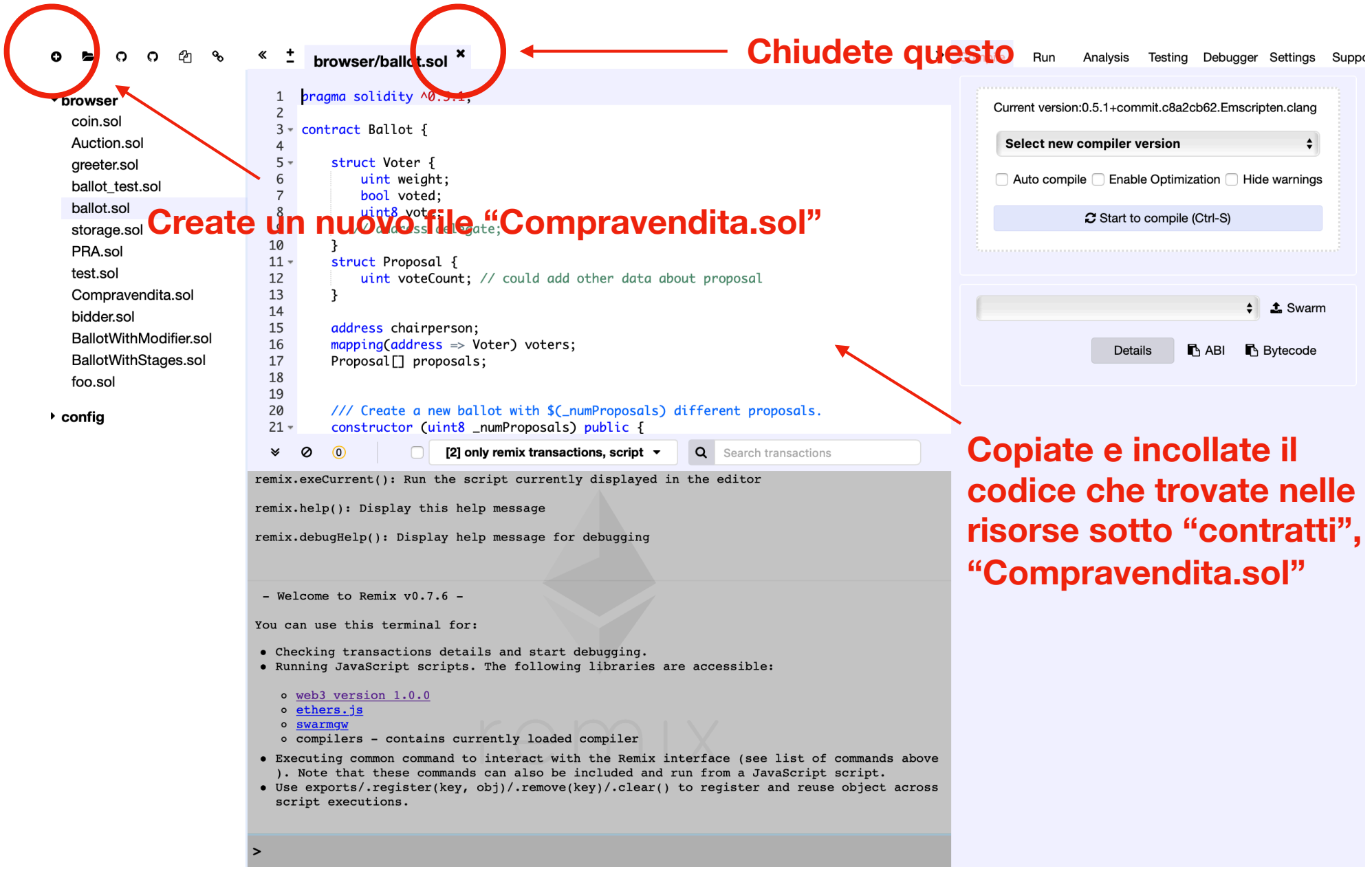
Andate su

http://remix.ethereum.org/

Trovate le risorse su

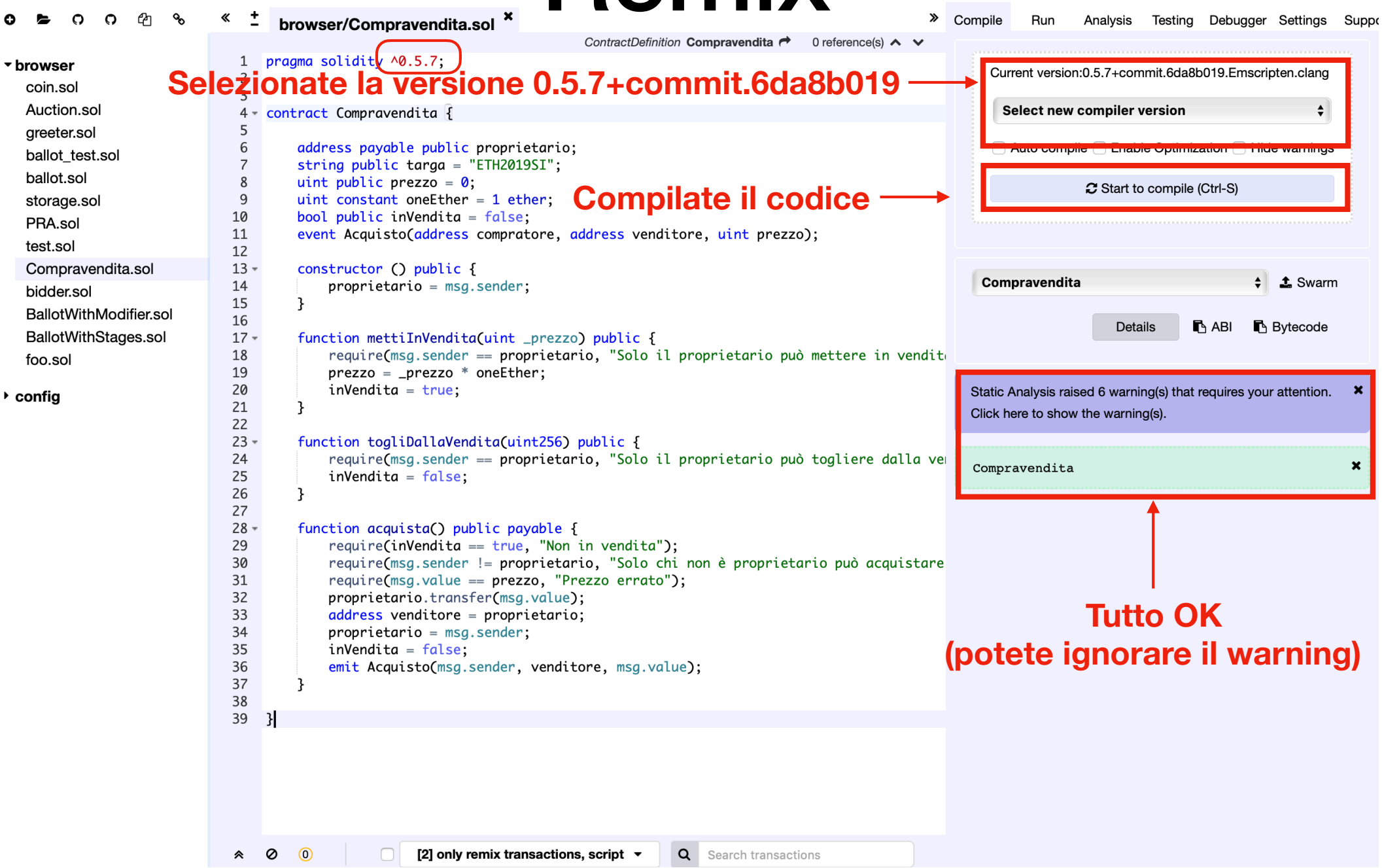
https://github.com/raptored01/siena-2019-bc-sc

Remix




```
pragma solidity ^0.5.1;

contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount; // could add other data about proposal
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;


    /// Create a new ballot with $(_numProposals) different proposals.
    constructor (uint8 _numProposals) public {
```

Remix


browser/Compravendita.sol
Compile
Run
Analysis
Testing
Debugger
Settings
browser
coin.sol
Auction.sol
greeter.sol
ballot_test.sol
ballot.sol
storage.sol
PRA.sol
test.sol
Compravendita.sol
bidder.sol
BallotWithModifier.sol
BallotWithStages.sol
foo.sol
config
ContractDefinition Compravendita
0 reference(s)
pragma solidity ^0.5.7;
Selezionate la versione 0.5.7+commit.6da8b019
contract Compravendita {
address payable public proprietario;
string public targa = "ETH2019SI";
uint public prezzo = 0;
uint constant oneEther = 1 ether;
bool public inVendita = false;
event Acquisto(address compratore, address venditore, uint prezzo);
constructor () public {
proprietario = msg.sender;
}
function mettiInVendita(uint _prezzo) public {
require(msg.sender == proprietario, "Solo il proprietario può mettere in vendit
prezzo = _prezzo * oneEther;
inVendita = true;
}
function togliDallaVendita(uint256) public {
require(msg.sender == proprietario, "Solo il proprietario può togliere dalla ve
inVendita = false;
}
function acquista() public payable {
require(inVendita == true, "Non in vendita");
require(msg.sender != proprietario, "Solo chi non è proprietario può acquistare
require(msg.value == prezzo, "Prezzo errato");
proprietario.transfer(msg.value);
address venditore = proprietario;
proprietario = msg.sender;
inVendita = false;
emit Acquisto(msg.sender, venditore, msg.value);
}
}
Current version:0.5.7+commit.6da8b019.Emscripten.clang
Select new compiler version
Start to compile (Ctrl-S)
Compile codice
Compravendita
Swarm
Details
ABI
Bytecode
Static Analysis raised 6 warning(s) that requires your attention.
Click here to show the warning(s).
Compravendita
Tutto OK
(potete ignorare il warning)
[2] only remix transactions, script
Search transactions

Remix


browser/Compravendita.sol
Compile
Run
Analysis
Testing
Debugger
Settings
browser
coin.sol
Auction.sol
greeter.sol
ballot_test.sol
ballot.sol
storage.sol
PRA.sol
test.sol
Compravendita.sol
bidder.sol
BallotWithModifier.sol
BallotWithStages.sol
foo.sol
config
Scegliete il portafogli con cui effettuare le operazioni
ContractDefinition Compravendita
0 reference(s)
contract Compravendita {
address payable public proprietario;
string public targa = "ETH2019SI";
uint public prezzo = 0;
uint constant oneEther = 1 ether;
bool public inVendita = false;
event Acquisto(address compratore, address venditore, uint prezzo);
constructor () public {
proprietario = msg.sender;
}
Immettete il contratto nella blockchain
function mettiInVendita(uint _prezzo) public {
require(msg.sender == proprietario, "Solo il proprietario può mettere in vendit
prezzo = _prezzo * oneEther;
inVendita = true;
}
function togliDallaVendita(uint256) public {
require(msg.sender == proprietario, "Solo il proprietario può togliere dalla ve
inVendita = false;
}
function acquista() public payable {
require(inVendita == true, "Non in vendita");
Chiamate le funzioni del contratto
[2] only remix transactions, script
Search transactions
ethers.js
swarmgw
compilers - contains currently loaded compiler
Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.
Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
Vedete i log del contratto
creation of Compravendita pending...
[vm] from:0xca3...a733c to:Compravendita.(constructor) value:0 wei data:0x608...90029 logs:0 hash:0x2d3...b1fb2
Debug
Environment
JavaScript VM
Account
0xca3...a733c (99.999999999999310
Gas limit
3000000
Value
0
wei
Compravendita
Deploy
or
At Address
Load contract from Address
Transactions recorded: 1
Deployed Contracts
Compravendita at 0x692...77b3a (memory)
acquista
mettiInVendita
uint256 _prezzo
togliDallaVendita
uint256
inVendita
prezzo
proprietario
targa
Interrogate le variabili pubbliche del contratto

Usiamo uno Smart Contract su una Blockchain reale (Ethereum Ropsten Testnet)

Una versione smart del PRA

Il contratto si trova all'indirizzo
0xD9c453dC11773866e4f89b65A34164ACfb4C2dab
(cliccate sull'indirizzo per vederlo su etherscan)

L'applicazione web si trova qui: http://smartsiena.ddns.net

Decreto Semplificazioni (d.l. 14 dicembre 2018, n. 135) convertito in legge con modificazioni dalla L. 12/2019

Art. 8-ter
Tecnologie basate su registri distribuiti e smart contract

1. Si definiscono "tecnologie basate su registri distribuiti" le tecnologie e i protocolli informatici che usano un registro condiviso, distribuito, replicabile, accessibile simultaneamente, architetturalmente decentralizzato su basi crittografiche, tali da consentire la registrazione, la convalida, l'aggiornamento e l'archiviazione di dati sia in chiaro che ulteriormente protetti da crittografia verificabili da ciascun partecipante, non alterabili e non modificabili.

2. **Si definisce "smart contract" un programma per elaboratore che opera su tecnologie basate su registri distribuiti** e la cui esecuzione vincola automaticamente due o più parti sulla base di effetti predefiniti dalle stesse. **Gli smart contract soddisfano il requisito della forma scritta** previa identificazione informatica delle parti interessate, attraverso un processo avente i requisiti fissati dall'Agenzia per l'Italia digitale con linee guida da adottare entro novanta giorni dalla data di entrata in vigore della legge di conversione del presente decreto.

3. La memorizzazione di un documento informatico attraverso l'uso di tecnologie basate su registri distribuiti produce gli effetti giuridici della validazione temporale elettronica di cui all'articolo 41 del regolamento (UE) n. 910/2014 del Parlamento europeo e del Consiglio, del 23 luglio 2014.

4. Entro novanta giorni dalla data di entrata in vigore della legge di conversione del presente decreto, l'Agenzia per l'Italia digitale individua gli standard tecnici che le tecnologie basate su registri distribuiti debbono possedere ai fini della produzione degli effetti di cui al comma 3.